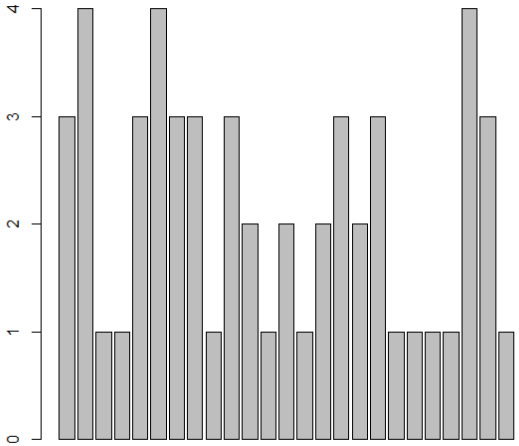
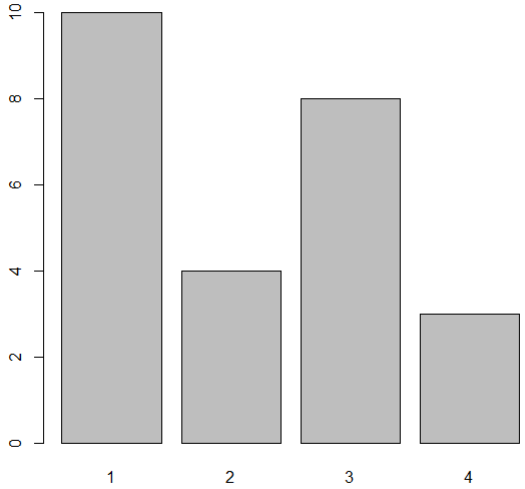
데이터 사이언스 과제2

< Section 3: Univariate Data >

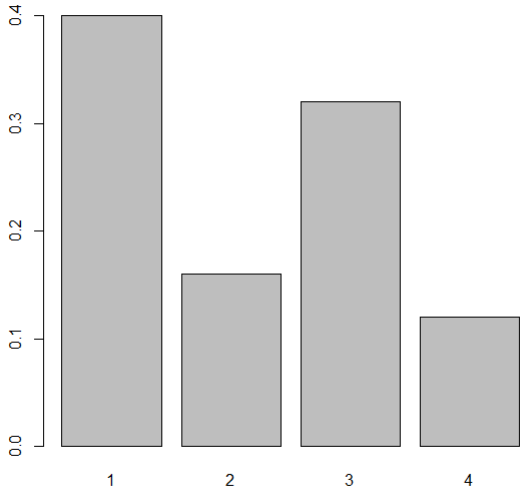
```
> x=c("Yes","No","No","Yes","Yes")
> table(x)
x
 No Yes
  2   3
>
> x=c("Yes","No","No","Yes","Yes")
> x # print out values in x
[1] "Yes" "No"  "No"  "Yes" "Yes"
> factor(x)# print out value in factor(x)
[1] Yes No  No  Yes Yes
Levels: No Yes
>
> beer = scan()
1: 3 4 1 1 3 4 3 3 1 3 2 1 2 1 2 3 2 3 1 1 1 1 4 3 1
26:
Read 25 items
> barplot(beer) # this isn't correct
```




```
> barplot(table(beer)) # Yes, call with summarized data
```

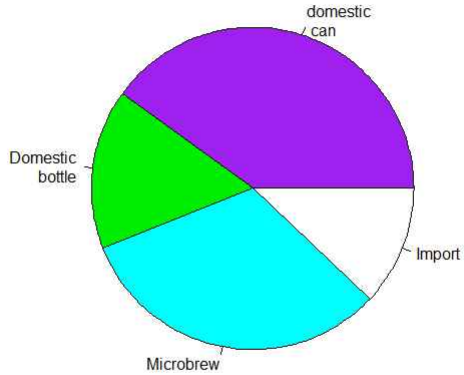



```
> barplot(table(beer)/length(beer)) # divide by n for proportion
```



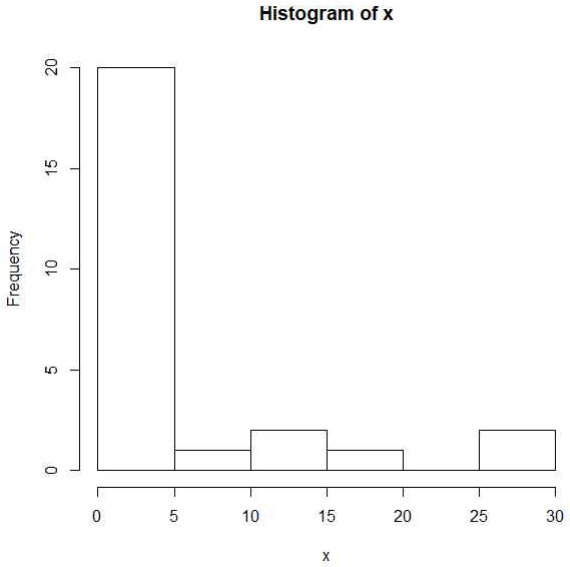

```
> table(beer)/length(beer)
beer
   1    2    3    4
0.40 0.16 0.32 0.12
>
> beer.counts = table(beer) # store the table result
> pie(beer.counts) # first pie -- kind of dull
```

```
> names(beer.counts) = c("domestic\n can","Domestic\n bottle",
"Microbrew","Import") # give names
> pie(beer.counts) # prints out names
> pie(beer.counts,col=c("purple","green2","cyan","white")) # now with colors
```

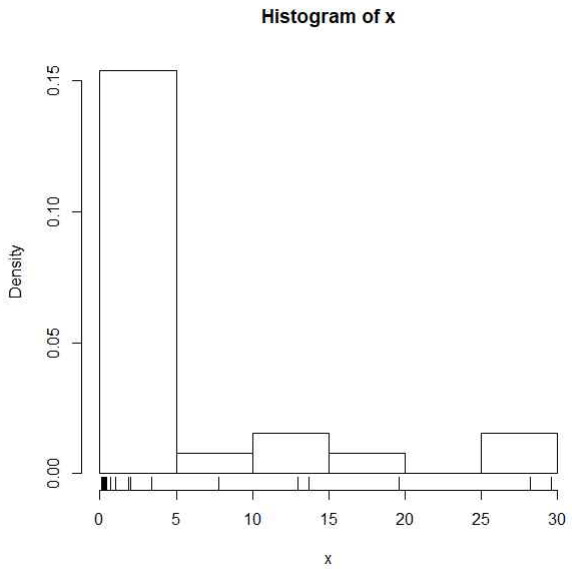



```
> sals = scan() # read in with scan
1: 12 .4 5 2 50 8 3 1 4 0.25
11:
Read 10 items
> mean(sals) # the average
[1] 8.56
> var(sals) # the variance
[1] 226
> sd(sals) # the standard deviation
[1] 15
> median(sals) # the median
[1] 3.5
> fivenum(sals) # min, lower hinge, Median, upper hinge, max
[1]  0.25  1.00  3.50  8.00 50.00
> summary(sals)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.2     1.2     3.5     8.6     7.2    50.0
>
> data=c(10, 17, 18, 25, 28, 28)
> summary(data)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   10.0    17.2    21.5    21.0    27.2    28.0
> quantile(data,.25)
 25%
17.2
> quantile(data,c(.25,.75)) # two values of p at once
 25%  75%
17.2 27.2
>
> sort(sals)
 [1]  0.25  0.40  1.00  2.00  3.00  4.00  5.00  8.00 12.00 50.00
> fivenum(sals) # note 1 is the 3rd value, 8 the 8th.
[1]  0.25  1.00  3.50  8.00 50.00
> summary(sals) # note 3.25 value is 1/4 way between 1 and 2
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.2     1.2     3.5     8.6     7.2    50.0
>
> mean(sals,trim=1/10) # trim 1/10 off top and bottom
[1] 4.42
> mean(sals,trim=2/10)
[1] 3.83
>
> IQR(sals)
[1] 6
>
> mad(sals)
[1] 4.15
>
> median(abs(sals - median(sals))) # without normalizing constant
[1] 2.8
> median(abs(sals - median(sals)))*1.4826
[1] 4.15
>
> scores = scan()
1: 2 3 16 23 14 12 4 13 2 0 0 0 6 28 31 14 4 8 2 5
21:
Read 20 items
> apropos("stem") # What exactly is the name?
[1] "R_system_version" "stem"              "system"
[4] "system.file"      "system.time"       "system2"
> stem(scores)

  The decimal point is 1 digit(s) to the right of the |

  0 | 000222344568
  1 | 23446
  2 | 38
  3 | 1

>
> stem(scores,scale=2)

  The decimal point is 1 digit(s) to the right of the |

  0 | 000222344
  0 | 568
  1 | 2344
  1 | 6
  2 | 3
```

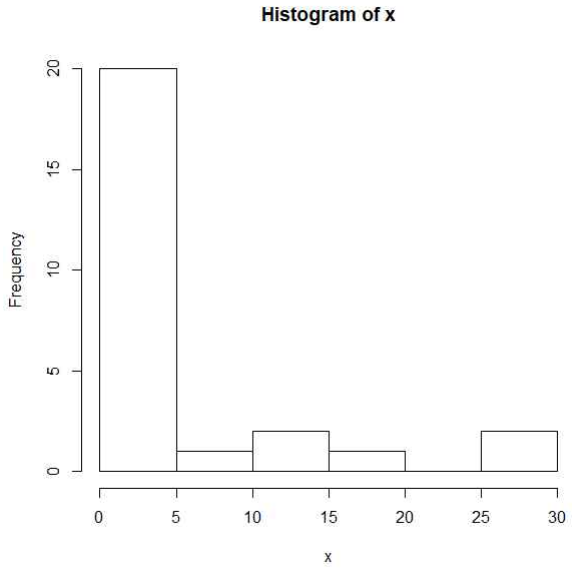
```
  2 | 8
  3 | 1

>
> sals = c(12, .4, 5, 2, 50, 8, 3, 1, 4, .25) # enter data
> cats = cut(sals,breaks=c(0,1,5,max(sals))) # specify the breaks
> cats # view the values
 [1] (5,50] (0,1]  (1,5]  (1,5]  (5,50] (5,50] (1,5]  (0,1]  (1,5]  (0,1]
Levels: (0,1] (1,5] (5,50]
> table(cats) # organize
cats
 (0,1]  (1,5] (5,50]
     3      4      3
> levels(cats) = c("poor","rich","rolling in it") # change labels
> table(cats)
cats
          poor           rich rolling in it
             3              4              3
>
> x=scan()
1: 29.6 28.2 19.6 13.7 13.0 7.8 3.4 2.0 1.9 1.0 0.7 0.4 0.4 0.3 0.3
16: 0.3 0.3 0.3 0.2 0.2 0.2 0.1 0.1 0.1 0.1 0.1
27:
Read 26 items
> hist(x) # frequencies
```



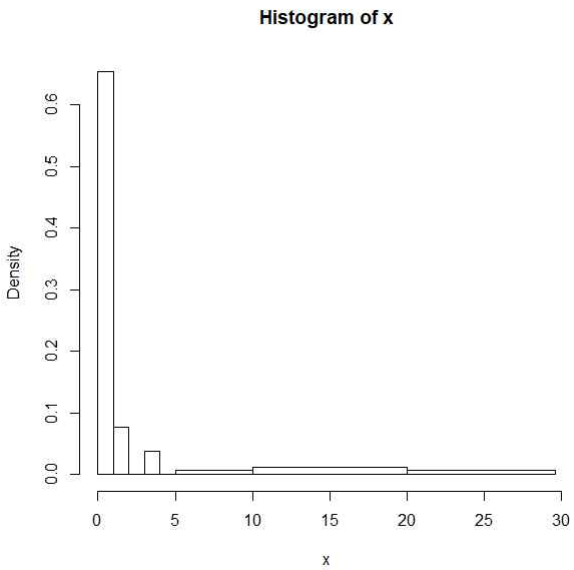

```
> hist(x,probability=TRUE) # proportions (or probabilities)
> rug(jitter(x)) # add tick marks
```



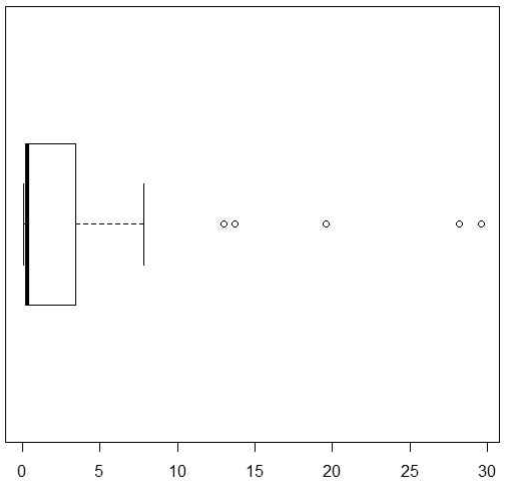

```
> hist(x,breaks=10) # 10 breaks, or just hist(x,10)
```



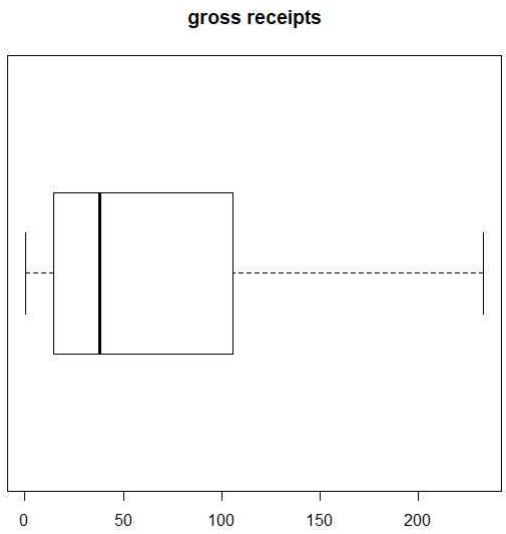

```
> hist(x,breaks=c(0,1,2,3,4,5,10,20,max(x))) # specify break points
```



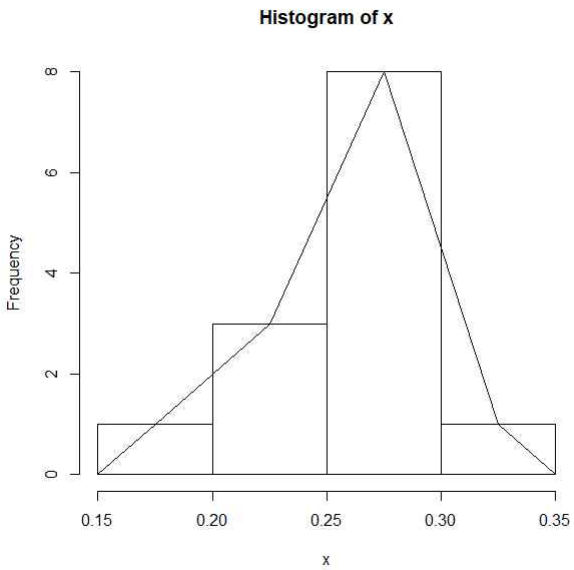

```
> library("UsingR") # read in library for these notes
> data(movies) # read in data set for gross.
> names(movies)
[1] "title"     "current"   "previous" "gross"
> attach(movies) # to access the names above
> boxplot(current,main="current receipts",horizontal=TRUE)
```



