Statistical Analysis Using R Unit-1

MOHAMMAD WASIQ

9/4/2020

Introduction to R language

What is R ?

R is a language and environment for statistical computing . It has following features:

- It is free and can be downloaded from intenet: http://cran.r-project.org. It requires only 80 Mb spaces.

*It is a functional language.

- It has very powerful graphic capabilities.
- It has capabilities to interface with other important packages for data analysis and graphics.

Creating a vector in R—'c()'

Let us define a vector x as

```
x=c(5,4,3,1,4) # define x vector
x
```

```
## [1] 5 4 3 1 4
```

```
length(x)
```

```
## [1] 5
```

Note that R commands are written in a chunk which begins with three back ticks and ends with three back ticks. The command `{r vect1}` specifies that chunk is for R language and it is labeled by 'vect1'. All the R commands are to be written inside the chunk.

Arithmetic Operations on vectors

There are five basic operators in R, namely '+, -, *, / ^' , which are meant for addition , negation , multiplication , dividatiuon , exponentiation respectively. Besides these basic operators, a large number of function are also available for arithmetic computations. Let us create a new chumk:

```
sum(x) # sum of elements
```

```
## [1] 17
```

```
sum(x^2) # sum of squares
```

```
## [1] 67
```

```
sqrt(x) # square root of x
```

```
## [1] 2.236068 2.000000 1.732051 1.000000 2.000000
```

```r
log(x) # log of x with base e
```

```
## [1] 1.609438 1.386294 1.098612 0.000000 1.386294
```

```r
log(x, base=10)# log of x with 10
```

```
## [1] 0.6989700 0.6020600 0.4771213 0.0000000 0.6020600
```

```r
log(x, base=5) # log of x with 5
```

```
## [1] 1.0000000 0.8613531 0.6826062 0.0000000 0.8613531
```

```r
1/x # inverse of x
```

```
## [1] 0.2000000 0.2500000 0.3333333 1.0000000 0.2500000
```

```r
x^-1 # same as above
```

```
## [1] 0.2000000 0.2500000 0.3333333 1.0000000 0.2500000
```

```r
sin(x)
```

```
## [1] -0.9589243 -0.7568025  0.1411200  0.8414710 -0.7568025
```

```r
cos(x)
```

```
## [1]  0.2836622 -0.6536436 -0.9899925  0.5403023 -0.6536436
```

```r
tan(x)
```

```
## [1] -3.3805150  1.1578213 -0.1425465  1.5574077  1.1578213
```

```r
mean(x)
```

```
## [1] 3.4
```

```r
var(x) # variance of x
```

```
## [1] 2.3
```

```r
sd(x) # standard deviation of x
```

```
## [1] 1.516575
```

```r
cv=function(x) sd(x)/mean(x)*100
dump("cv", file="cv.txt")
source("cv.txt")
cv(x)
```

```
## [1] 44.60515
```

Concept of recycling rule

The basic rele is that smaller vector recycles upto the length of the larger vector. Let us see the commands in the following chunk;

```r
x=c(5,4,3,1,4)
x+1 # 1 will recycle up to the length of x andd add
```

```
## [1] 6 5 4 2 5
```

```r
2*x
```

```
## [1] 10  8  6  2  8
```

```
log(x+1)
```

```
## [1] 1.7917595 1.6094379 1.3862944 0.6931472 1.6094379
```

```
sin(2*x)
```

```
## [1] -0.5440211  0.9893582 -0.2794155  0.9092974  0.9893582
```

The function repeat—'rep()'

```
rep(c(1,2),c(3,3)) # repeat 1 3 times and 2 also 3 times
```

```
## [1] 1 1 1 2 2 2
```

```
rep(c(1,2),each=4)
```

```
## [1] 1 1 1 1 2 2 2 2
```

```
tasbihefatimi=rep(c("SubhanAllah","Alhamadulliah","Allahuakbar"),c(33,33,34))
tasbihefatimi
```

```
##   [1] "SubhanAllah"   "SubhanAllah"   "SubhanAllah"   "SubhanAllah"
##   [5] "SubhanAllah"   "SubhanAllah"   "SubhanAllah"   "SubhanAllah"
##   [9] "SubhanAllah"   "SubhanAllah"   "SubhanAllah"   "SubhanAllah"
##  [13] "SubhanAllah"   "SubhanAllah"   "SubhanAllah"   "SubhanAllah"
##  [17] "SubhanAllah"   "SubhanAllah"   "SubhanAllah"   "SubhanAllah"
##  [21] "SubhanAllah"   "SubhanAllah"   "SubhanAllah"   "SubhanAllah"
##  [25] "SubhanAllah"   "SubhanAllah"   "SubhanAllah"   "SubhanAllah"
##  [29] "SubhanAllah"   "SubhanAllah"   "SubhanAllah"   "SubhanAllah"
##  [33] "SubhanAllah"   "Alhamadulliah" "Alhamadulliah" "Alhamadulliah"
##  [37] "Alhamadulliah" "Alhamadulliah" "Alhamadulliah" "Alhamadulliah"
##  [41] "Alhamadulliah" "Alhamadulliah" "Alhamadulliah" "Alhamadulliah"
##  [45] "Alhamadulliah" "Alhamadulliah" "Alhamadulliah" "Alhamadulliah"
##  [49] "Alhamadulliah" "Alhamadulliah" "Alhamadulliah" "Alhamadulliah"
##  [53] "Alhamadulliah" "Alhamadulliah" "Alhamadulliah" "Alhamadulliah"
##  [57] "Alhamadulliah" "Alhamadulliah" "Alhamadulliah" "Alhamadulliah"
##  [61] "Alhamadulliah" "Alhamadulliah" "Alhamadulliah" "Alhamadulliah"
##  [65] "Alhamadulliah" "Alhamadulliah" "Allahuakbar"   "Allahuakbar"
##  [69] "Allahuakbar"   "Allahuakbar"   "Allahuakbar"   "Allahuakbar"
##  [73] "Allahuakbar"   "Allahuakbar"   "Allahuakbar"   "Allahuakbar"
##  [77] "Allahuakbar"   "Allahuakbar"   "Allahuakbar"   "Allahuakbar"
##  [81] "Allahuakbar"   "Allahuakbar"   "Allahuakbar"   "Allahuakbar"
##  [85] "Allahuakbar"   "Allahuakbar"   "Allahuakbar"   "Allahuakbar"
##  [89] "Allahuakbar"   "Allahuakbar"   "Allahuakbar"   "Allahuakbar"
##  [93] "Allahuakbar"   "Allahuakbar"   "Allahuakbar"   "Allahuakbar"
##  [97] "Allahuakbar"   "Allahuakbar"   "Allahuakbar"   "Allahuakbar"
```

```
rep(c("R1","R2","R3"),c(3,2,4))
```

```
## [1] "R1" "R1" "R1" "R2" "R2" "R3" "R3" "R3" "R3"
```

Logical operators—TRUE(1),FALSE(0)

>, <, <=,>=, == exactly equal to, != not equal to

```
x=c(2,5,3,1,8)
x>3
```

```
## [1] FALSE  TRUE FALSE FALSE  TRUE
```

```
sum(x>3)
```

```
## [1] 2
```

```
(x[x>2])
```

```
## [1] 5 3 8
```

```
mean(x[x>2])
```

```
## [1] 5.333333
```

```
mean(x[x<3])
```

```
## [1] 1.5
```

```
mean(x<3)
```

```
## [1] 0.4
```

```
#logical expression | or, & , and
x>2 & x<4
```

```
## [1] FALSE FALSE  TRUE FALSE FALSE
```

```
x[x>2 & x<4]
```

```
## [1] 3
```

Missing values— NA is.na()

```
x=c(2,5,3,NA,6)
is.na(x)
```

```
## [1] FALSE FALSE FALSE  TRUE FALSE
```

```
mean(x)
```

```
## [1] NA
```

```
mean(x,na.rm=TRUE)
```

```
## [1] 4
```

```
!is.na(x)
```

```
## [1]  TRUE  TRUE  TRUE FALSE  TRUE
```

```
mean(x[!is.na(x)])
```

```
## [1] 4
```

```
## Not a number, NaN, 0/0, Inf-Inf, Inf/Inf
x=c(0,5,3)
x/x
```

```
## [1] NaN   1   1
```

```
is.na(x/x)
```

```
## [1]  TRUE FALSE FALSE
```