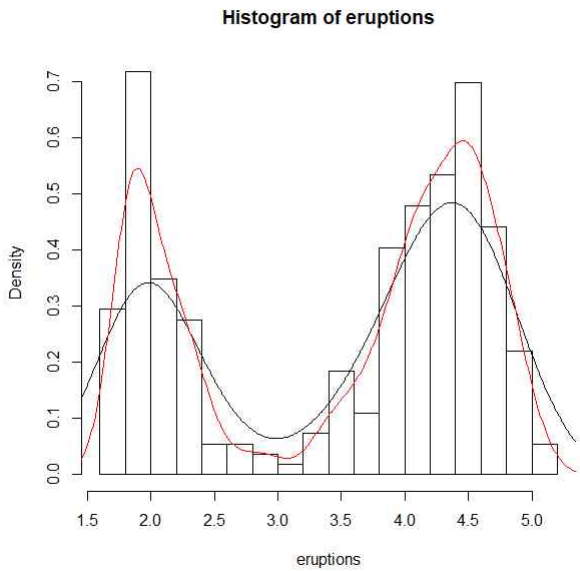

```
> boxplot(gross,main="gross receipts",horizontal=TRUE)
```




```
> detach(movies) # tidy up
>
> #install.packages("ts")
> library("ts") # load the library
Error in library("ts") : 'ts'이라고 불리는 패키지가 없습니다
> data("lynx") # load the data
> summary(lynx) # Just what is lynx?
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
     39     348     771    1538    2567    6991
>
> x = c(.314,.289,.282,.279,.275,.267,.266,.265,.256,.250,.249,.211,.161)
> tmp = hist(x) # store the results
>
lines(c(min(tmp$breaks),tmp$mids,max(tmp$breaks)),c(0,tmp$counts,0),type="
l")
```



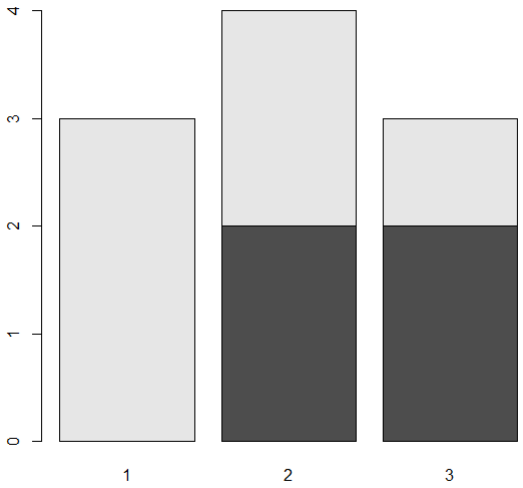

```
> data(faithful)
> attach(faithful) # make eruptions visible
The following objects are masked from faithful (pos = 7):

    eruptions, waiting

The following objects are masked from faithful (pos = 10):

    eruptions, waiting

> hist(eruptions,15,prob=T) # proportions, not frequencies
> lines(density(eruptions)) # lines makes a curve, default bandwidth
> lines(density(eruptions,bw="SJ"),col='red') # Use SJ bandwidth, in red
```



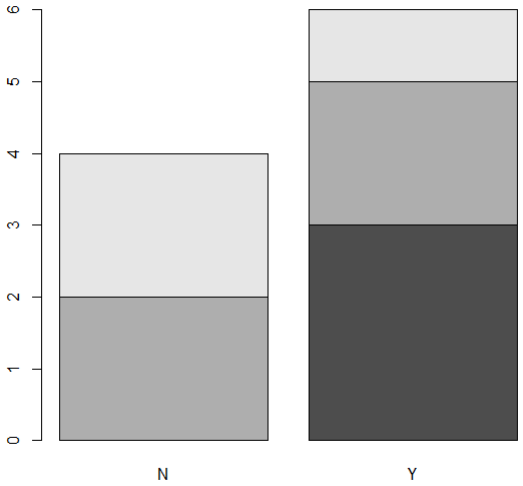
< Section 4: Bivariate Data >

```
> smokes = c("Y","N","N","Y","N","Y","Y","Y","N","Y")
> amount = c(1,2,2,3,3,1,2,1,3,2)
> table(smokes,amount)
       amount
smokes 1 2 3
     N 0 2 2
     Y 3 2 1
>
> tmp=table(smokes,amount) # store the table
> old.digits = options("digits") # store the number of digits
> options(digits=3) # only print 3 decimal places
> prop.table(tmp,1) # the rows sum to 1 now
       amount
smokes     1     2     3
     N 0.000 0.500 0.500
     Y 0.500 0.333 0.167
> prop.table(tmp,2) # the columns sum to 1 now
       amount
smokes     1     2     3
     N 0.000 0.500 0.667
     Y 1.000 0.500 0.333
> prop.table(tmp)# all the numbers sum to 1
       amount
smokes   1   2   3
     N 0.0 0.2 0.2
     Y 0.3 0.2 0.1
> options(digits=old.digits) # restore the number of digits
```

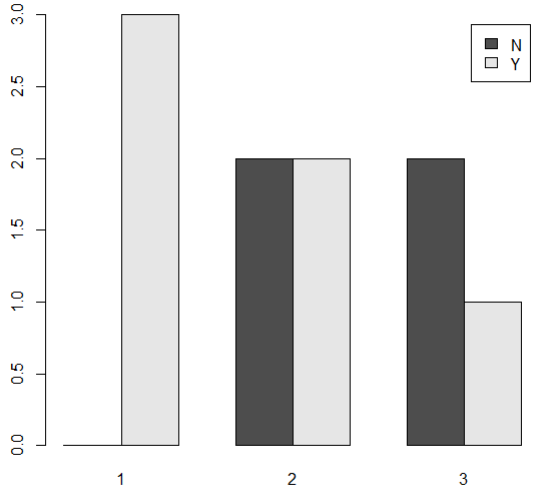
```
Error in options(digits = old.digits) :
  유효하지 않은 'digits' 파라미터입니다. 0...22만을 허용합니다
>
> barplot(table(smokes,amount))
```



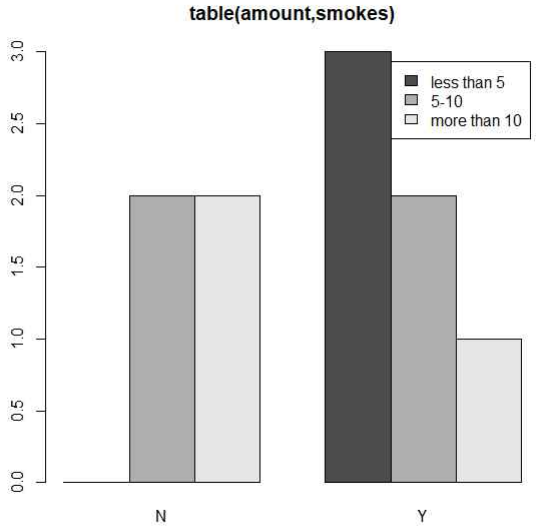

```
> barplot(table(amount,smokes))
```




```
> barplot(table(smokes,amount),
+     beside=TRUE, # put beside not stacked
+     legend.text=T) # add legend
```

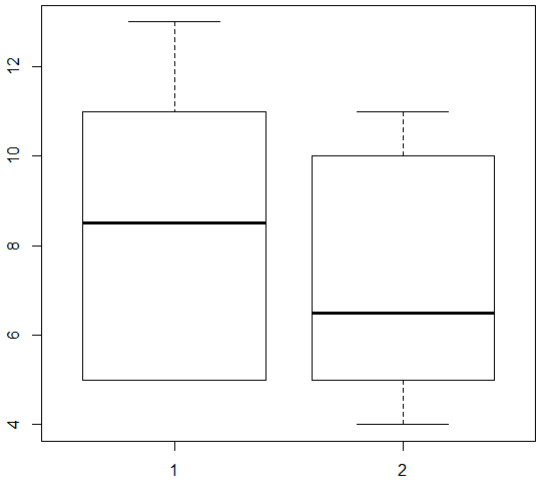



```
> barplot(table(amount,smokes),main="table(amount,smokes)",
+     beside=TRUE,
+     legend.text=c("less than 5","5-10","more than 10"))
```

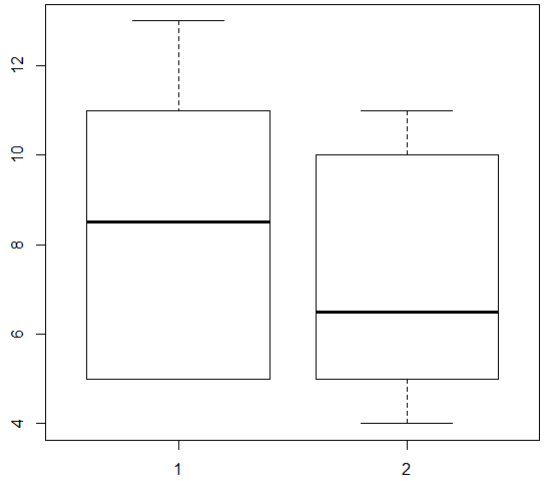



```
> prop = function(x) x/sum(x)
>
> apply(x,2,prop)
Error in apply(x, 2, prop) :
  dim(X)는 반드시 양의 값을 가지는 길이를 가져야 합니다
```

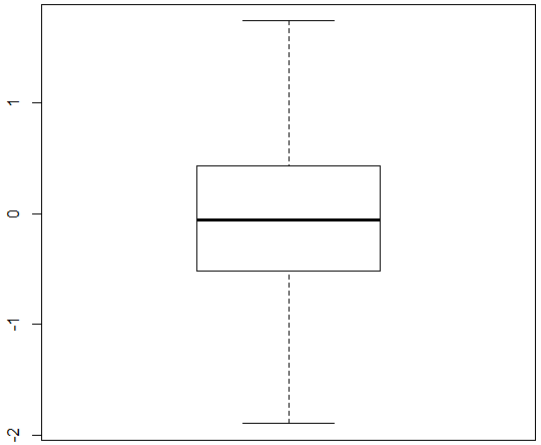
```
>
> t(apply(x,1,prop))
Error in apply(x, 1, prop) :
  dim(X)는 반드시 양의 값을 가지는 길이를 가져야 합니다
>
> x = c(5, 5, 5, 13, 7, 11, 11, 9, 8, 9)
> y = c(11, 8, 4, 5, 9, 5, 10, 5, 4, 10)
> boxplot(x,y)
```



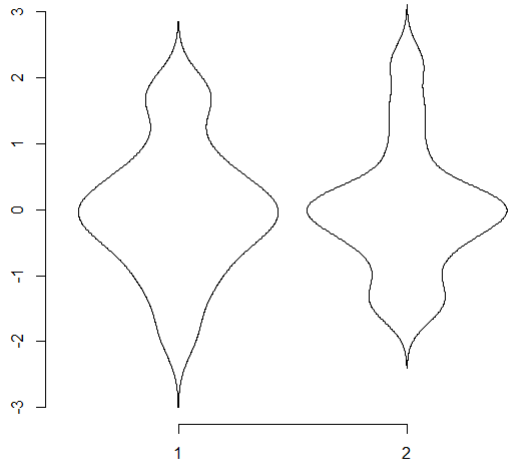

```
> amount = scan()
1: 5 5 5 13 7 11 11 9 8 9 11 8 4 5 9 5 10 5 4 10
21:
Read 20 items
> category = scan()
1: 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2
21:
Read 20 items
> boxplot(amount ~ category) # note the tilde
```



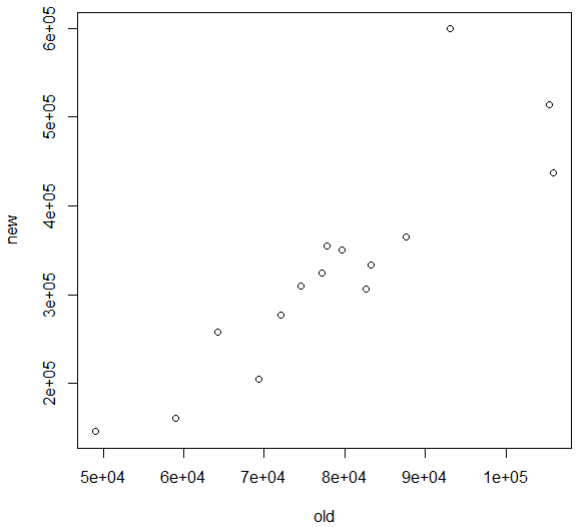

```
> library("UsingR"); data(home) # read in dataset home
> attach(home)
The following objects are masked from home (pos = 9):

     new, old

The following objects are masked from home (pos = 10):

     new, old

> names(home)
[1] "old" "new"
> boxplot(scale(old),scale(new)) # make boxplot after scaling each
경고메시지(들):
In if (use.cols) { :
  length > 1 이라는 조건이 있고, 첫번째 요소만이 사용될 것입니다
```



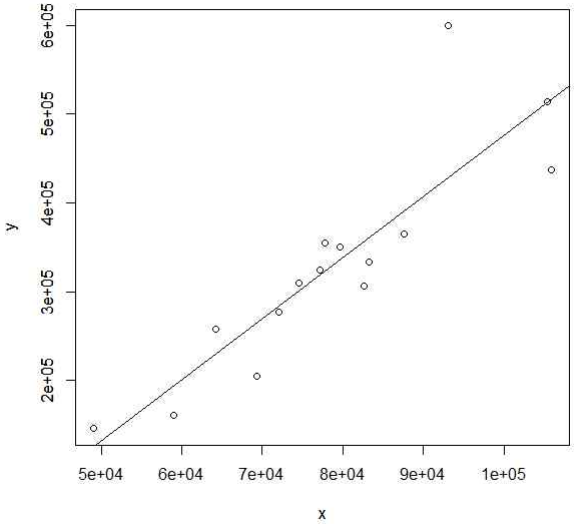

```
> detach(home)
>
> stripchart(scale(old),scale(new))
Error in stripchart.default(scale(old), scale(new)) :
  유효하지 않은 플롯팅 메소드입니다
>
> simple.violinplot(scale(old),scale(new))
경고메시지(들):
1: In if (add) { :
  length > 1 이라는 조건이 있고, 첫번째 요소만이 사용될 것입니다
2: In if (add) { :
  length > 1 이라는 조건이 있고, 첫번째 요소만이 사용될 것입니다
```



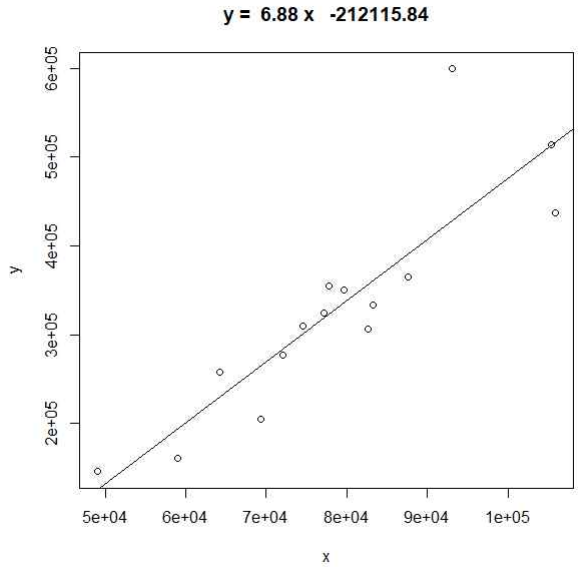

```
> data(home); attach(home)
The following objects are masked from home (pos = 9):

    new, old

The following objects are masked from home (pos = 10):

    new, old

> plot(old,new)
```