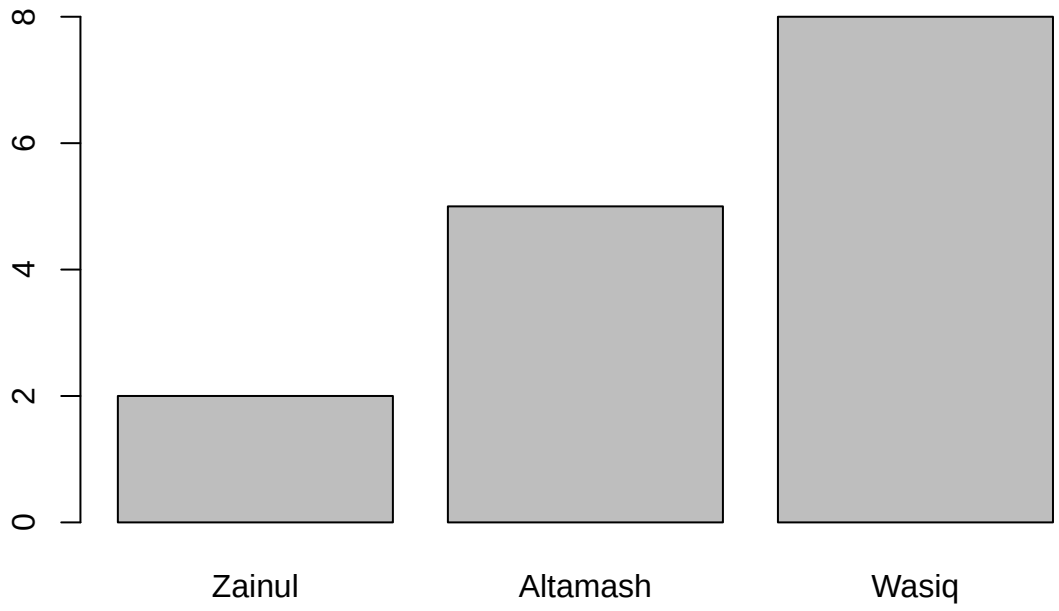
```
x=c(2,5,8)
x
```

```
## [1] 2 5 8
```

```
names(x)=c("Zainul","Altamash","Wasiq")
x
```

```
##   Zainul Altamash    Wasiq
##        2        5        8
```

```
barplot(x)
```



paste() The function 'paste()' behave like 'c()' but it always returns a character vector.

letters and **LETTERS**

```
letters
```

```
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s"
## [20] "t" "u" "v" "w" "x" "y" "z"
```

```
LETTERS
```

```
##  [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q" "R" "S"
## [20] "T" "U" "V" "W" "X" "Y" "Z"
```

```
letters[c(3,6)]
```

```
## [1] "c" "f"
```

```
LETTERS[1:6]
```

```
## [1] "A" "B" "C" "D" "E" "F"
```

```
month.abb
```

```
##  [1] "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep" "Oct" "Nov" "Dec"
```

```
month.abb[c(3,9)]
```

```
## [1] "Mar" "Sep"
```

```
month.abb[3:9]
```

```
## [1] "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep"
```

```
month.name
```

```
##  [1] "January"   "February"  "March"     "April"     "May"       "June"
##  [7] "July"      "August"    "September" "October"   "November"  "December"
```

substrin

```
substr(month.name,1,4)
```

```
##  [1] "Janu" "Febr" "Marc" "Apri" "May"  "June" "July" "Augu" "Sept" "Octo"
## [11] "Nove" "Dece"
```

Factor vector—'factor()', 'ordered()'

Factor is like the

```
grp=c("control","treatment","control","treatment")
grp
```

```
## [1] "control"   "treatment" "control"   "treatment"
```

```
grp=factor(grp)
as.integer(grp)
```

```
## [1] 1 2 1 2
```

```
# factor are efficient ways of storing data
as.integer(grp)
```

```
## [1] 1 2 1 2
```

```
levels(grp)
```

```
## [1] "control"   "treatment"
```

Matrices and arrays— 'matrix()' , 'arrays()'

The function 'matrix()' is used to create a matrix.

```
m=matrix(1:6,nrow=2,ncol=3)
m
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

```r
dim(m)
```

```
## [1] 2 3
```

```r
dimnames(m)=list(LETTERS[1:2],letters[1:3])
m
```

```
##   a b c
## A 1 3 5
## B 2 4 6
```

```r
colnames(m)=c("Samad","Radha","Umar")
m
```

```
##   Samad Radha Umar
## A     1     3    5
## B     2     4    6
```

```r
rownames(m)=c("Moradabad","Aligarh")
m
```

```
##           Samad Radha Umar
## Moradabad     1     3    5
## Aligarh       2     4    6
```

```r
matrix(1:6,ncol=3,byrow=TRUE)
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
```

```r
# Extraction of a matrix---[]
m[1,3]
```

```
## [1] 5
```

```r
m[,c(1,2)]
```

```
##           Samad Radha
## Moradabad     1     3
## Aligarh       2     4
```

```r
m[c(1,2),3]
```

```
## Moradabad   Aligarh
##         5         6
```

```r
## arrays()
array(1:8,c(2,2,2))
```

```
## , , 1
##
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
##
## , , 2
##
##      [,1] [,2]
## [1,]    5    7
```

```
## [2,]    6    8
```

```
# naming arrays
dim(array)
```

```
## NULL
```

Data frame—'data.farme()'

These are objects, a kind of generalization of matrices. Generalization in the sense that columns can be of different types from each other. This flexibility is not with matrices. Use 'data.frame()' to construct a data from vector:

```
colors=c("red","yellow","blue")
numbers=c(1,2,3)
colors_numbers=data.frame(colors, numbers)
colors_numbers
```

```
##   colors numbers
## 1    red       1
## 2 yellow       2
## 3   blue       3
```

```
colors=c("red","yellow","blue")
numbers=c(1,2,3)
more_numbers=c(5,4,7)
colors_numbers=data.frame(colors,numbers,more_numbers)
colors_numbers
```

```
##   colors numbers more_numbers
## 1    red       1            5
## 2 yellow       2            4
## 3   blue       3            7
```

```
colors=c("red","yellow","blue")
numbers=c(1,2,3)
more_numbers=c(5,4,7)
colors_numbers=data.frame(Colors=colors,Numbers=numbers,MoreNUMBERS=more_numbers)
colors_numbers
```

```
##   Colors Numbers MoreNUMBERS
## 1    red       1           5
## 2 yellow       2           4
## 3   blue       3           7
```

```
# change row names
row.names(colors_numbers)=paste0("Row",1:3)
colors_numbers
```

```
##      Colors Numbers MoreNUMBERS
## Row1    red       1           5
## Row2 yellow       2           4
## Row3   blue       3           7
```

```
# extraction from data.frame
colors_numbers[c(1,3),]
```

```
##      Colors Numbers MoreNUMBERS
## Row1    red       1           5
```

```
## Row3   blue       3           7
```

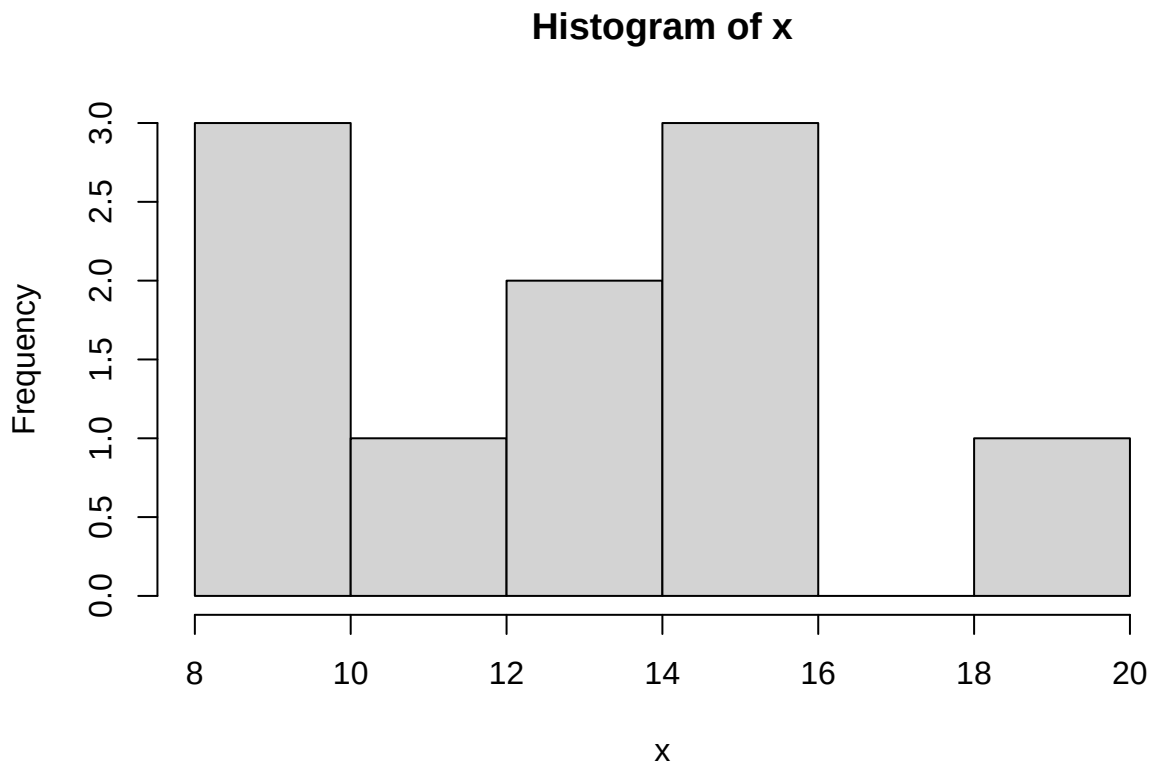
```
DF2=colors_numbers[,c(1,3)]
DF2
```