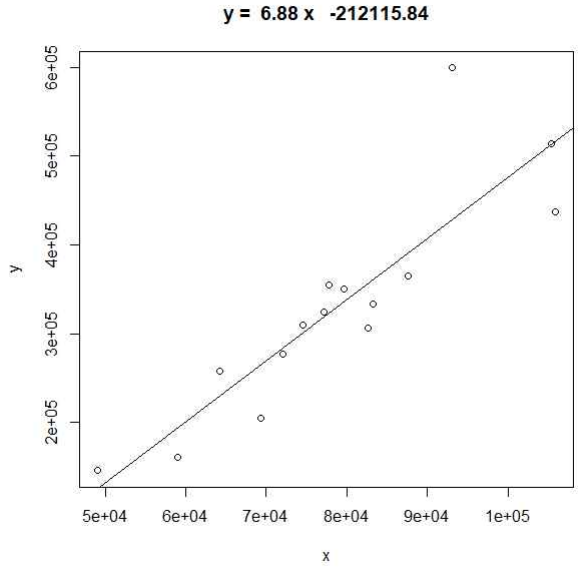

```
> detach(homedata)
>
> x = 1:2; y = c(2,4); df = data.frame(x=x,y=y)
> ls() # list all the varibles known
 [1] "amount"          "beer"            "beer.counts"     "Cars93"
 [5] "category"        "cats"            "cor.sp"          "data"
 [9] "df"              "emissions"       "f"               "faithful"
[13] "florida"         "gender"          "height"          "home"
[17] "homedata"        "InsectSprays"    "lm.res"          "lynx"
[21] "miles"           "movies"          "mpg"             "old.digits"
[25] "PlantGrowth"     "plot.regression" "price"           "prop"
[29] "sals"            "scores"          "smokes"          "study"
[33] "the.residuals"   "tmp"             "ToothGrowth"     "tread"
[37] "weight.ctrl"     "x"               "y"
> rm(y) # delete the y variable
> attach(df) # attach the data frame
The following object is masked _by_ .GlobalEnv:

    x

> ls() # y is visible, but doesn't show up
 [1] "amount"          "beer"            "beer.counts"     "Cars93"
 [5] "category"        "cats"            "cor.sp"          "data"
 [9] "df"              "emissions"       "f"               "faithful"
[13] "florida"         "gender"          "height"          "home"
[17] "homedata"        "InsectSprays"    "lm.res"          "lynx"
[21] "miles"           "movies"          "mpg"             "old.digits"
[25] "PlantGrowth"     "plot.regression" "price"           "prop"
[29] "sals"            "scores"          "smokes"          "study"
[33] "the.residuals"   "tmp"             "ToothGrowth"     "tread"
[37] "weight.ctrl"     "x"
> ls(pos=2) # y is in position 2 from being attached
[1] "x" "y"
> y # y is visible because df is attached
[1] 2 4
> x # which x did we find, x or df[['x']]
[1] 1 2
> x=c(1,3) # assign to x
> df # not the x in df
  x y
1 1 2
2 2 4
> detach(df)
> x # assigned to real x variable
[1] 1 3
> y
에러: 객체 'y'를 찾을 수 없습니다
>
> data(home); attach(home)
The following objects are masked from home (pos = 3):

    new, old

The following objects are masked from home (pos = 10):

    new, old

The following objects are masked from home (pos = 11):

    new, old

> x = old # use generic variable names
> y = new # for illustration only.
> plot(x,y)
> abline(lm(y ~ x))
```




```
> detach(home)
>
> data(home); attach(home)
The following objects are masked from home (pos = 3):

    new, old

The following objects are masked from home (pos = 10):

    new, old

The following objects are masked from home (pos = 11):

    new, old

> x = old; y = new
> simple.lm(x,y)

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
  -2.12e+05     6.88e+00
```




```
> detach(home)
>
> lm.res = simple.lm(x,y) # store the answers in lm.res
```

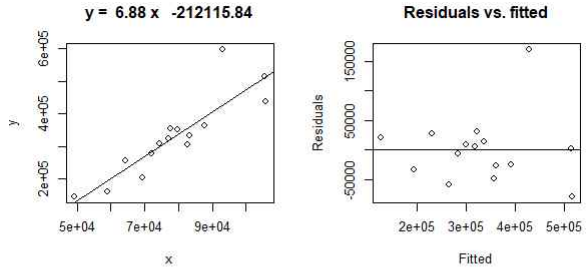



```
> coef(lm.res)
(Intercept)            x
  -2.12e+05     6.88e+00
> coef(lm.res)[1] # first one, use [2] for second
(Intercept)
```

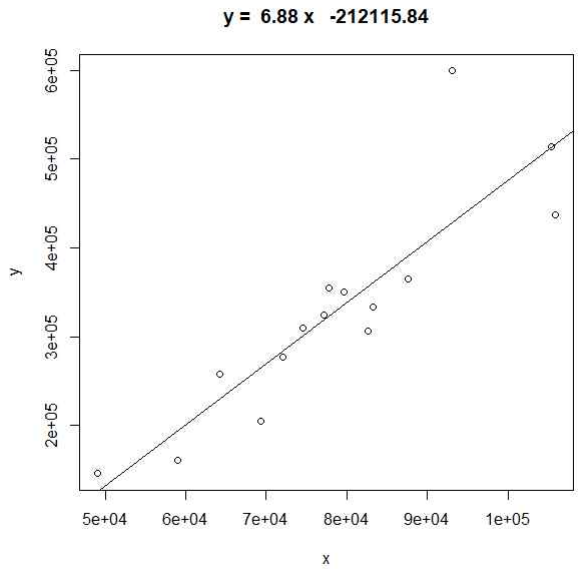
-212116
>
> simple.lm(x,y,show.residuals=TRUE)

Call:
lm(formula = y ~ x)

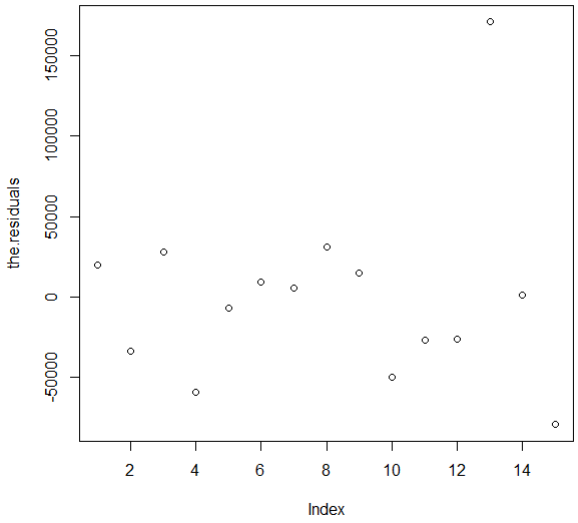
Coefficients:
(Intercept) x
  -2.12e+05 6.88e+00



> lm.res = simple.lm(x,y)

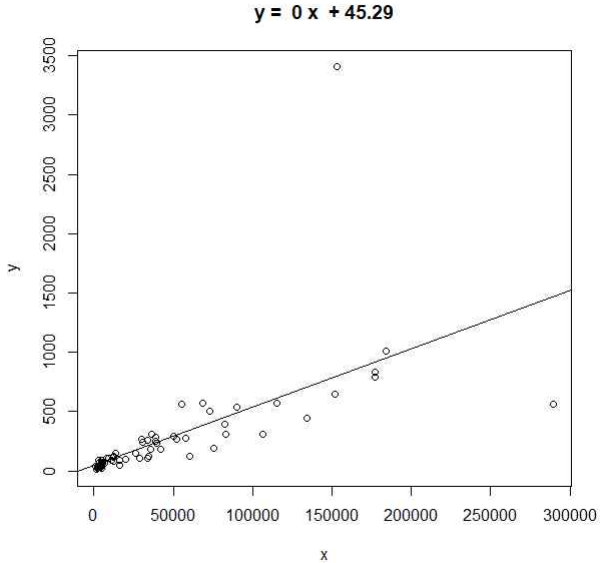


> the.residuals = resid(lm.res) # how to get residuals
> plot(the.residuals)

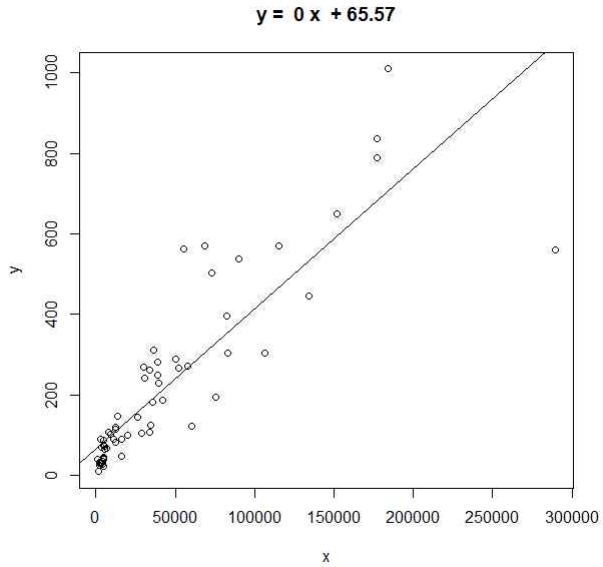


> cor(x,y) # to find R
[1] 0.881
> cor(x,y)^2 # to find R^2
[1] 0.776
>
> rank(c(2,3,5,7,11)) # already in order

```
[1] 1 2 3 4 5
> rank(c(5,3,2,7,11)) # for example, 5 is 3rd largest
[1] 3 2 1 4 5
> rank(c(5,5,2,7,5)) # ties have ranks averaged (2+3+4)/3=3
[1] 3 3 1 5 3
>
> cor(rank(x),rank(y))
[1] 0.925
>
> cor.sp <- function(x,y) cor(rank(x),rank(y))
>
> cor.sp(x,y)
[1] 0.925
>
> data("florida") # or read.table on florida.txt
> names(florida)
 [1] "County"      "GORE"        "BUSH"        "BUCHANAN"    "NADER"
 [6] "BROWN"       "HAGELIN"     "HARRIS"      "MCREYNOLDS"
"MOOREHEAD"
[11] "PHILLIPS"    "Total"
> attach(florida) # so we can get at the names BUSH, ...
> simple.lm(BUSH,BUCHANAN)

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
    45.28986      0.00492
```

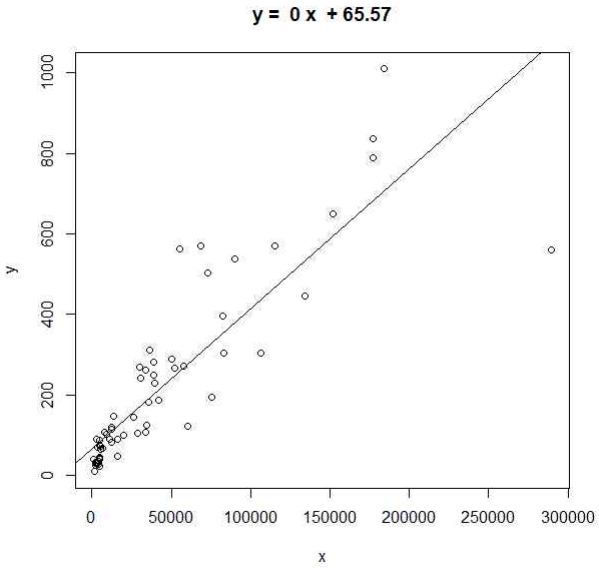



```
> detach(florida) # clean up
>
> identify(BUSH,BUCHANAN,n=2) # n=2 gives two points
Error in identify.default(BUSH, BUCHANAN, n = 2) :
  plot.new has not been called yet
> BUSH[50]
[1] 152846
> BUCHANAN[50]
[1] 3407
> florida[50,]
       County   GORE    BUSH BUCHANAN NADER BROWN HAGELIN
HARRIS MCREYNOLDS
50 PALM BEACH 268945 152846      3407   5564   743      143      45
 302
   MOOREHEAD PHILLIPS  Total
50       103      188 432286
>
> simple.lm(BUSH[-50],BUCHANAN[-50])

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
    65.57350      0.00348
```

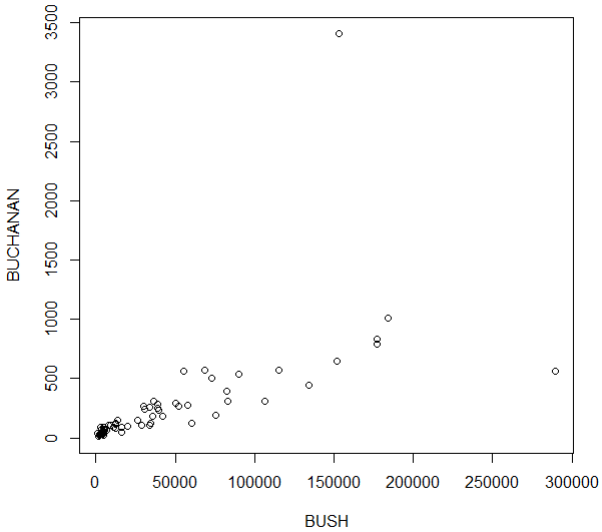



```
> 65.57350 + 0.00348 * BUSH[50]
[1] 597
>
> simple.lm(BUSH[-50],BUCHANAN[-50],pred=BUSH[50])
  1
598
```

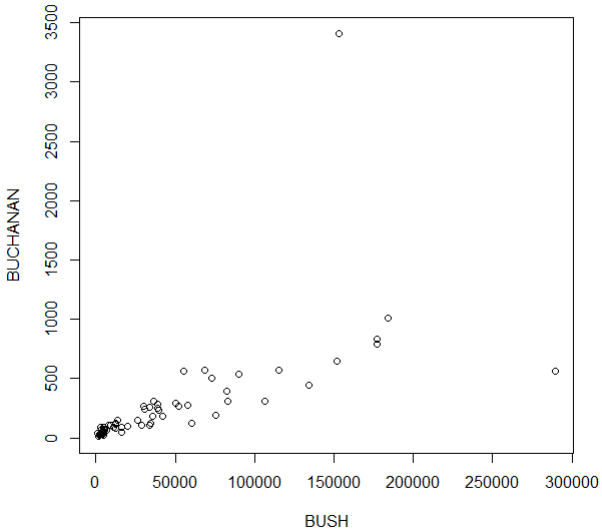
```
Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)                x
    65.57350         0.00348
> abline(65.57350,0.00348) # numbers from above
```



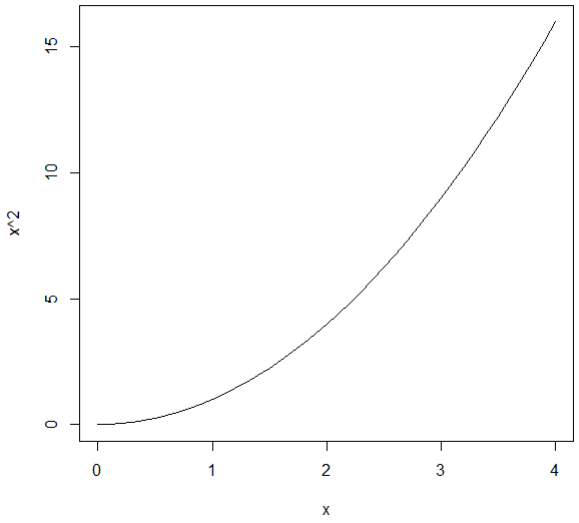

```
> library(MASS) # read in the external library
> attach(florida)
The following objects are masked from florida (pos = 3):

    BROWN, BUCHANAN, BUSH, County, GORE, HAGELIN, HARRIS,
    MCREYNOLDS, MOOREHEAD, NADER, PHILLIPS, Total

> plot(BUSH,BUCHANAN) # a scatter plot
> abline(lm(BUCHANAN ~ BUSH),lty="1") # lty sets line type
Error in int_abline(a = a, b = b, h = h, v = v, untf = untf, ...) :
  라인 타입이 유효하지 않습니다: 반드시 길이가 2, 4, 6 또는 8 이어야 합니다
> abline(rlm(BUCHANAN ~ BUSH),lty="2")
Error in int_abline(a = a, b = b, h = h, v = v, untf = untf, ...) :
  라인 타입이 유효하지 않습니다: 반드시 길이가 2, 4, 6 또는 8 이어야 합니다
> legend(locator(1),legend=c('lm','rlm'),lty=1:2)   # add legend
```