```
##      Colors MoreNUMBERS
## Row1    red           5
## Row2 yellow           4
## Row3   blue           7
```

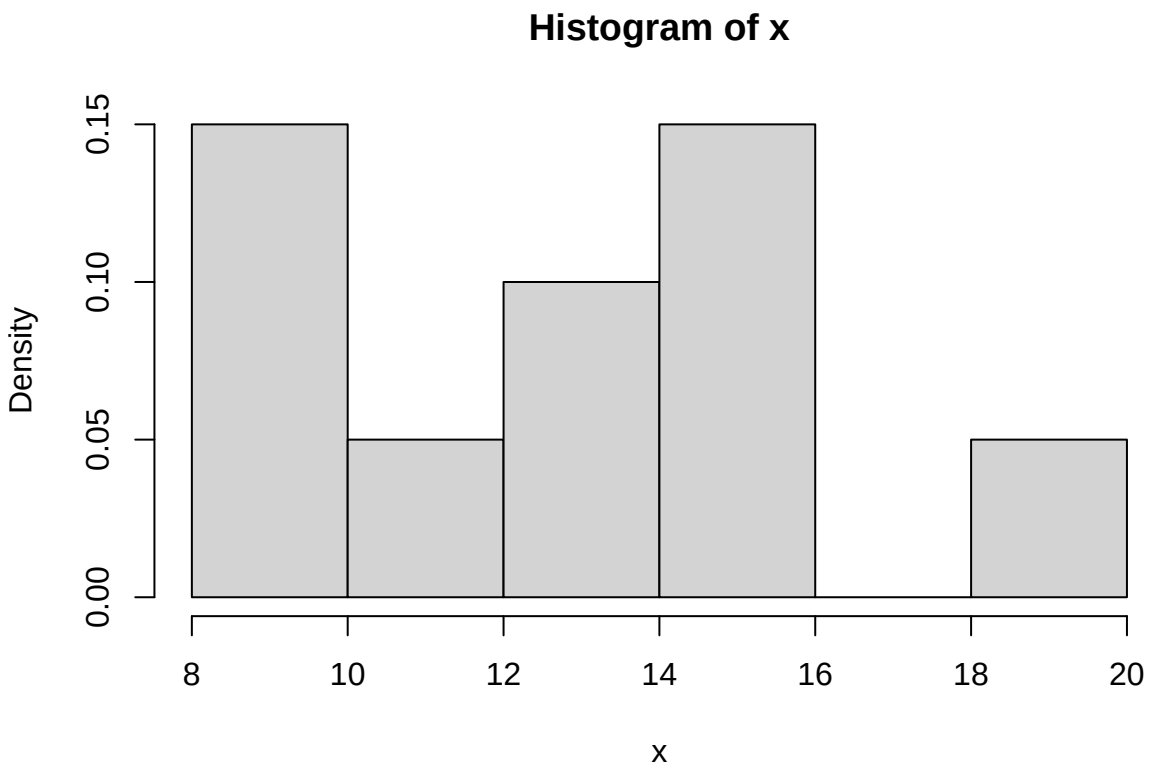
Built in graphic function—'hist()', 'plot()', 'curve()'

** Histogram **

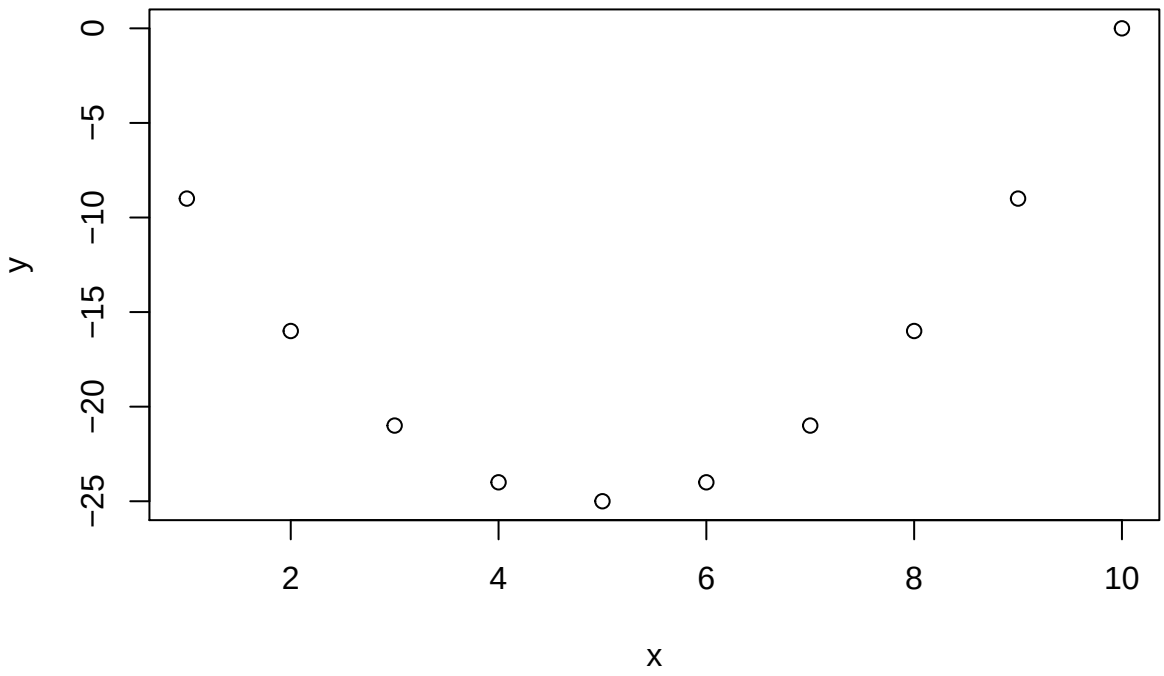
```
x=c(12,15,13,20,14,16,10,10,8,15)
hist(x)
```



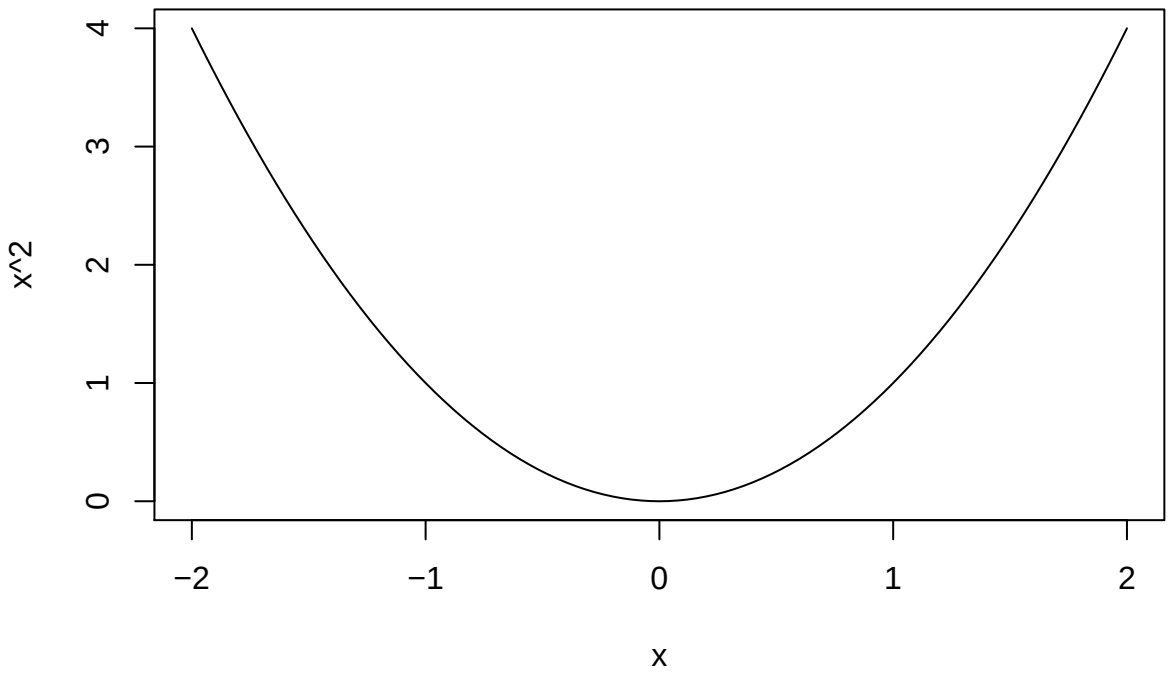

```
hist(x,prob=TRUE)
```



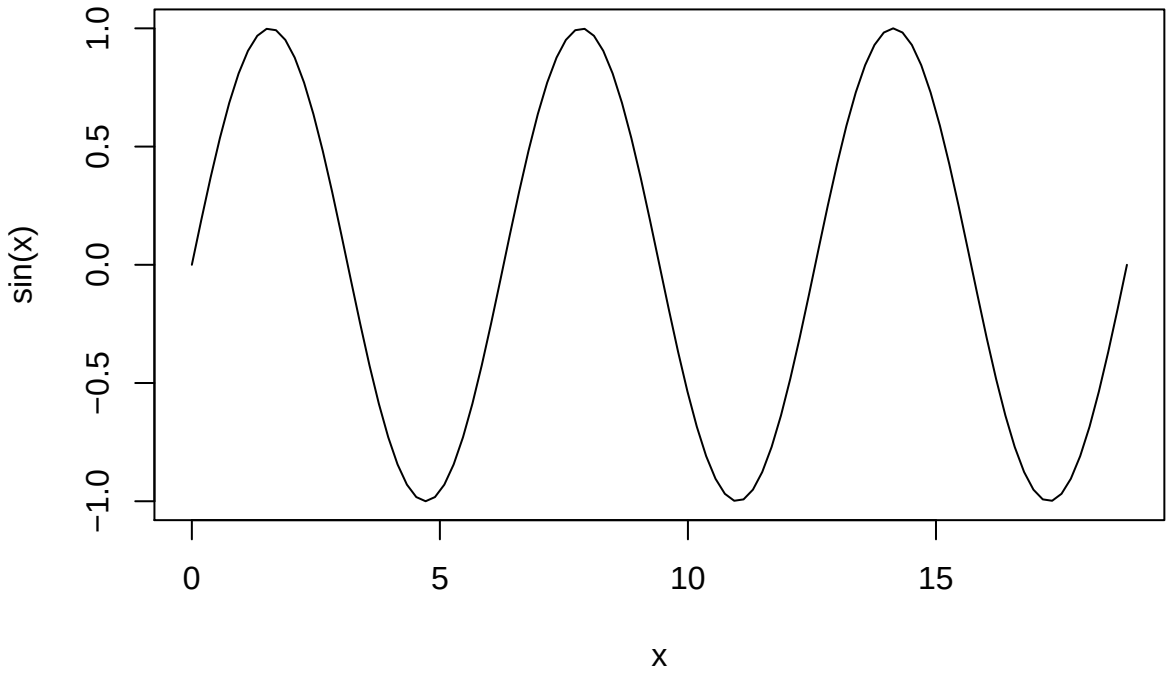

```
## Scatter plot
x=seq(1,10)
y=x^2-10*x
plot(y~x)
```



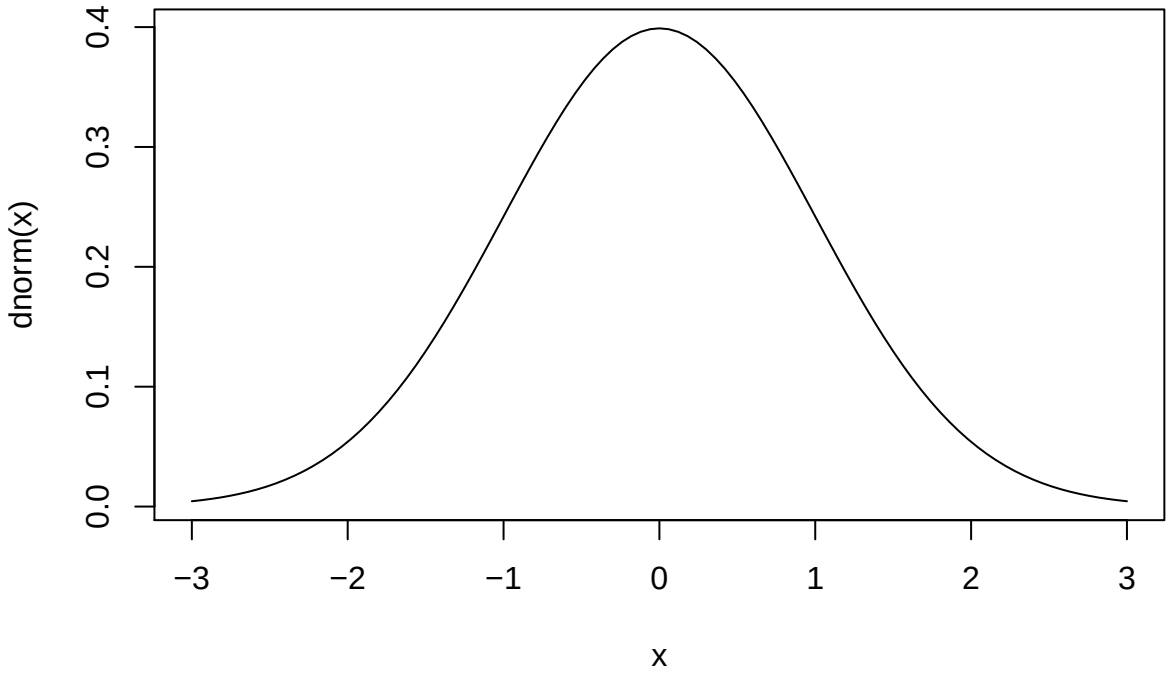

```
## Curve
curve(x^2,from=-2,to=2)
```



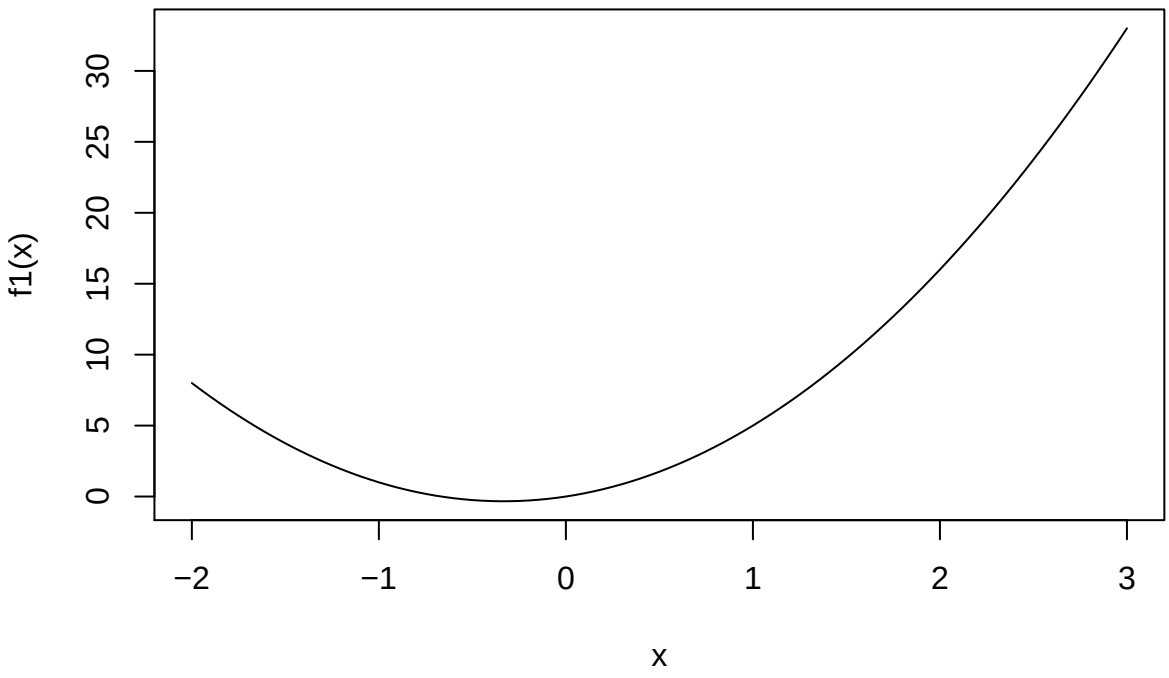

```
curve(expr=sin,from=0,to=6*pi)
```



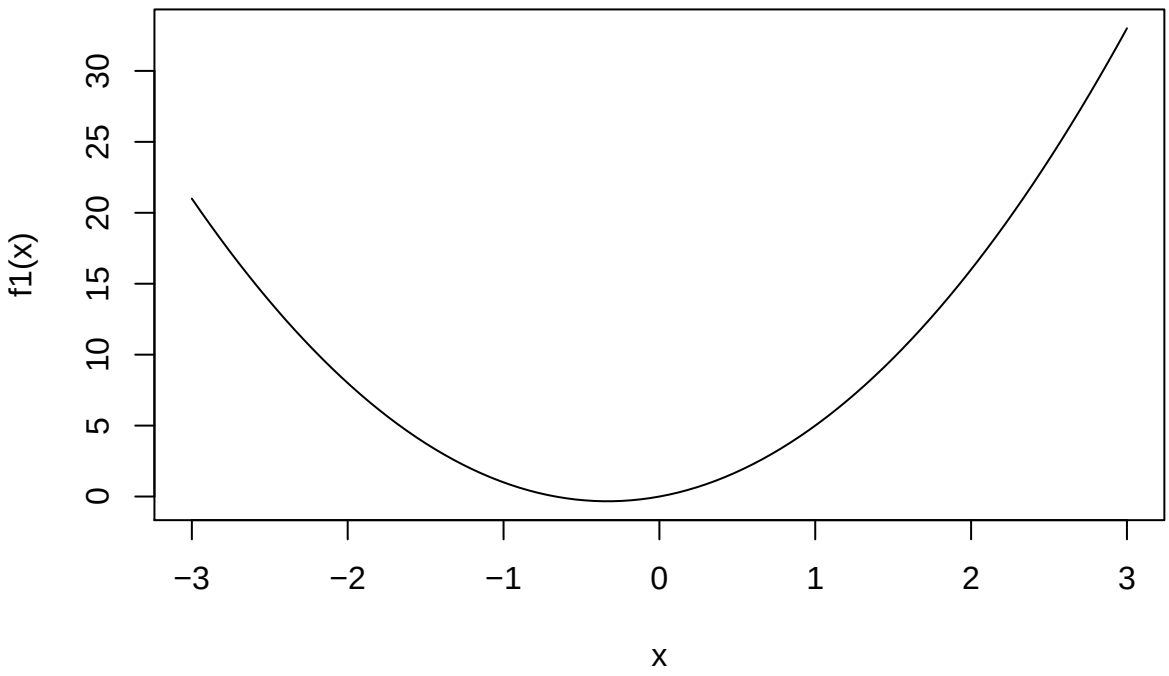