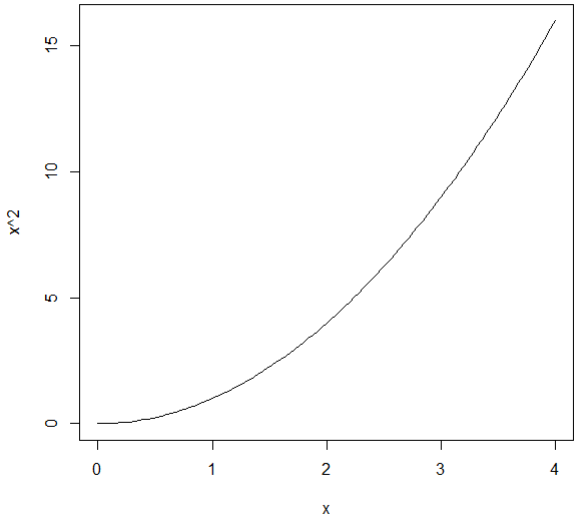

```
> detach(florida) # tidy up
>
> plot(BUSH,BUCHANAN)
> abline(rlm(BUCHANAN ~ BUSH),lty='1')
Error in int_abline(a = a, b = b, h = h, v = v, untf = untf, ...) :
  라인 타입이 유효하지 않습니다: 반드시 길이가 2, 4, 6 또는 8 이어야 합니다
> abline(rlm(BUCHANAN[-50] ~ BUSH[-50]),lty='2')
Error in int_abline(a = a, b = b, h = h, v = v, untf = untf, ...) :
  라인 타입이 유효하지 않습니다: 반드시 길이가 2, 4,
```



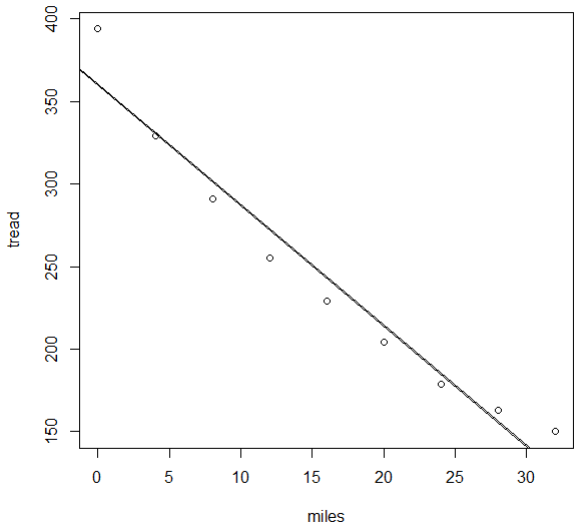

```
> x=seq(0,4,by=.1) # create the x values
> plot(x,x^2,type="l") # type="l" to make line
```




```
> curve(x^2,0,4)
```



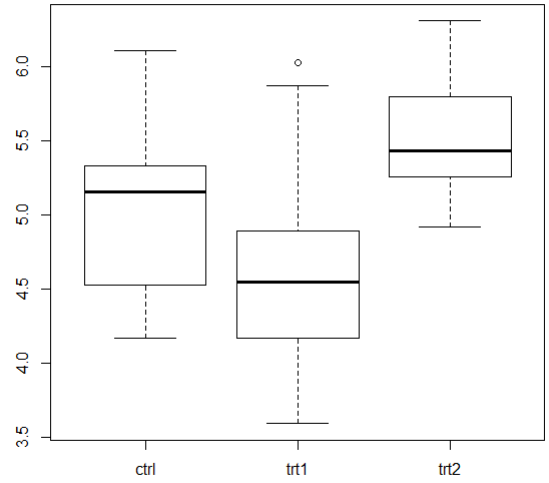

```
> miles = (0:8)*4 # 0 4 8 ... 32
> tread = scan()
1: 394 329 291 255 229 204 179 163 150
10:
Read 9 items
> plot(miles,tread) # make the scatterplot
> abline(lm(tread ~ miles))
> abline(360,-7.3)
> points(miles,360 - 7.3*miles,type="l")
> lines(miles,360 - 7.3*miles)
> curve(360 - 7.3*x,add=T) # add a function of x
```



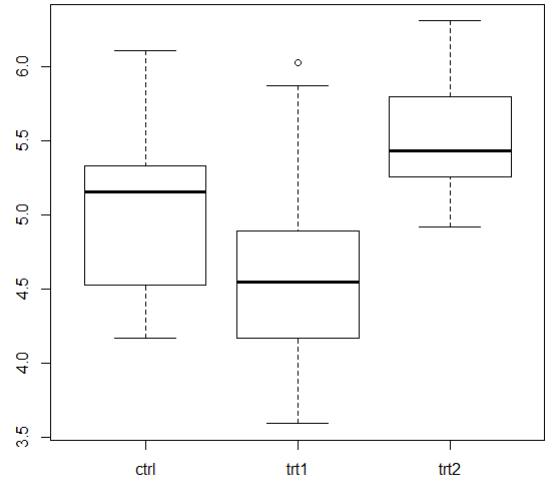
< Section 5: Multivariate Data >

```
> weight = c(150, 135, 210, 140)
> height = c(65, 61, 70, 65)
> gender = c("Fe","Fe","M","Fe")
> study = data.frame(weight,height,gender) # make the data frame
> study
  weight height gender
1    150     65     Fe
2    135     61     Fe
3    210     70      M
4    140     65     Fe
>
> study = data.frame(w=weight,h=height,g=gender)
>
> row.names(study)<-c("Mary","Alice","Bob","Judy")
>
> study
       w  h  g
Mary   150 65 Fe
Alice  135 61 Fe
Bob    210 70  M
Judy   140 65 Fe
> rm(weight) # clean out an old copy
> weight
 [1] 4.17 5.58 5.18 6.11 4.50 4.61 5.17 4.53 5.33 5.14 4.81 4.17 4.41 3.59
[15] 5.87 3.83 6.03 4.89 4.32 4.69 6.31 5.12 5.54 5.50 5.37 5.29 4.92 6.15
[29] 5.80 5.26
> attach(study)
The following objects are masked from study (pos = 9):

    g, h, w

> weight
 [1] 4.17 5.58 5.18 6.11 4.50 4.61 5.17 4.53 5.33 5.14 4.81 4.17 4.41 3.59
[15] 5.87 3.83 6.03 4.89 4.32 4.69 6.31 5.12 5.54 5.50 5.37 5.29 4.92 6.15
[29] 5.80 5.26
>
> study[,'weight'] # all rows, just the weight column
Error in `[.data.frame`(study, , "weight") : undefined columns selected
> study[,1] # all rows, just the first column
[1] 150 135 210 140
> study[,1:2]
       w  h
Mary   150 65
Alice  135 61
Bob    210 70
Judy   140 65
>
> study['Mary',]
       w  h  g
Mary 150 65 Fe
> study['Mary','weight']
NULL
>
> study$weight # using $
NULL
> study[['weight']] # using the name.
NULL
> study[['w']] # unambiguous shortcuts are okay
[1] 150 135 210 140
> study[[1]] # by position
[1] 150 135 210 140
>
> study[study$gender == 'Fe', ] # use $ to access gender via a list
[1] w h g
<0 행> <또는 row.names의 길이가 0입니다>
>
> data(PlantGrowth)
> PlantGrowth
   weight group
1    4.17  ctrl
2    5.58  ctrl
3    5.18  ctrl
4    6.11  ctrl
5    4.50  ctrl
6    4.61  ctrl
7    5.17  ctrl
8    4.53  ctrl
9    5.33  ctrl
10   5.14  ctrl
11   4.81  trt1
```

```
12    4.17   trt1
13    4.41   trt1
14    3.59   trt1
15    5.87   trt1
16    3.83   trt1
17    6.03   trt1
18    4.89   trt1
19    4.32   trt1
20    4.69   trt1
21    6.31   trt2
22    5.12   trt2
23    5.54   trt2
24    5.50   trt2
25    5.37   trt2
26    5.29   trt2
27    4.92   trt2
28    6.15   trt2
29    5.80   trt2
30    5.26   trt2
>
> attach(PlantGrowth)
The following objects are masked from PlantGrowth (pos = 9):

    group, weight

> weight.ctrl = weight[group == "ctrl"]
>
> unstack(PlantGrowth)
   ctrl trt1 trt2
1  4.17 4.81 6.31
2  5.58 4.17 5.12
3  5.18 4.41 5.54
4  6.11 3.59 5.50
5  4.50 5.87 5.37
6  4.61 3.83 5.29
7  5.17 6.03 4.92
8  4.53 4.89 6.15
9  5.33 4.32 5.80
10 5.14 4.69 5.26
>
> boxplot(unstack(PlantGrowth))
```

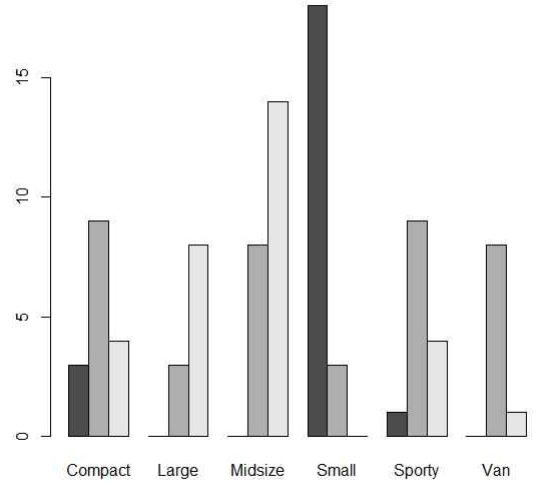



```
> boxplot(weight ~ group)
```

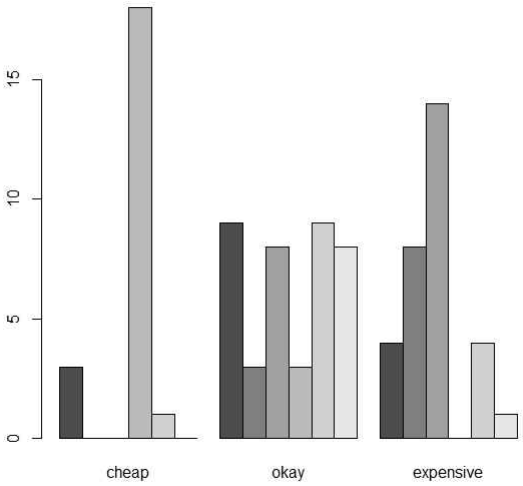



```
> library(MASS); data(Cars93); attach(Cars93)
The following objects are masked from Cars93 (pos = 3):

    AirBags, Cylinders, DriveTrain, EngineSize, Fuel.tank.capacity,
    Horsepower, Length, Luggage.room, Make, Man.trans.avail,
    Manufacturer, Max.Price, Min.Price, Model, MPG.city,
    MPG.highway, Origin, Passengers, Price, Rear.seat.room,
```

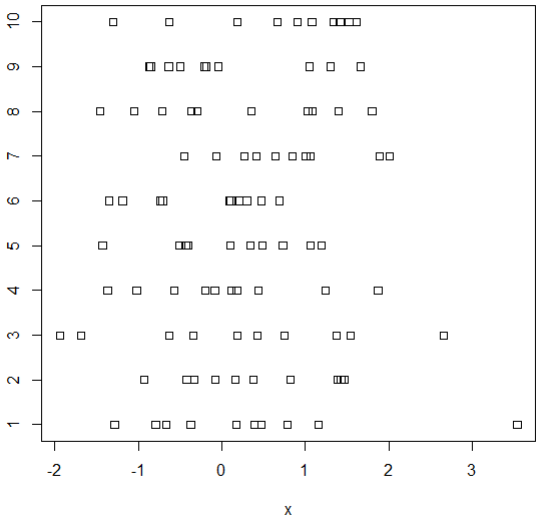
```
    Rev.per.mile, RPM, Turn.circle, Type, Weight, Wheelbase, Width

The following objects are masked from Cars93 (pos = 4):

    AirBags, Cylinders, DriveTrain, EngineSize, Fuel.tank.capacity,
    Horsepower, Length, Luggage.room, Make, Man.trans.avail,
    Manufacturer, Max.Price, Min.Price, Model, MPG.city,
    MPG.highway, Origin, Passengers, Price, Rear.seat.room,
    Rev.per.mile, RPM, Turn.circle, Type, Weight, Wheelbase, Width

The following objects are masked from Cars93 (pos = 10):

    AirBags, Cylinders, DriveTrain, EngineSize, Fuel.tank.capacity,
    Horsepower, Length, Luggage.room, Make, Man.trans.avail,
    Manufacturer, Max.Price, Min.Price, Model, MPG.city,
    MPG.highway, Origin, Passengers, Price, Rear.seat.room,
    Rev.per.mile, RPM, Turn.circle, Type, Weight, Wheelbase, Width

The following objects are masked from Cars93 (pos = 11):

    AirBags, Cylinders, DriveTrain, EngineSize, Fuel.tank.capacity,
    Horsepower, Length, Luggage.room, Make, Man.trans.avail,
    Manufacturer, Max.Price, Min.Price, Model, MPG.city,
    MPG.highway, Origin, Passengers, Price, Rear.seat.room,
    Rev.per.mile, RPM, Turn.circle, Type, Weight, Wheelbase, Width

> ## make some categorical variables using cut
> price = cut(Price,c(0,12,20,max(Price)))
> levels(price)=c("cheap","okay","expensive")
> mpg = cut(MPG.highway,c(0,20,30,max(MPG.highway)))
> levels(mpg) = c("gas guzzler","okay","miser")
> ## now look at the relationships
> table(Type)
Type
Compact   Large Midsize   Small  Sporty     Van
     16      11      22      21      14       9
> table(price,Type)
           Type
price       Compact Large Midsize Small Sporty Van
  cheap           3     0       0    18      1   0
  okay            9     3       8     3      9   8
  expensive       4     8      14     0      4   1
> table(price,Type,mpg)
, , mpg = gas guzzler

           Type
price       Compact Large Midsize Small Sporty Van
  cheap           0     0       0     0      0   0
  okay            0     0       0     0      0   2
  expensive       0     0       0     0      0   0

, , mpg = okay

           Type
price       Compact Large Midsize Small Sporty Van
  cheap           1     0       0     4      0   0
  okay            5     3       6     0      6   6
  expensive       4     8      14     0      4   1

, , mpg = miser

           Type
price       Compact Large Midsize Small Sporty Van
  cheap           2     0       0    14      1   0
  okay            4     0       2     3      3   0
  expensive       0     0       0     0      0   0

>
> barplot(table(price,Type),beside=T) # the price by different types
```



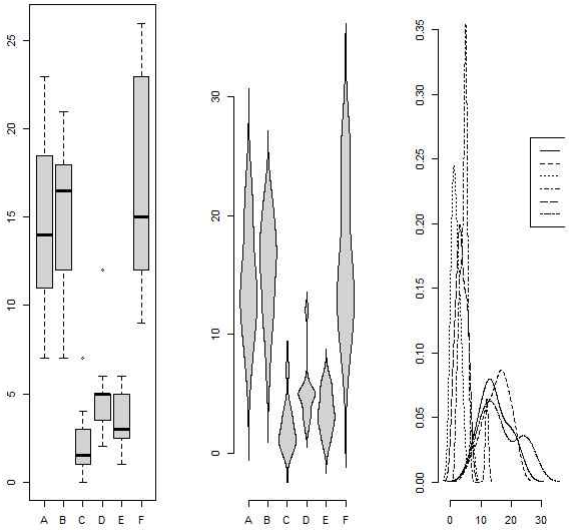

```
> barplot(table(Type,price),beside=T) # type by different prices
```