```
curve(dnorm(x),from=-3,to=3)
```



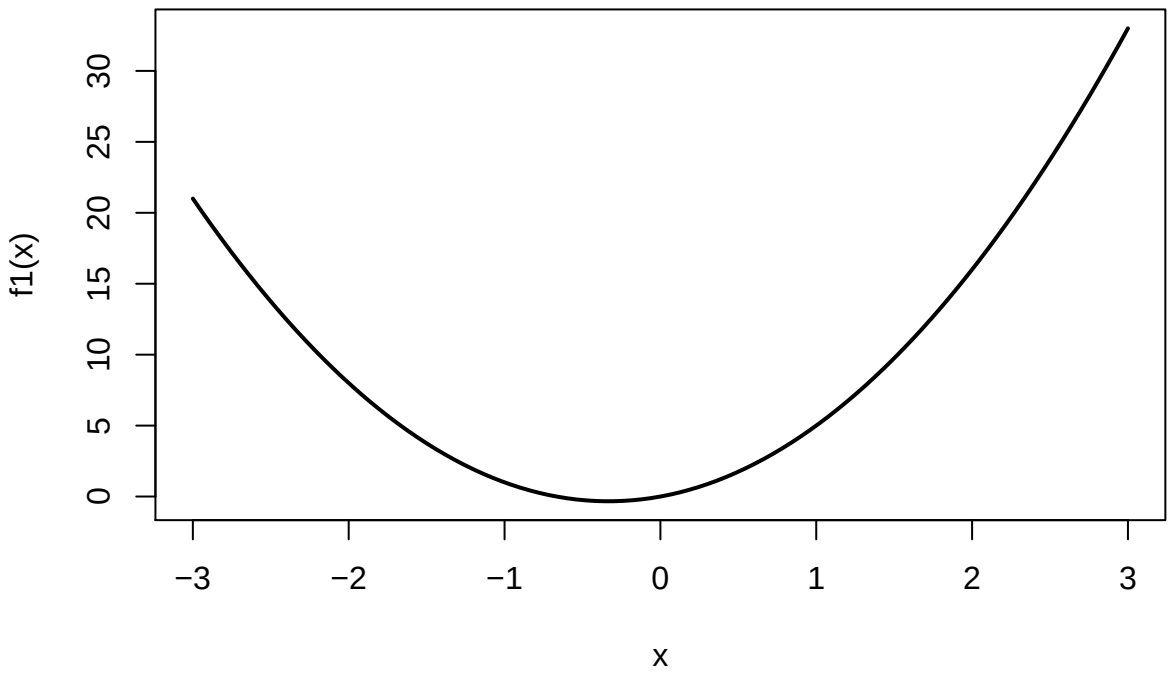

```
## Suppose you want to plot a function which is not defined in R.
# Plot f1(x)=2x+3x^2, -2 to 3
f1=function(x) 2*x+3*x^2
curve(f1,from=-2,to=3)
```



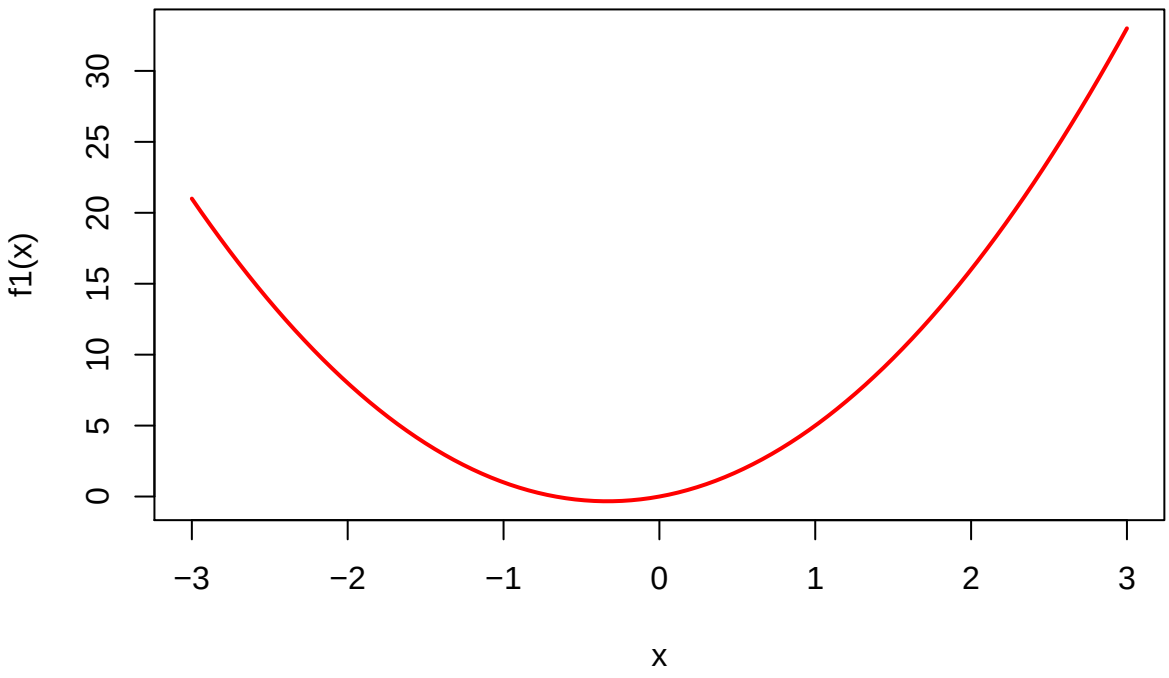