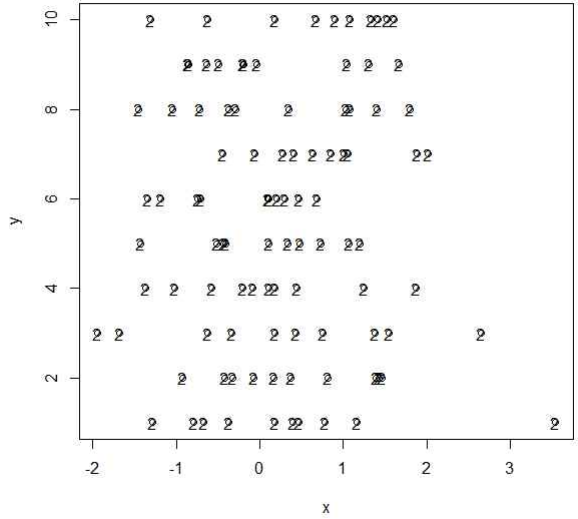

```
> x = rnorm(100)
> y = factor(rep(1:10,10))
> stripchart(x ~ y)
```




```
> par(mfrow=c(1,3)) # 3 graphs per page
> data(InsectSprays) # load in the data
> boxplot(count ~ spray, data = InsectSprays, col = "lightgray")
> simple.violinplot(count ~ spray, data = InsectSprays, col = "lightgray")
> simple.densityplot(count ~ spray, data = InsectSprays)
```

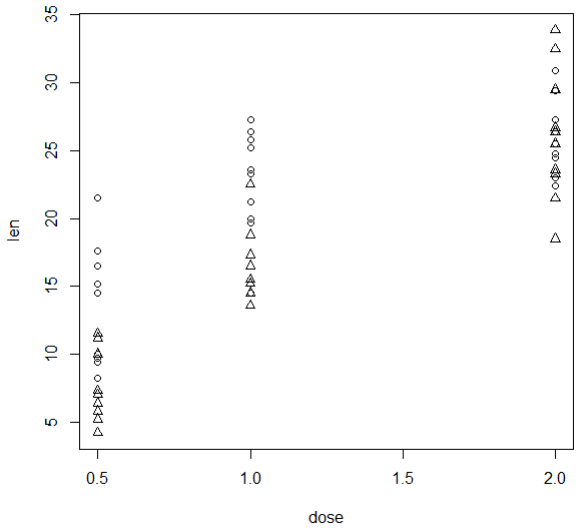


```
> plot(x,y) # simple scatterplot
> points(x,y,pch="2") # plot these with a triangle
```

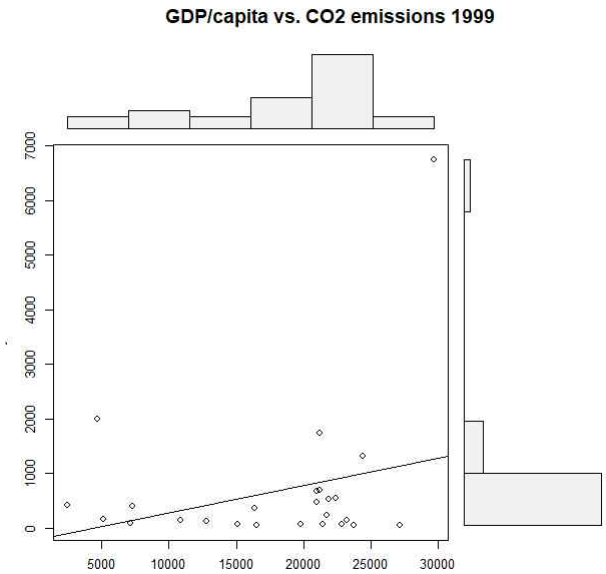



```
> data("ToothGrowth")
> attach(ToothGrowth)
The following objects are masked from ToothGrowth (pos = 3):

      dose, len, supp
```

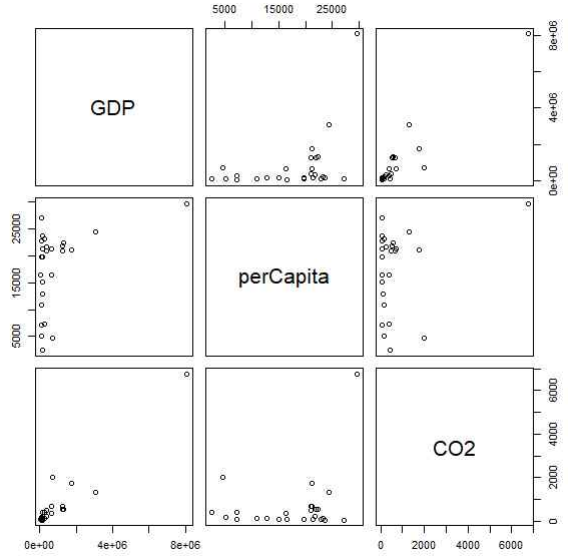
```
> plot(len ~ dose,pch=as.numeric(supp))
> ## click mouse to add legend.
> tmp = levels(supp) # store for a second
> legend(locator(1),legend=tmp,pch=1:length(tmp))
```



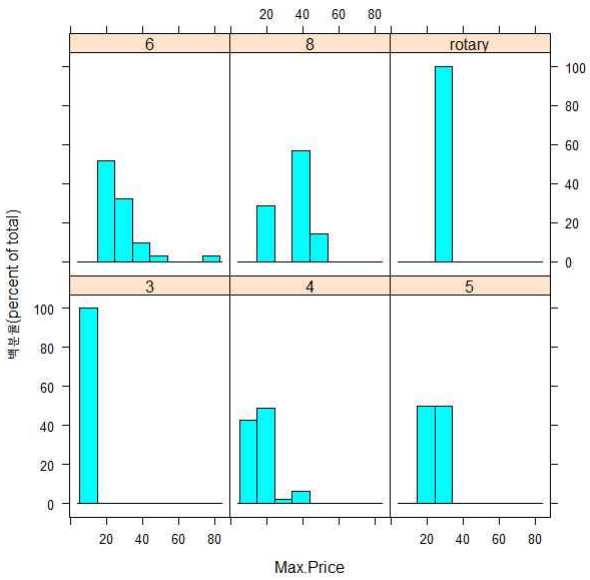

```
> detach(ToothGrowth)
>
> data(emissions) # or read in from dataset
> attach(emissions)
The following object is masked from package:datasets:

    CO2

> simple.scatterplot(perCapita,CO2)
> title("GDP/capita vs. CO2 emissions 1999")
```



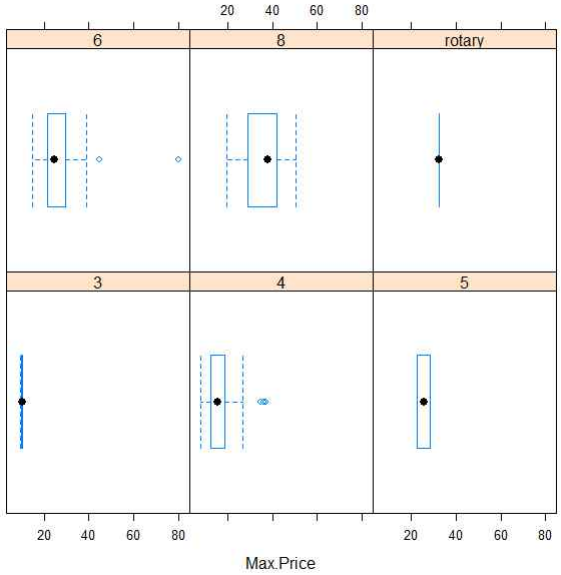

```
> detach(emissions)
>
> pairs(emissions)
```



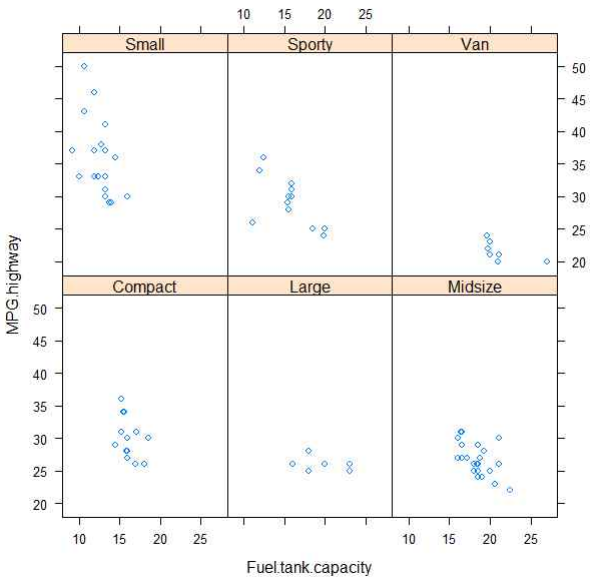

```
> histogram( ~ Max.Price | Cylinders , data = Cars93)
```



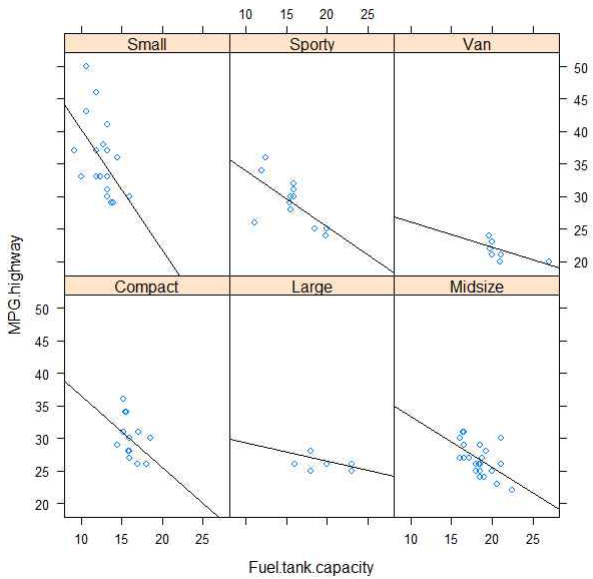

```
> bwplot( ~ Max.Price | Cylinders , data = Cars93)
```




```
> attach(Cars93) # don't need data = Cars93 now
The following objects are masked from Cars93 (pos = 3):

    AirBags, Cylinders, DriveTrain, EngineSize, Fuel.tank.capacity,
    Horsepower, Length, Luggage.room, Make, Man.trans.avail,
    Manufacturer, Max.Price, Min.Price, Model, MPG.city,
    MPG.highway, Origin, Passengers, Price, Rear.seat.room,
    Rev.per.mile, RPM, Turn.circle, Type, Weight, Wheelbase, Width

The following objects are masked from Cars93 (pos = 5):

    AirBags, Cylinders, DriveTrain, EngineSize, Fuel.tank.capacity,
    Horsepower, Length, Luggage.room, Make, Man.trans.avail,
    Manufacturer, Max.Price, Min.Price, Model, MPG.city,
    MPG.highway, Origin, Passengers, Price, Rear.seat.room,
    Rev.per.mile, RPM, Turn.circle, Type, Weight, Wheelbase, Width

The following objects are masked from Cars93 (pos = 6):

    AirBags, Cylinders, DriveTrain, EngineSize, Fuel.tank.capacity,
    Horsepower, Length, Luggage.room, Make, Man.trans.avail,
    Manufacturer, Max.Price, Min.Price, Model, MPG.city,
    MPG.highway, Origin, Passengers, Price, Rear.seat.room,
    Rev.per.mile, RPM, Turn.circle, Type, Weight, Wheelbase, Width

The following objects are masked from Cars93 (pos = 7):

    AirBags, Cylinders, DriveTrain, EngineSize, Fuel.tank.capacity,
    Horsepower, Length, Luggage.room, Make, Man.trans.avail,
    Manufacturer, Max.Price, Min.Price, Model, MPG.city,
    MPG.highway, Origin, Passengers, Price, Rear.seat.room,
    Rev.per.mile, RPM, Turn.circle, Type, Weight, Wheelbase, Width

The following objects are masked from Cars93 (pos = 13):

    AirBags, Cylinders, DriveTrain, EngineSize, Fuel.tank.capacity,
    Horsepower, Length, Luggage.room, Make, Man.trans.avail,
    Manufacturer, Max.Price, Min.Price, Model, MPG.city,
    MPG.highway, Origin, Passengers, Price, Rear.seat.room,
    Rev.per.mile, RPM, Turn.circle, Type, Weight, Wheelbase, Width

The following objects are masked from Cars93 (pos = 14):

    AirBags, Cylinders, DriveTrain, EngineSize, Fuel.tank.capacity,
    Horsepower, Length, Luggage.room, Make, Man.trans.avail,
    Manufacturer, Max.Price, Min.Price, Model, MPG.city,
    MPG.highway, Origin, Passengers, Price, Rear.seat.room,
    Rev.per.mile, RPM, Turn.circle, Type, Weight, Wheelbase, Width

> xyplot(MPG.highway ~ Fuel.tank.capacity | Type)
```

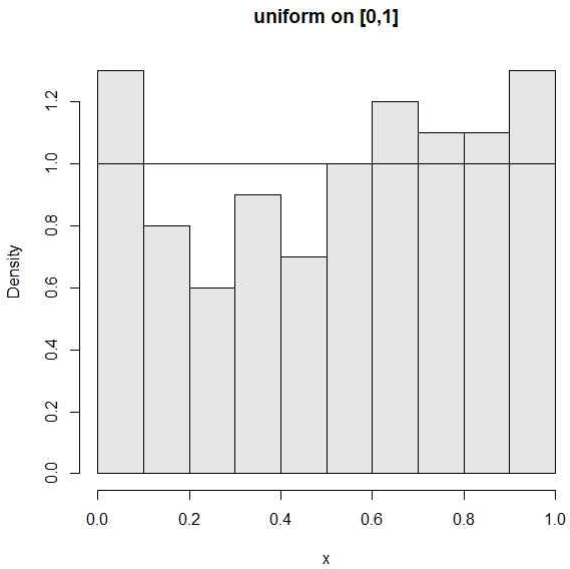



```
> ## plot with a regression line
> ## first define a regression line drawing function
> plot.regression = function(x,y) {
+     panel.xyplot(x,y)
+     panel.abline(lm(y~x))
+ }
> trellis.device(bg="white") # set background to white.
경고메시지(들):
In trellis.device(bg = "white") :
  'rellis.device'가 변경되었습니다.  'bg'는 아마도 생각하는 것과 같이 수행되지
않을 수 있습니다.
> xyplot(MPG.highway ~ Fuel.tank.capacity | Type, panel = plot.regression)
```

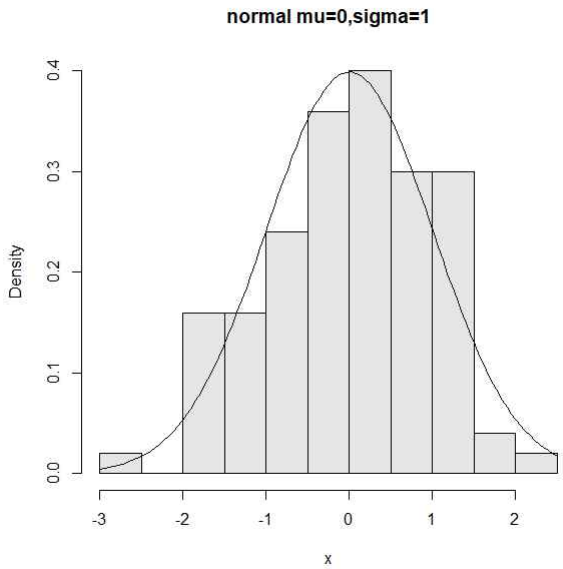


< Section 6: Random Data >

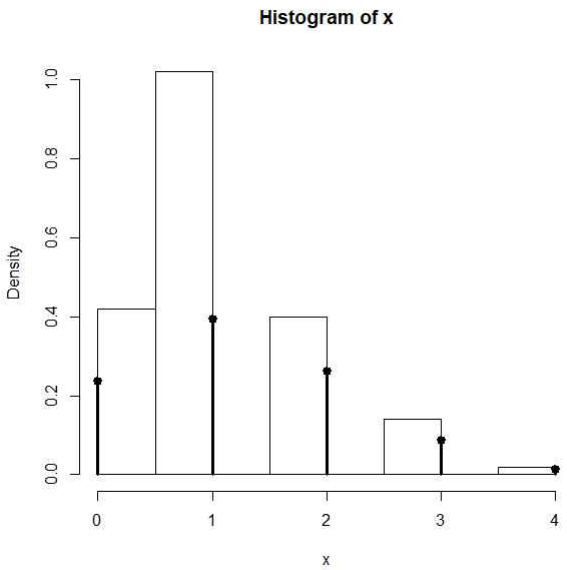
```
> sample(1:6,10,replace=T)
 [1] 5 1 4 5 1 3 4 2 2 2
>
> RollDie = function(n) sample(1:6,n,replace=T)
> RollDie(5)
[1] 3 4 1 5 5
>
> runif(1,0,2) # time at light
[1] 1.28
> runif(5,0,2) # time at 5 lights
[1] 0.0885 0.9188 1.7302 1.5819 0.5400
> runif(5) # 5 random numbers in [0,1]
[1] 0.30460 0.45515 0.59144 0.57971 0.00433
>
> x=runif(100) # get the random numbers
> hist(x,probability=TRUE,col=gray(.9),main="uniform on [0,1]")
> curve(dunif(x,0,1),add=T)
```