```
curve(f1,from=-3,to=3)
```



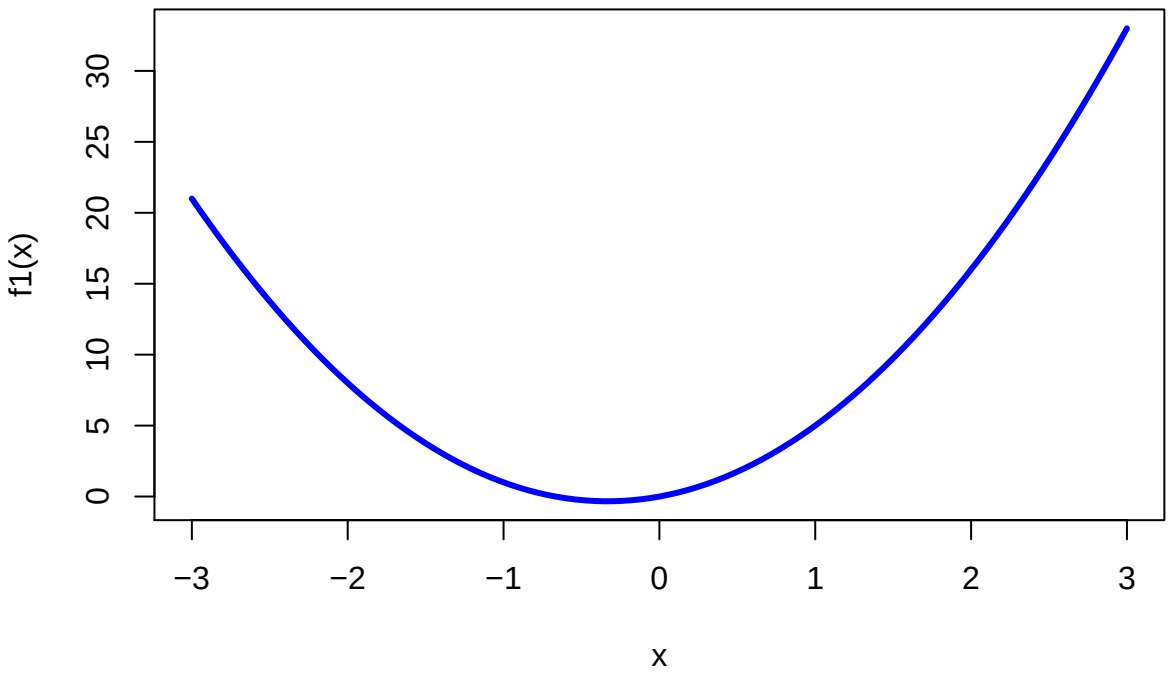

```
curve(f1,from=-3,to=3,lwd=2)
```



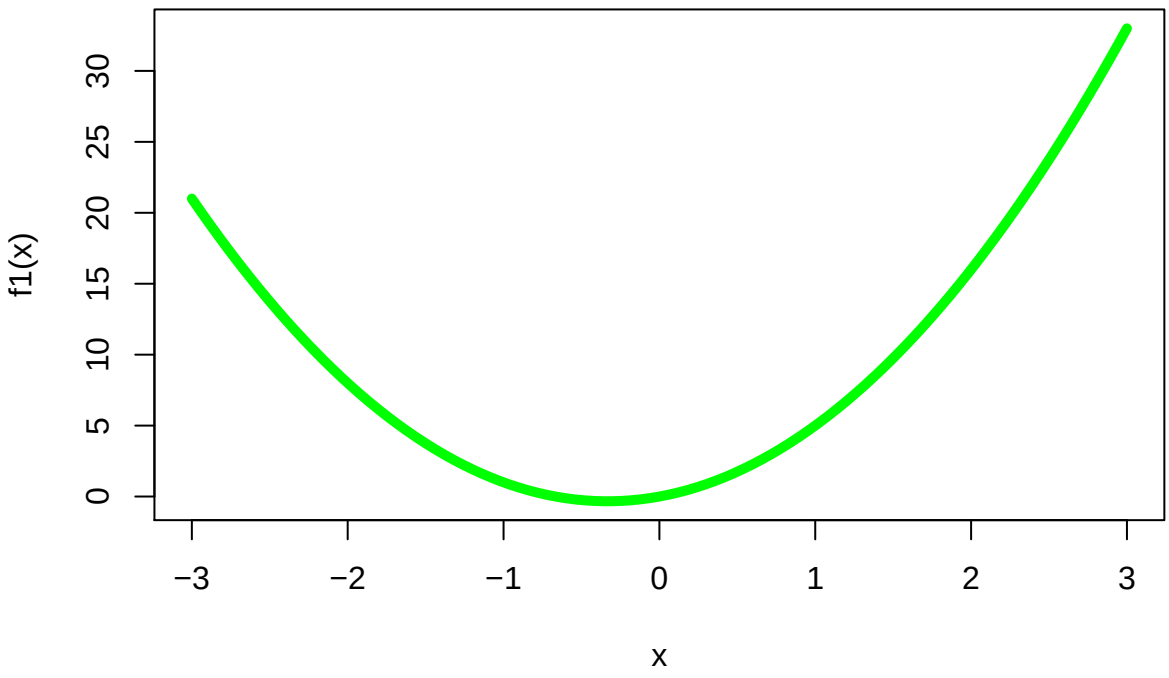

```
curve(f1,from=-3,to=3,lwd=2,col="red")
```




```
curve(f1,from=-3,to=3,lwd=3,col="blue")
```




```
curve(f1,from=-3,to=3,lwd=5,col="green")
```



Histogram, Frequency Curve, Polygon, Ogive for a data

```
# weight of 30 students
weight=c(68,53,69.5,55,71,63,76.5,65.5,69,75,76,57,70.5,71.5,56,81.5,69,59,67.5,61,68,59.5,56.5,73,61,7
dump("weight","weight.dat")
length(weight)
```

```
## [1] 30
```

```
# Create Histogram
h1=hist(weight)
h1
```

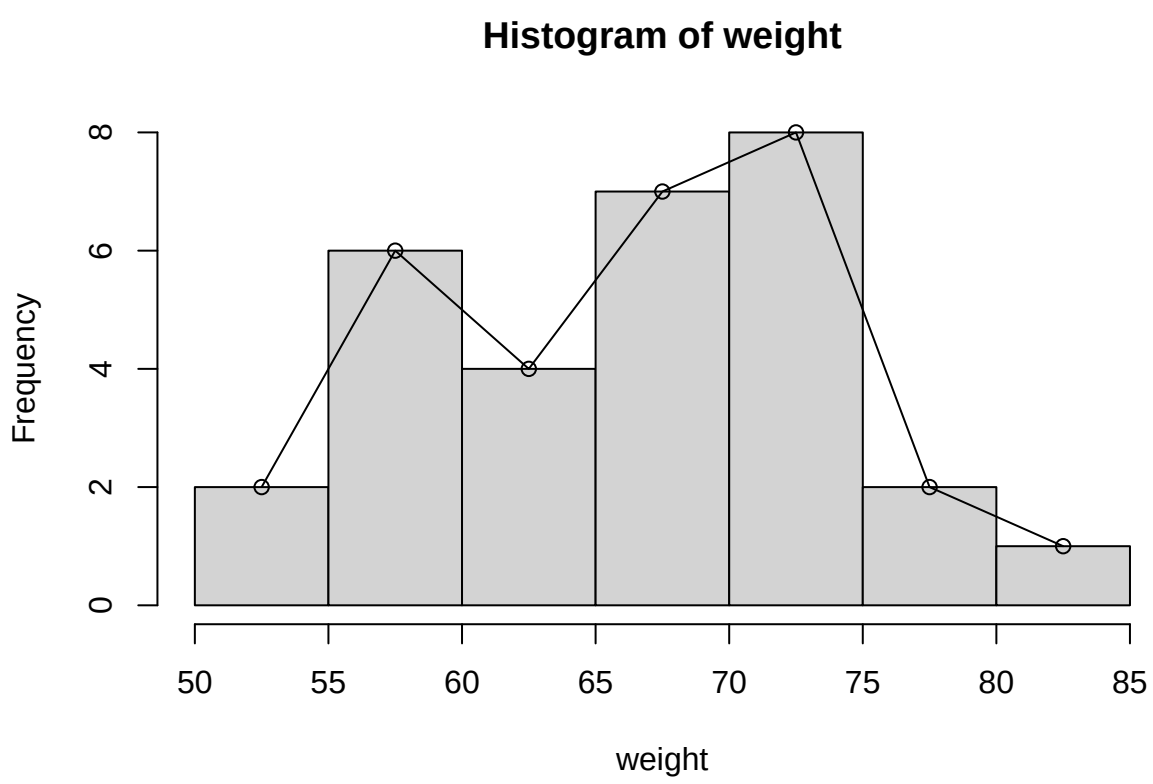
```
## $breaks
## [1] 50 55 60 65 70 75 80 85
##
## $counts
## [1] 2 6 4 7 8 2 1
##
## $density
## [1] 0.013333333 0.040000000 0.026666667 0.046666667 0.053333333 0.013333333
## [7] 0.006666667
##
## $mids
## [1] 52.5 57.5 62.5 67.5 72.5 77.5 82.5
##
## $xname
```

```
## [1] "weight"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

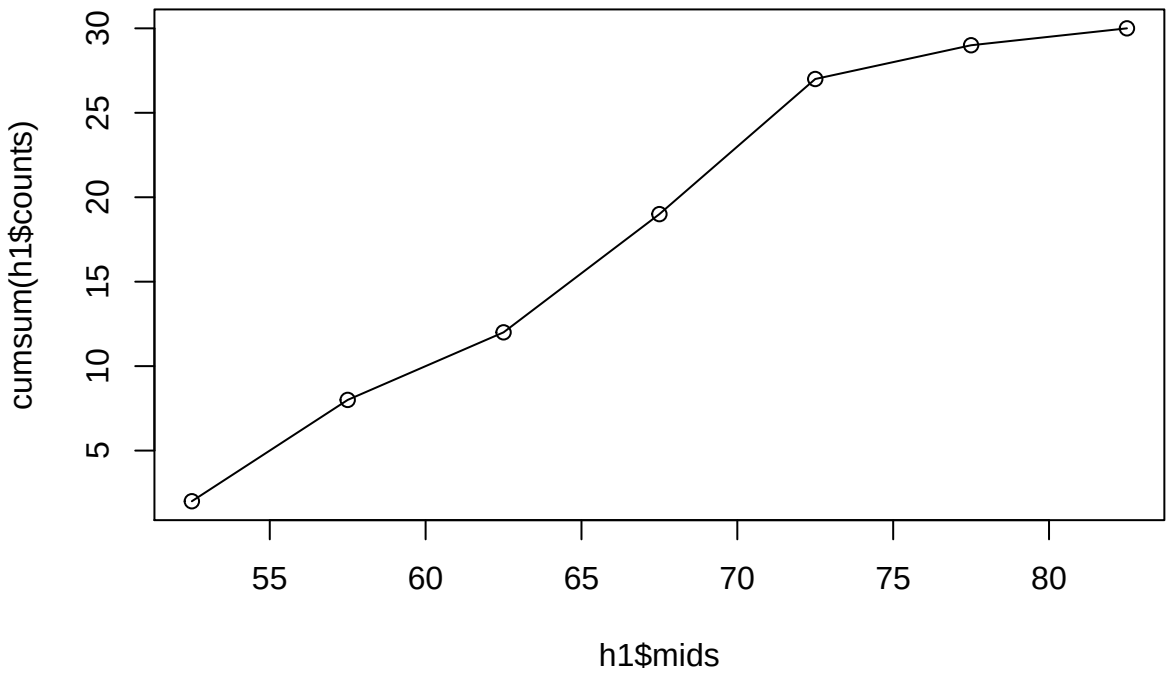
```
h1$breaks
```

```
## [1] 50 55 60 65 70 75 80 85
```

```
# Frequency polygon on histogram itself
lines(h1$mids,h1$counts,type="o")
```



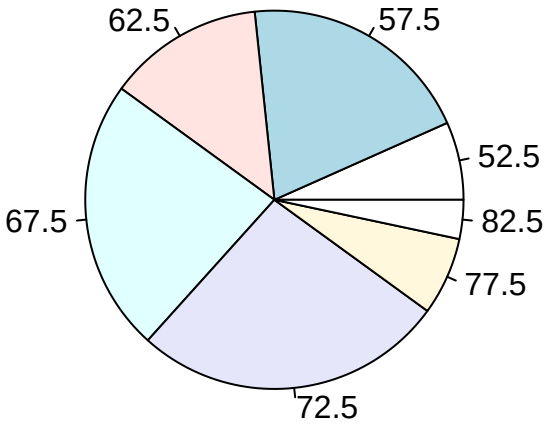

```
plot(h1$mids,cumsum(h1$counts),type="o")
```




```
stem(weight)
```

```
##
##   The decimal point is 1 digit(s) to the right of the |
##
##   5 | 3
##   5 | 56779
##   6 | 001133
##   6 | 688899
##   7 | 0112233
##   7 | 5567
##   8 | 2
```

```
# Create a pie chart for the same data
pie(h1$count,label=h1$mids)
```



##Graphics with R There are two type of graphic functions:

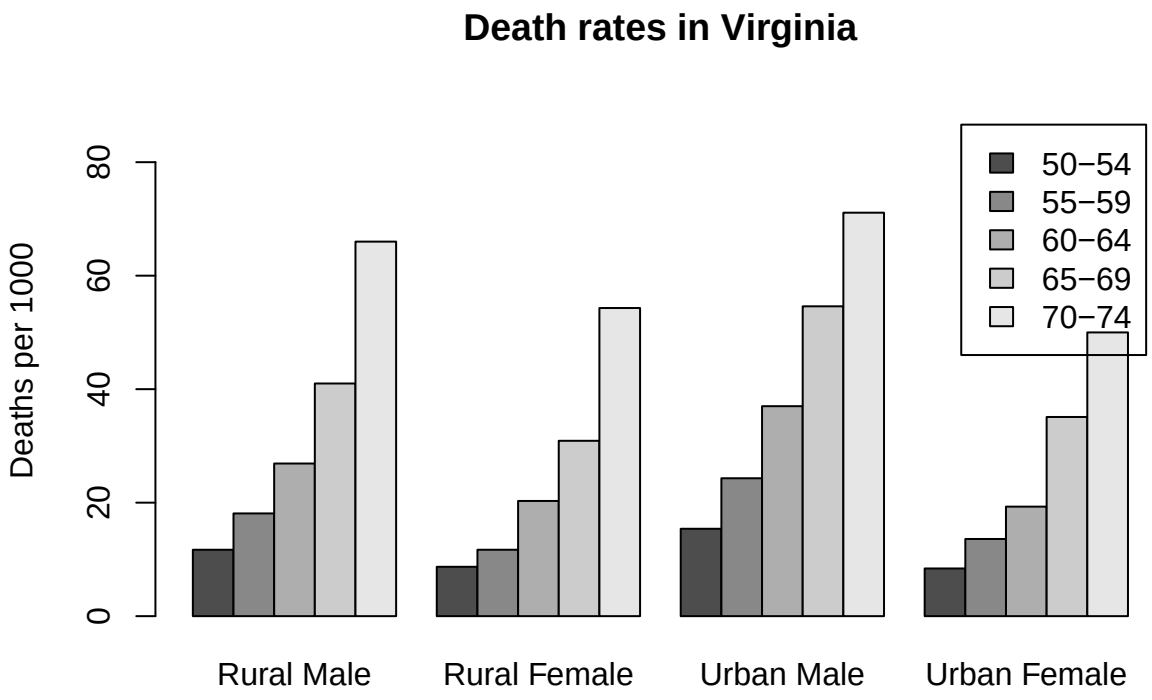
1. High level graphic functions which always create a new plot. Examples are; plot(),hist(), barplot(), dotchart(), pie(), curve().
2. Low level graphic functions: They add on existing plot. For example, lines(), points(), mtext(), main(), sub().

Example: Bar charts and dot charts The data VADeaths a data set in R contains death rates (number of deaths per 1000 population per year) in various subpopulations within state of Viginia in 1940. This may be plotted as barchart.

```
VADeaths
```

```
##       Rural Male Rural Female Urban Male Urban Female
## 50-54       11.7          8.7       15.4          8.4
## 55-59       18.1         11.7       24.3         13.6
## 60-64       26.9         20.3       37.0         19.3
## 65-69       41.0         30.9       54.6         35.1
## 70-74       66.0         54.3       71.1         50.0
```

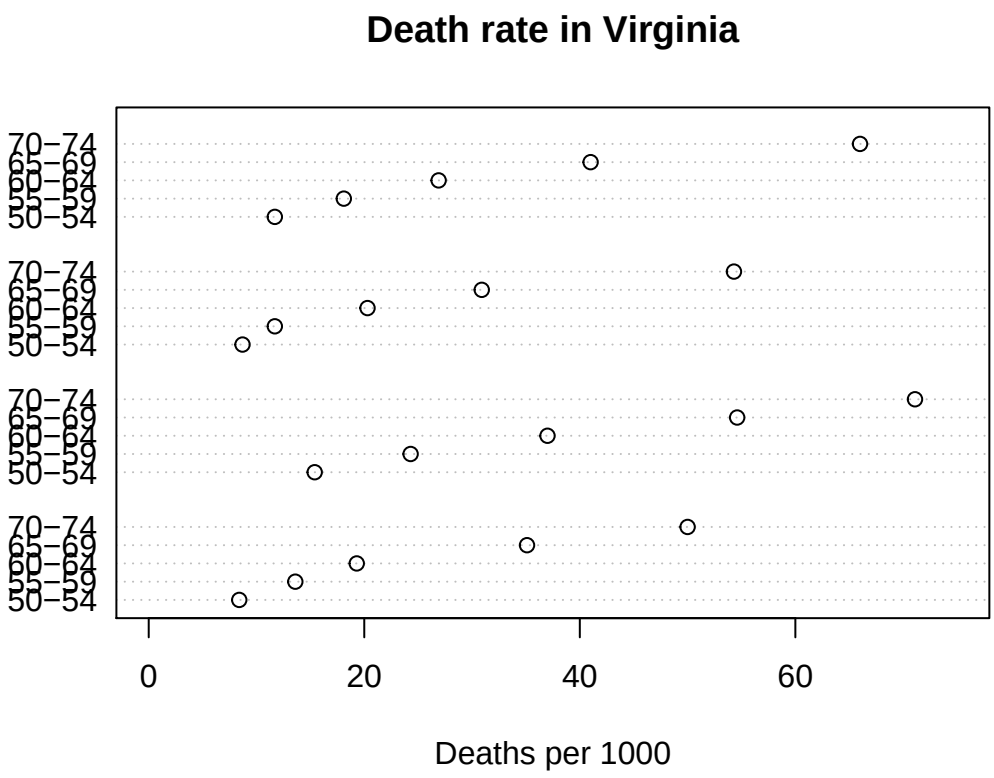
```
barplot(VADeaths, beside = TRUE, legend=TRUE,ylim=c(0,90),
ylab = "Deaths per 1000", main="Death rates in Virginia")
```



Dot Chart—'dotchart()'

Position of dot of corresponding number.

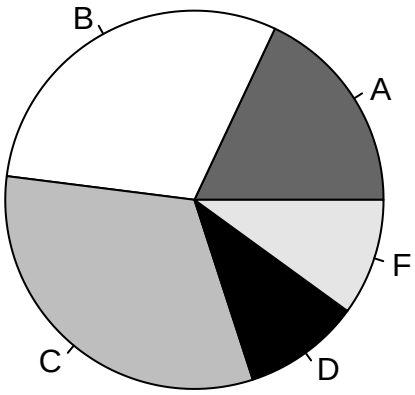
```
dotchart(VADeaths, xlim=c(0,75), xlab="Deaths per 1000", main="Death rate in Virginia")
```



Pie chart

Pie chart displays a vector of numbers by breaking up a circular disk into pieces whose angle (and hence area) is proportional to each number. For example, the letter grades assigned to each class might arise in the proportions shown in the following figure:

```
groupsizes=c(18,30,32,10,10)
labels=c("A","B","C","D","F")
pie(groupsizes, labels, col=c("grey40", "white","grey","black", "grey90"))
```



Histogram——hist()