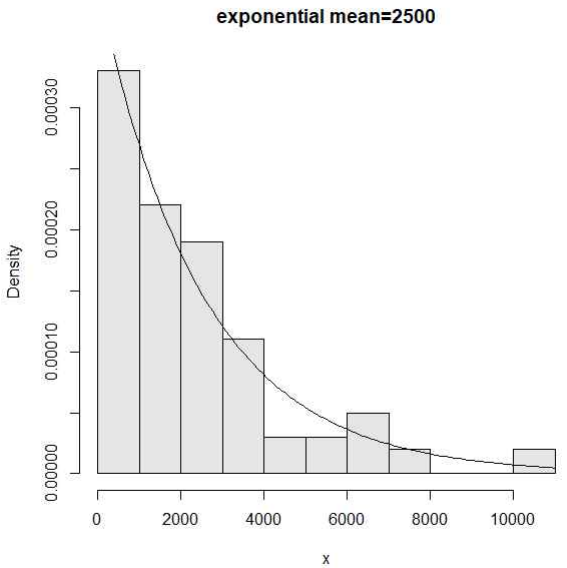

```
> rnorm(1,100,16) # an IQ score
[1] 81.2
> rnorm(1,mean=280,sd=10)# how long for a baby (10 days early)
[1] 273
>
> x=rnorm(100)
> hist(x,probability=TRUE,col=gray(.9),main="normal mu=0,sigma=1")
> curve(dnorm(x),add=T)
> ## also for IQs using rnorm(100,mean=100,sd=16)
```



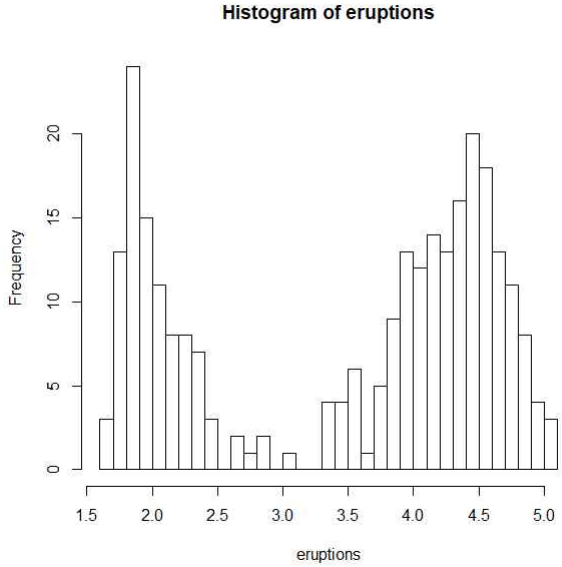

```
> n=1; p=.5 # set the probability
> rbinom(1,n,p) # different each time
[1] 0
> rbinom(10,n,p) # 10 different such numbers
 [1] 0 0 1 1 0 1 1 1 1 0
>
> n = 10; p=.5
> rbinom(1,n,p) # 6 successes in 10 trials
[1] 5
> rbinom(5,n,p) # 5 binomial number
[1] 5 3 7 4 9
>
> n=5; p=.25 # change as appropriate
> x=rbinom(100,n,p) # 100 random numbers
> hist(x,probability=TRUE,)
> ## use points, not curve as dbinom wants integers only for x
> xvals=0:n;points(xvals,dbinom(xvals,n,p),type="h",lwd=3)
> points(xvals,dbinom(xvals,n,p),type="p",lwd=3)
```



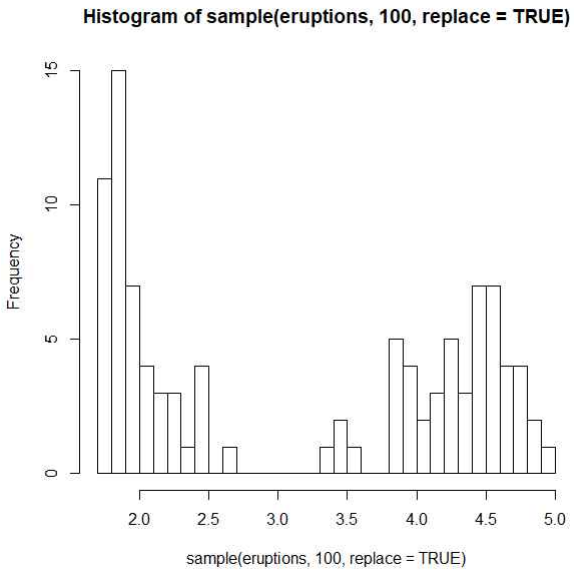

```
> x=rexp(100,1/2500)
> hist(x,probability=TRUE,col=gray(.9),main="exponential mean=2500")
> curve(dexp(x,1/2500),add=T)
```




```
> ## Roll a die
> sample(1:6,10,replace=TRUE)  # no sixes!
 [1] 4 3 1 4 4 2 1 2 3 1
> ## toss a coin
> sample(c("H","T"),10,replace=TRUE)
 [1] "H" "T" "H" "H" "T" "H" "T" "T" "H" "H"
> ## pick 6 of 54 (a lottery)
> sample(1:54,6) # no replacement
[1]  5 40 39 43 54  2
> ## pick a card. (Fancy! Uses paste, rep)
> cards = paste(rep(c("A",2:10,"J","Q","K"),4),c("H","D","S","C"))
> sample(cards,5) # a pair of jacks, no replacement
[1] "9 D" "A C" "9 S" "7 S" "A S"
> ## roll 2 die. Even fancier
> dice = as.vector(outer(1:6,1:6,paste))
> sample(dice,5,replace=TRUE) # replace when rolling dice
[1] "1 4" "5 4" "4 1" "4 3" "3 5"
>
> data(faithful) # part of R's base
> names(faithful) # find the names for faithful
[1] "eruptions" "waiting"
> eruptions = faithful[['eruptions']] # or attach and detach faithful
> sample(eruptions,10,replace=TRUE)
 [1] 4.35 4.25 2.00 1.70 4.72 1.67 4.35 4.00 4.45 5.03
> hist(eruptions,breaks=25) # the dataset
```



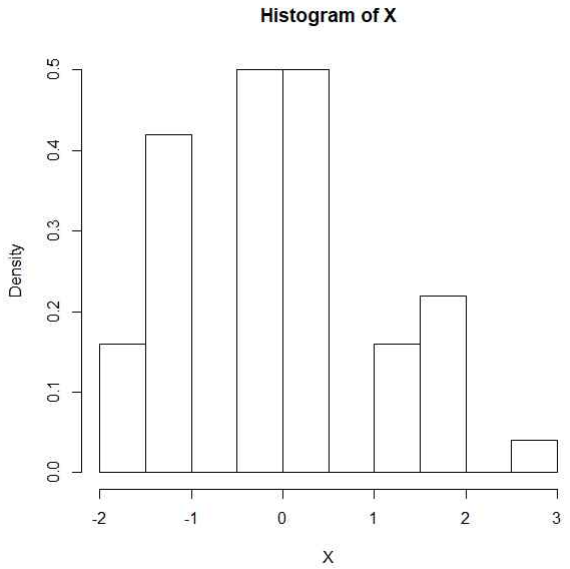

```
> ## the bootstrap sample
> hist(sample(eruptions,100,replace=TRUE),breaks=25)
```



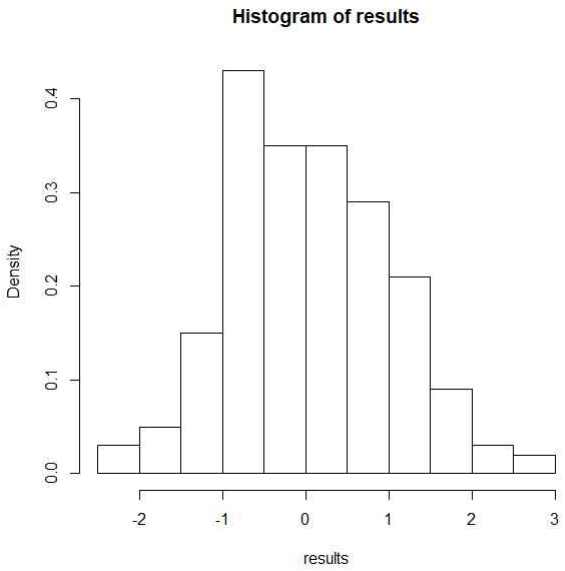

```
> pnorm(.7) # standard normal
[1] 0.758
> pnorm(.7,1,1) # normal mean 1, std 1
[1] 0.382
>
> pnorm(.7,lower.tail=F)
[1] 0.242
>
> qnorm(.75)
[1] 0.674
>
> x = rnorm(5,100,16)
> x
[1] 122.9 105.5 110.7  88.1 106.2
> z = (x-100)/16
> z
[1]  1.434  0.341  0.671 -0.746  0.389
>
> pnorm(z)
[1] 0.924 0.634 0.749 0.228 0.651
> pnorm(x,100,16) # enter in parameters
[1] 0.924 0.634 0.749 0.228 0.651
```

< Section 7: Simulations >

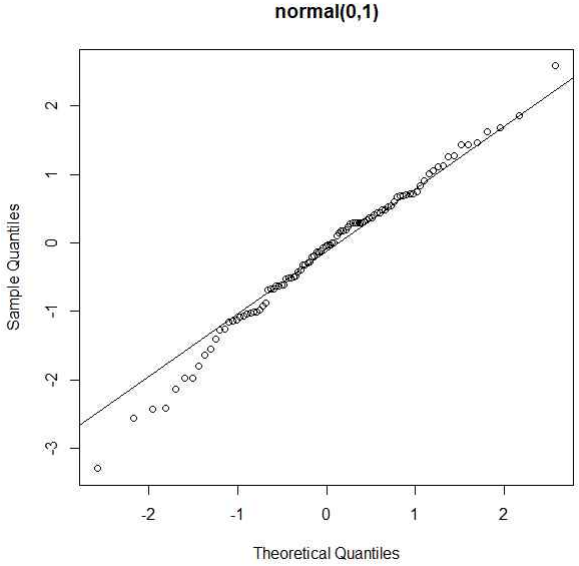
```
> n=10; p=.25; S= rbinom(1,n,p)
> (S - n*p)/sqrt(n*p*(1-p))
[1] 0.365
>
> n=10; p=.25; S=rbinom(100,n,p)
> X = (S - n*p)/sqrt(n*p*(1-p)) # has 100 random numbers
>
> hist(X,prob=T)
```



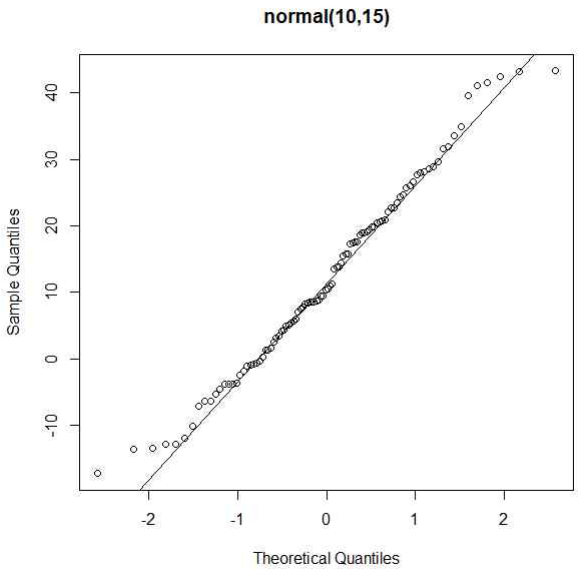

```
> results =numeric(0) # a place to store the results
> for (i in 1:100) { # the for loop
+   S = rbinom(1,n,p) # just 1 this time
+   results[i]=(S- n*p)/sqrt(n*p*(1-p)) # store the answer
+ }
>
> primes=c(2,3,5,7,11);## loop over indices of primes with this
> for(i in 1:5) print(primes[i])## or better, loop directly
[1] 2
[1] 3
[1] 5
[1] 7
[1] 11
> for(i in primes) print(i)
[1] 2
[1] 3
[1] 5
[1] 7
[1] 11
>
> results = c();
> mu = 0; sigma = 1
> for(i in 1:200) {
+   X = rnorm(100,mu,sigma) # generate random data
+   results[i] = (mean(X) - mu)/(sigma/sqrt(100))
+ }
> hist(results,prob=T)
```



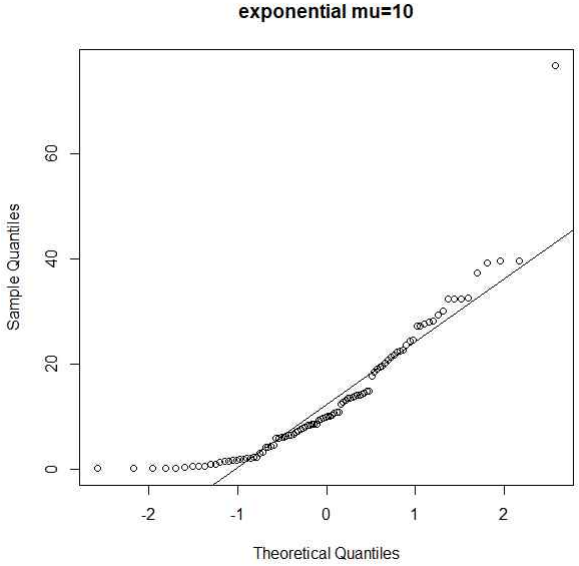
> x = rnorm(100,0,1);qqnorm(x,main='normal(0,1)'); qqline(x)



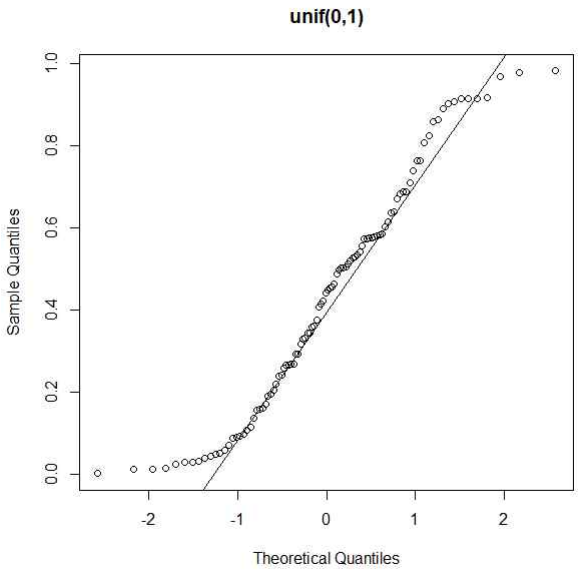
> x = rnorm(100,10,15);qqnorm(x,main='normal(10,15)'); qqline(x)



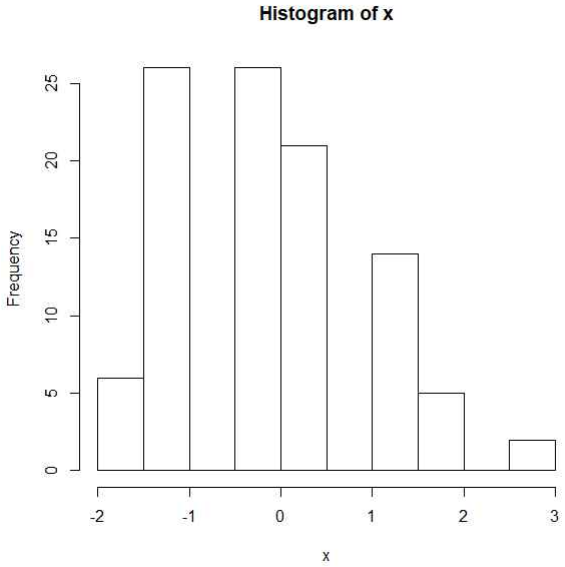
> x = rexp(100,1/10);qqnorm(x,main='exponential mu=10'); qqline(x)



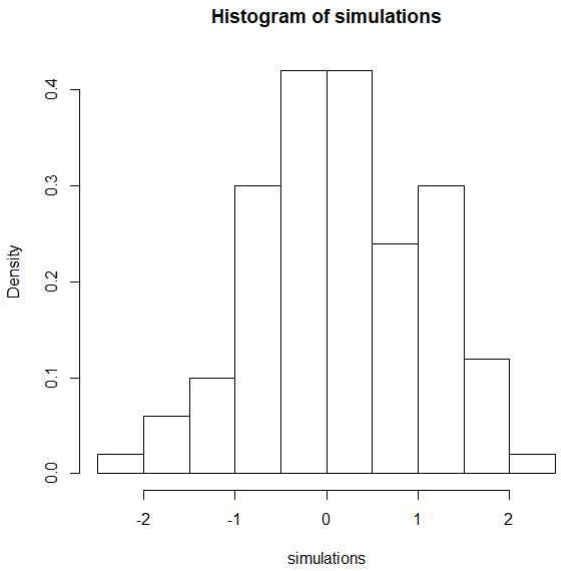
> x = runif(100,0,1);qqnorm(x,main='unif(0,1)'); qqline(x)



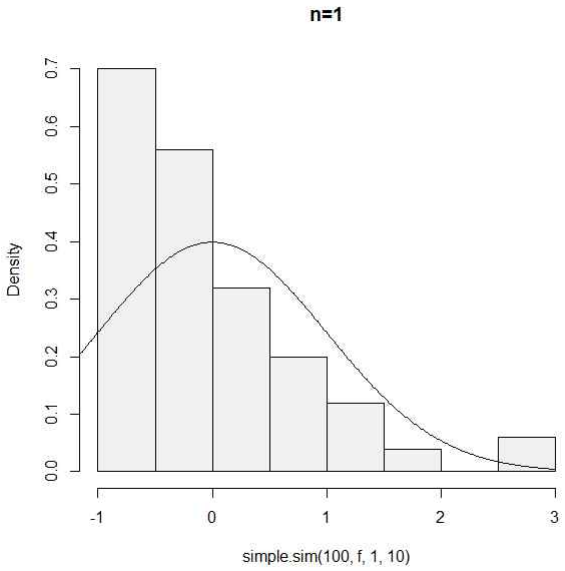

```
> f = function () {
+    S = rbinom(1,n,p)
+    (S- n*p)/sqrt(n*p*(1-p))
+ }
>
> x=simple.sim(100,f)
> hist(x)
```




```
> f = function(n=100,p=.5) {
+    S = rbinom(1,n,p)
+    (S- n*p)/sqrt(n*p*(1-p))
+ }
>
> simple.sim(1000,f,100,.5)
   [1] -1.0 -1.2 -0.2  0.4  0.2  0.2  0.6  0.0  0.8  0.6 -0.6  1.2 -0.6 -0.4
  [15]  3.8 -1.6 -1.4 -1.2  0.0  0.6  0.6  0.0  1.4  0.0  1.8 -0.6 -0.2  0.8
 (중략)
 [981] -0.4 -1.6 -0.2 -2.4 -1.2 -0.4 -0.6 -2.2 -1.6  0.2  0.8  0.2  0.4 -1.2
 [995] -0.4 -1.6  0.6  1.6 -1.2 -0.6
>
> the.range = function (x) max(x) - min(x)
>
> find.IQR = function(x) {
+    five.num = fivenum(x) # for Tukey's summary
+    five.num[4] - five.num[2]
+ }
>
> x = rnorm(100) # some sample data
> find.IQR # oops! no argument. Prints definition.
function(x) {
  five.num = fivenum(x) # for Tukey's summary
  five.num[4] - five.num[2]
}
> find.IQR(x) # this is better
[1] 1.37
>
> f = function(n=100,mu=0,sigma=1) {
+    nos = rnorm(n,mu,sigma)
+    (mean(nos)-mu)/(sigma/sqrt(n))
+ }
> simulations = simple.sim(100,f,100,5,5)
> hist(simulations,breaks=10,prob=TRUE)
```

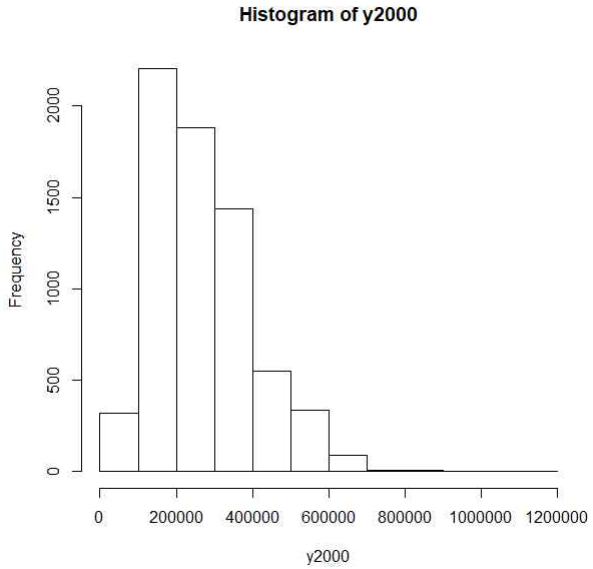



```
> f = function(n=100,mu=10) (mean(rexp(n,1/mu))-mu)/(mu/sqrt(n))
>
> xvals = seq(-3,3,.01) # for the density plot
> hist(simple.sim(100,f,1,10),probability=TRUE,main="n=1",col=gray(.95))
> points(xvals,dnorm(xvals,0,1),type="l") # plot normal curve
```

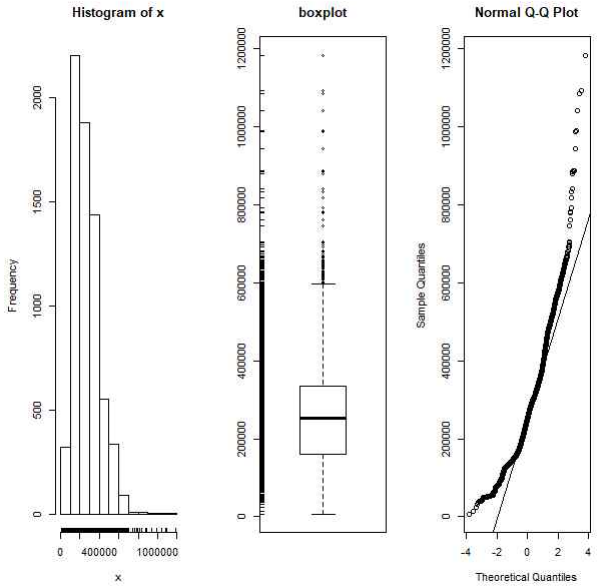


< Section 8: Exploratory Data Analysis >

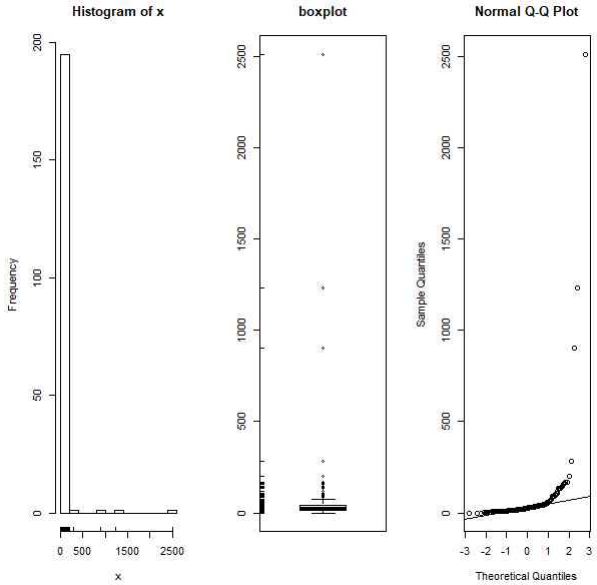
```
> data(homedata) # from simple package
> attach(homedata)
> hist(y1970); hist(y2000) # make two histograms
```



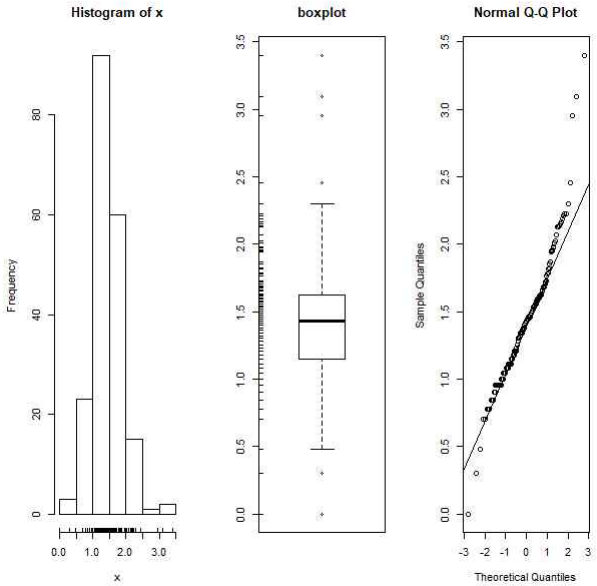

```
> detach(homedata) # clean up
>
> attach(homedata)
> simple.eda(y1970); simple.eda(y2000)
```



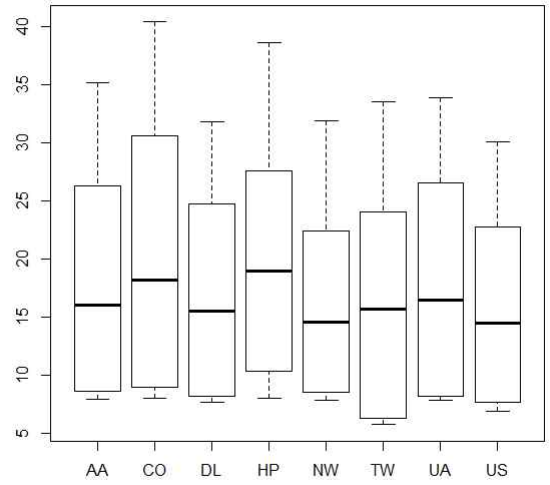

```
> detach(homedata) # clean up
>
> data(exec.pay) # or read in from file
> simple.eda(exec.pay)
```




```
> log.exec.pay = log(exec.pay[exec.pay >0])/log(10) # 0 is a problem
> simple.eda(log.exec.pay)
```

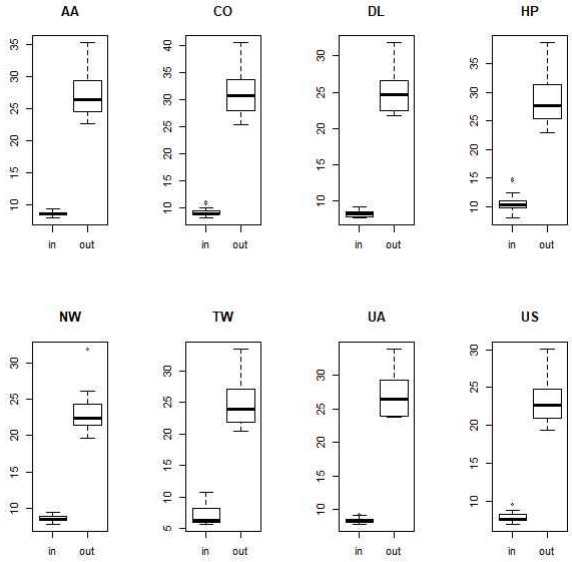



```
> data(ewr)
> names(ewr) # only 3-10 are raw data
 [1] "Year"    "Month"   "AA"      "CO"      "DL"      "HP"      "NW"
 [8] "TW"      "UA"      "US"      "inorout"
> airnames = names(ewr) # store them for later
> ewr.actual = ewr[,3:10] # get the important columns
> boxplot(ewr.actual)
```

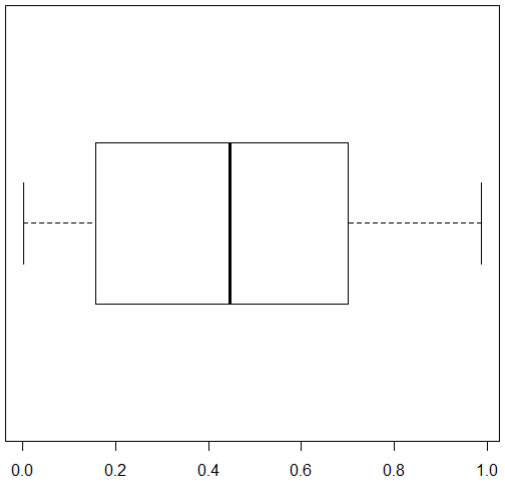



```
> par(mfrow=c(2,4)) # 2 rows 4 columns
```

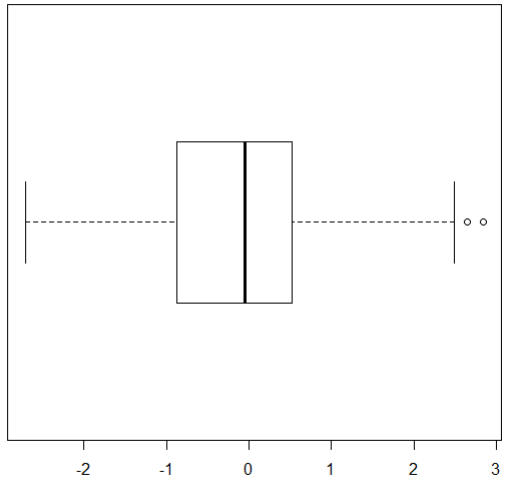
```
> attach(ewr)
> for(i in 3:10) boxplot(ewr[,i] ~ as.factor(inorout),main=airnames[i])
```



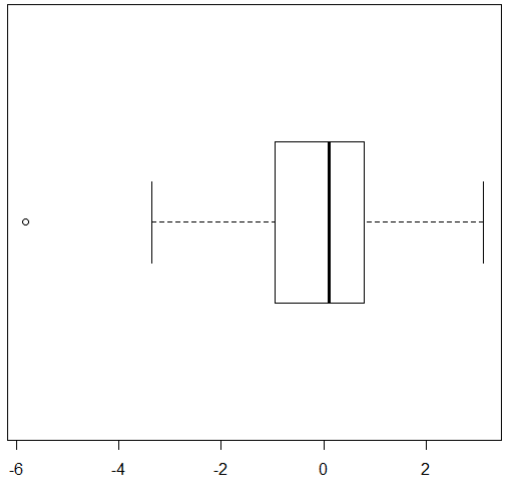

```
> detach(ewr)
> par(mfrow=c(1,1)) # return graphics as is (or close window)
>
> ## symmetric: short, regular then long
> X=runif(100);boxplot(X,horizontal=T,bty=n)
```



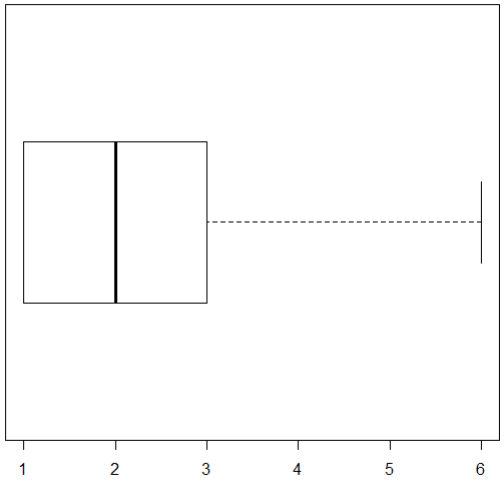

```
> X=rnorm(100);boxplot(X,horizontal=T,bty=n)
```



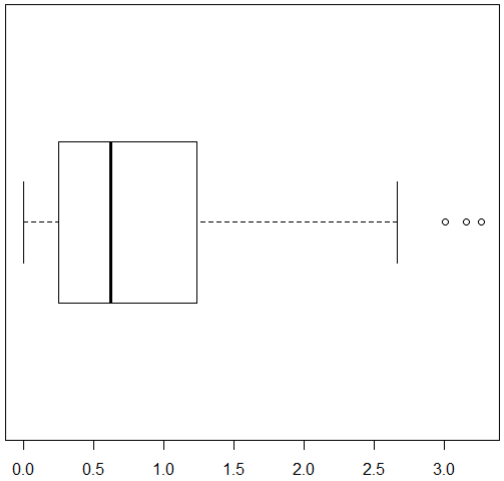

```
> X=rt(100,2);boxplot(X,horizontal=T,bty=n)
```



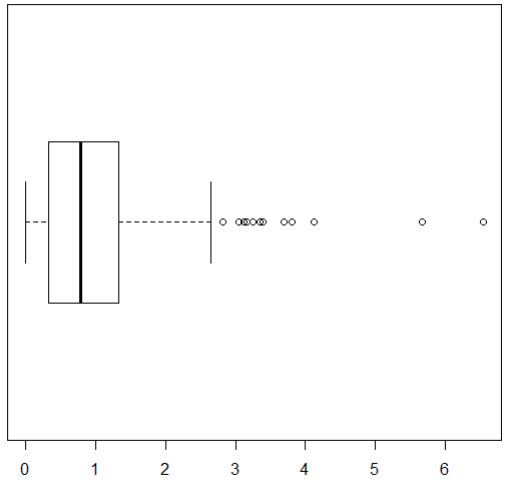

```
> ## skewed: short, regular then long
> # triangle distribution
> X=sample(1:6,100,p=7-(1:6),replace=T);boxplot(X,horizontal=T,bty=n)
```



> X=abs(rnorm(200));boxplot(X,horizontal=T,bty=n)

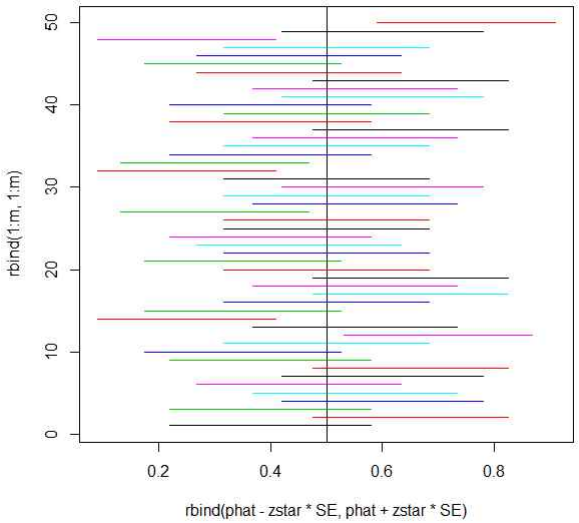


> X=rexp(200);boxplot(X,horizontal=T,bty=n)

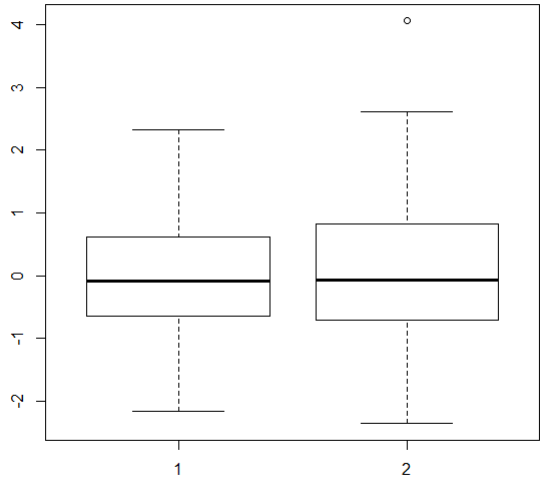


< Section 9: Condence Interval Estimation >

```
> alpha = c(0.2,0.1,0.05,0.001)
> zstar = qnorm(1 - alpha/2)
> zstar
[1] 1.28 1.64 1.96 3.29
>
> 2*(1-pnorm(zstar))
[1] 0.200 0.100 0.050 0.001
>
> m = 50; n=20; p = .5; # toss 20 coins 50 times
> phat = rbinom(m,n,p)/n # divide by n for proportions
> SE = sqrt(phat*(1-phat)/n) # compute SE
> alpha = 0.10:zstar = qnorm(1-alpha/2)
> matplot(rbind(phat - zstar*SE, phat + zstar*SE),
rbind(1:m,1:m),type="l",lty=1)
> abline(v=p) # draw line for p=0.5
```

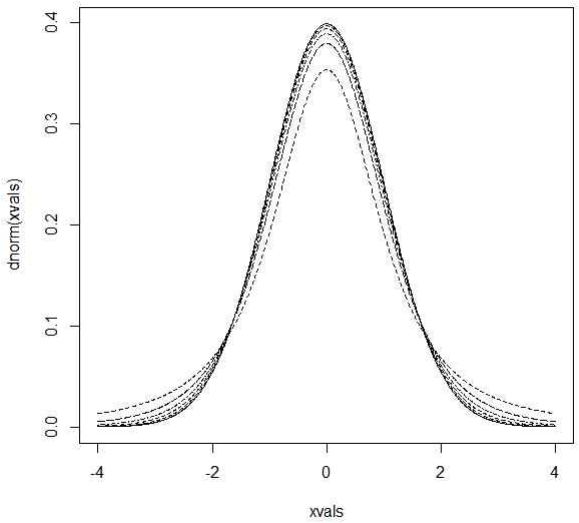



```
> prop.test(42,100)

        1-sample proportions test with continuity correction

data:  42 out of 100, null probability 0.5
X-squared = 2, df = 1, p-value = 0.1
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.323 0.523
sample estimates:
   p
0.42

>
> prop.test(42,100,conf.level=0.90)

        1-sample proportions test with continuity correction

data:  42 out of 100, null probability 0.5
X-squared = 2, df = 1, p-value = 0.1
alternative hypothesis: true p is not equal to 0.5
90 percent confidence interval:
 0.337 0.507
sample estimates:
   p
0.42

>
> ## define a function
> simple.z.test = function(x,sigma,conf.level=0.95) {
+    n = length(x);xbar=mean(x)
+    alpha = 1 - conf.level
+    zstar = qnorm(1-alpha/2)
+    SE = sigma/sqrt(n)
+    xbar + c(-zstar*SE,zstar*SE)
+ }
> ## now try it
> simple.z.test(x,1.5)
[1] -0.298  0.290
>
> t.test(x)

        One Sample t-test

data:  x
t = -0.05, df = 100, p-value = 1
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 -0.198  0.189
sample estimates:
mean of x
 -0.00449

>
> x=rnorm(100);y=rt(100,9)
> boxplot(x,y)
```




```
> qqnorm(x);qqline(x)
> qqnorm(y);qqline(y)
>
> xvals=seq(-4,4,.01)
```

```
> plot(xvals,dnorm(xvals),type="l")
> for(i in c(2,5,10,20,50)) points(xvals,dt(xvals,df=i),type="l",lty=i)
```



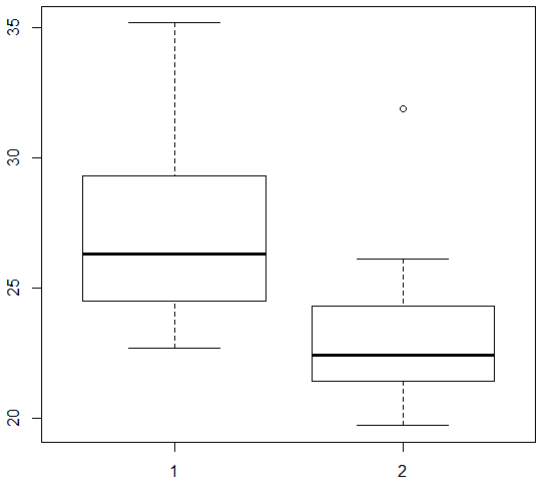

```
> x = c(110, 12, 2.5, 98, 1017, 540, 54, 4.3, 150, 432)
> wilcox.test(x,conf.int=TRUE)

        Wilcoxon signed rank test

data:  x
V = 60, p-value = 0.002
alternative hypothesis: true location is not equal to 0
95 percent confidence interval:
  33 514
sample estimates:
(pseudo)median
           150
```

< Section 10: Hypothesis Testing >

```
> prop.test(42,100,p=.5)

        1-sample proportions test with continuity correction

data:  42 out of 100, null probability 0.5
X-squared = 2, df = 1, p-value = 0.1
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.323 0.523
sample estimates:
   p
0.42
>
> prop.test(420,1000,p=.5)

        1-sample proportions test with continuity correction

data:  420 out of 1000, null probability 0.5
X-squared = 30, df = 1, p-value = 5e-07
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.389 0.451
sample estimates:
   p
0.42

>
> ## Compute the t statistic. Note we assume mu=25 under H_0
> xbar=22;s=1.5;n=10
> t = (xbar-25)/(s/sqrt(n))
> t
[1] -6.32
> ## use pt to get the distribution function of t
> pt(t,df=n-1)
[1] 6.85e-05
>
> x = c(12.8,3.5,2.9,9.4,8.7,.7,.2,2.8,1.9,2.8,3.1,15.8)
> stem(x)

  The decimal point is 1 digit(s) to the right of the |

  0 | 01233334
  0 | 99
  1 | 3
  1 | 6

>
> wilcox.test(x,mu=5,alt="greater")

        Wilcoxon signed rank test with continuity correction

data:  x
V = 40, p-value = 0.5
alternative hypothesis: true location is greater than 5

경고메시지(들):
In wilcox.test.default(x, mu = 5, alt = "greater") :
  tie가 있어 정확한 p값을 계산할 수 없습니다
>
> x = c(12.8,3.5,2.9,9.4,8.7,.7,.2,2.8,1.9,2.8,3.1,15.8)
> simple.median.test(x,median=5) # accept
[1] 0.388
> simple.median.test(x,median=10) # reject
[1] 0.0386
```

< Section 11: Two-sample tests >

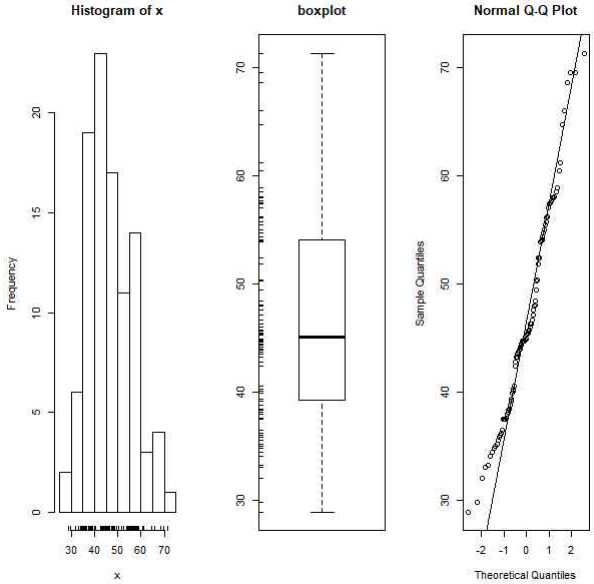
```
> prop.test(c(45,56),c(45+35,56+47))

        2-sample test for equality of proportions with continuity
        correction

data:  c(45, 56) out of c(45 + 35, 56 + 47)
X-squared = 0.01, df = 1, p-value = 0.9
alternative hypothesis: two.sided
95 percent confidence interval:
 -0.137  0.175
sample estimates:
prop 1 prop 2
 0.562  0.544

>
> x = c(15, 10, 13, 7, 9, 8, 21, 9, 14, 8)
> y = c(15, 14, 12, 8, 14, 7, 16, 10, 15, 12)
> t.test(x,y,alt="less",var.equal=TRUE)

        Two Sample t-test

data:  x and y
t = -0.5, df = 20, p-value = 0.3
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
 -Inf 2.03
sample estimates:
mean of x mean of y
     11.4      12.3

>
> t.test(x,y,alt="less")

        Welch Two Sample t-test

data:  x and y
t = -0.5, df = 20, p-value = 0.3
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
 -Inf 2.04
sample estimates:
mean of x mean of y
     11.4      12.3

>
> x = c(3, 0, 5, 2, 5, 5, 5, 4, 4, 5)
> y = c(2, 1, 4, 1, 4, 3, 3, 2, 3, 5)
> t.test(x,y,paired=TRUE)

        Paired t-test

data:  x and y
t = 3, df = 9, p-value = 0.008
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.326 1.674
sample estimates:
mean of the differences
                      1

>
> t.test(x,y)

        Welch Two Sample t-test

data:  x and y
t = 1, df = 20, p-value = 0.2
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.427  2.427
sample estimates:
mean of x mean of y
      3.8       2.8

>
> data(ewr) # read in data set
> attach(ewr) # unattach later
> tmp=subset(ewr, inorout == "out",select=c("AA","NW"))
> x=tmp[['AA']] # alternately AA[inorout=='out']
> y=tmp[['NW']]
> boxplot(x,y) # not shown
```




```
> wilcox.test(x,y)

        Wilcoxon rank sum test with continuity correction

data:  x and y
W = 500, p-value = 2e-05
alternative hypothesis: true location shift is not equal to 0

경고메시지(들):
In wilcox.test.default(x, y) : tie가 있어 정확한 p값을 계산할 수 없습니다
```

< Section 12: Chi Square Tests >

```
> x = rchisq(100,5); y=rchisq(100,50)
> simple.eda(x); simple.eda(y)
```




```
> freq = c(22,21,22,27,22,36)# specify probabilities, (uniform, like this, is
default though)
> probs = c(1,1,1,1,1,1)/6 # or use rep(1/6,6)
> chisq.test(freq,p=probs)

        Chi-squared test for given probabilities

data:  freq
X-squared = 7, df = 5, p-value = 0.2

>
> x = c(100,110,80,55,14)
> probs = c(29, 21, 17, 17, 16)/100
> chisq.test(x,p=probs)

        Chi-squared test for given probabilities

data:  x
X-squared = 60, df = 4, p-value = 3e-11

>
> yesbelt = c(12813,647,359,42)
> nobelt = c(65963,4000,2642,303)
> chisq.test(data.frame(yesbelt,nobelt))

        Pearson's Chi-squared test

data:  data.frame(yesbelt, nobelt)
X-squared = 60, df = 3, p-value = 9e-13
```

```
>
> die.fair = sample(1:6,200,p=c(1,1,1,1,1,1)/6,replace=T)
> die.bias = sample(1:6,100,p=c(.5,.5,1,1,1,2)/6,replace=T)
> res.fair = table(die.fair);res.bias = table(die.bias)
> rbind(res.fair,res.bias)
            1  2  3  4  5  6
res.fair 36 37 32 31 34 30
res.bias 10  7 22 16  9 36
>
> chisq.test(rbind(res.fair,res.bias))

        Pearson's Chi-squared test

data:  rbind(res.fair, res.bias)
X-squared = 30, df = 5, p-value = 7e-05

>
> chisq.test(rbind(res.fair,res.bias))[['exp']]
NULL
```