Example for histogram

```
# simulate 5000 observations from N(0,1)
# make a histogram for the simulated data
x=rnorm(n=1000,mean=0,sd=1)
x
```

```
##    [1]  0.1052027104 -0.3084765708 -1.9991944892 -1.2592551526  1.5793188272
##    [6]  0.6267640375  0.6245298228  0.6818180820  0.1501646573  0.2379173873
##   [11]  1.0373886134 -1.5435226793  0.0690591245 -1.4123320962 -1.3784309980
##   [16]  0.0331350277 -0.5778872452 -0.8521736068 -0.4561704177 -0.0034668910
##   [21] -1.1259995016 -0.3204263716 -0.9013836436 -1.0294233224 -0.6264214813
##   [26]  0.0352620815  0.0668443206 -1.0116676377  0.1045350823 -2.4587902297
##   [31] -3.5688303218  0.0465314697 -0.1757626827  0.4510521704 -0.8581849776
##   [36]  1.1612630471 -0.1604437898  2.2169665525 -0.1179537327  0.3420286202
##   [41] -2.6651617940 -1.1289491477 -0.1791640809 -0.8830001109  0.8520587303
##   [46] -0.6203025608 -0.8658483430  0.5421691380 -0.6561837139  0.3531137151
##   [51] -0.2462745983 -1.3980889628  0.2470566082  0.2083143445 -0.3459443974
##   [56] -0.9562033580 -0.1449842623 -1.1213844753 -0.1029656661 -1.0147083239
##   [61]  1.5675404230 -0.9807919145 -0.2019073180  0.1466813370  0.5846578226
##   [66]  1.4184166920 -0.8350750803 -0.0749576766 -1.7432990409  0.2291369222
##   [71]  0.9326928551 -1.5969853623  0.6914682639  0.1682858253  0.1659984299
##   [76] -1.4310650525  0.0726533514 -0.3859728044  0.5674118621 -0.3010896485
##   [81]  0.0486396150 -1.1347819737  0.1911307367  0.9578724454 -1.1666822160
##   [86] -0.6575942340  0.4610191514  1.0774393011  0.9670699168  0.1908107757
```

```
##   [91] -1.7195928112  0.8400811605 -0.6959750825  0.1699992178 -0.6565531668
##   [96] -0.5289228390  0.1784596518  0.4245585934 -0.8998594154 -0.8218354145
##  [101] -0.1738457843  0.2269847437 -0.3162142293  1.1655313704  1.5875967579
##  [106] -0.0575688413 -0.0963126382 -0.8536239916  2.2044578193  0.2804928004
##  [111] -0.1014476749  0.7888834137  0.8756482979  0.2279183396  0.3118114044
##  [116]  1.4113044272  0.4157012733  1.6260145571  0.8880895572  0.6492714074
##  [121] -0.9807465766 -0.5654142064 -0.1133899496 -0.4666885893 -1.3128286225
##  [126]  0.6124603027  0.0049611289 -1.6466192509 -0.1989238988 -0.2933299581
##  [131]  1.3974790020  1.1130106461 -0.4073935484 -0.7902502667 -0.1013752638
##  [136]  0.2307602828  0.5509306623 -0.5589613449  0.9247838178  0.1707238867
##  [141]  1.3922475798 -1.7338151572 -0.7914079395  0.3028176460 -1.0500866354
##  [146]  2.5468751686 -0.1885388673 -0.9333131067 -0.4477550035  0.4401363750
##  [151] -0.0687636336  0.9668397019  1.2383406833  0.1051428200 -0.3569355029
##  [156] -1.4755981108  1.0744156467  0.6995810117  0.6595586968  1.0933899056
##  [161]  1.6674965341 -0.1689257657  0.5381289693  2.0375669592  0.1895509115
##  [166] -0.8735521338  0.1778286736  0.5200513976  0.3230036785  0.6663964867
##  [171] -1.5179967916 -0.8352607017 -0.5234986259  1.2243270496  0.3391618056
##  [176] -0.7445009304  0.7883244860  1.0118170807 -0.3217718242  0.2725596357
##  [181]  1.2065105532  1.4250255951 -0.3620283752  0.5105628113 -0.0846126109
##  [186]  1.7511430817  0.5343738239 -1.1548577719  1.4350530950 -0.0216621966
##  [191] -0.3472545953  1.5845184088  0.4402589134 -1.8569097517  0.9076399240
##  [196] -0.4518648388  0.7439258494 -2.0792832444  1.4386825152  0.2219914685
##  [201] -2.5954610239  0.5378439762  0.0272994922 -0.5648790618 -1.3364650387
##  [206]  2.5244622931 -0.5111638499 -1.6959042005 -0.4698322054  2.1965000981
##  [211] -0.0630407553  0.2879151411  0.8910188283  0.7280028830 -1.4951310514
##  [216] -1.5651641011 -0.4511219004  0.2366857917  0.0961083350 -1.3676449059
##  [221] -0.6813603705  0.6502673806 -1.1120961650 -1.9681840673 -0.7543566257
##  [226] -2.2735438379  0.1506340248  0.5868978097 -1.8420816882 -1.2339850536
##  [231] -0.9533834973 -0.1249277848 -0.1271370953  0.9014724587  1.3110041238
##  [236]  0.8069316595 -0.3594000684 -0.6207323475 -1.1160904331 -0.4353137292
##  [241]  0.6672204083  1.4973879956  0.9107660519 -1.4091453938  1.2173068013
##  [246] -1.1492953563 -1.3684128411 -0.0730945478  0.1800147958  0.2609138569
##  [251]  1.1313788034  0.1538538694  1.4256169068 -1.6877816105  1.0013975673
##  [256] -1.3641847409  0.1174426583  0.8208511231 -0.6341333848  0.2201309878
##  [261]  0.6842506971 -1.2289595124  1.2524055457  0.8348092949  0.1812652751
##  [266]  0.0474160154  1.0777705878  0.4022993938 -1.1633985203 -0.7023255491
##  [271] -0.4431680420  0.1456128839 -0.6445655471  1.2995331668 -1.7893169535
##  [276] -0.5858325405  1.3311752640 -0.6345734588 -0.6400960710  0.2693007585
##  [281]  0.6512966155 -0.4384703825  1.3187459000 -0.3677816442 -0.6900786151
##  [286] -0.4660907494  0.4575097565  0.3393484762 -0.7699902701 -0.2549630915
##  [291] -1.1268630429  1.4612449127 -1.2557479146 -0.3114631826  1.5136045302
##  [296]  0.8821568069 -1.4286746561  1.2864183151 -0.7991648816  0.3539004344
##  [301]  0.6094324860 -1.6676753134  0.0643147123 -0.7484311600  0.5760913747
##  [306] -0.0554159399 -0.5454665191 -0.7455273103  0.4134778061 -0.7010132232
##  [311] -0.3740483748  0.2107054696  1.4370052935 -1.1288042113  0.4128456445
##  [316]  1.7499364128  0.2257759058  0.1353673248  0.9503988255  1.4729942495
##  [321]  0.0546690468  0.9252116298 -0.2583306837 -0.6453280605  0.4664672133
##  [326] -0.3773913254  0.8895449281 -0.1814732856  0.9833232154 -0.8083995003
##  [331] -1.0832739613 -0.1774232789  1.0885164893 -0.6456685326 -0.1069787301
##  [336]  0.7234066815  0.4978460957  0.8599051522 -0.4844537809  0.2529832479
##  [341] -0.9461647407  0.4818313549  0.7543957530  0.4900184726  0.4532425155
##  [346]  0.3916823827 -1.7750953470  0.0180954682  0.3815226496  0.3620705204
##  [351]  0.4630417066  0.9414078026 -0.2879808697 -2.6911049488  0.0746920355
##  [356]  0.8910518733 -0.6054870149 -0.5977316592  1.0618992581  1.1654253391
```

```
##  [361]  0.5369101534  0.5602074660  1.5224977375  1.0220457390  0.0688876845
##  [366]  1.6199003681 -1.0061326587  0.1344431811  0.0683454913 -0.1694593948
##  [371]  0.6518249329  0.4411673435  1.0793245579  1.2751307371 -0.3862067638
##  [376] -1.1223930753  0.2164374462 -1.9918739892  0.9221618832 -0.7327630721
##  [381]  0.4911115749 -1.1553708047 -0.5100007149 -0.0251241815 -1.3676451999
##  [386] -0.7236398964  0.5785490815  0.4336505236  2.0831461374 -0.3080503355
##  [391] -0.4612134678  0.0534535819  0.1826402577  0.7078339786 -0.1579034787
##  [396] -1.1363286380  0.7926402033 -1.3263500846 -0.2515042767  0.1483118920
##  [401]  2.0187206211 -1.8719421698  0.3643159842 -1.1812016156  1.0055581894
##  [406]  1.8585336206  0.6967382546  1.1180347156  0.0632981088 -0.3166005403
##  [411] -0.0082325032  1.3966649232 -1.8000457815  0.3091612862  2.1985720784
##  [416]  0.5593555350  2.4758751663 -1.1160111957  1.3571186617  0.2545411461
##  [421] -0.4294326920  0.4084511380  1.2914141588  0.7421973875 -0.6272211969
##  [426]  1.2104899428 -0.8421681612  1.2733465154 -0.4320714270 -0.9991931647
##  [431]  1.8523205471  0.5099135131  0.1730245508  0.1209467046  0.6388266151
##  [436] -0.0063769555  3.2491017242 -0.2407815095  0.8512080998  0.9236720211
##  [441]  0.4386986851 -1.1363464083  1.0491711592  0.7430413586  0.2589553481
##  [446] -0.0978904039 -2.4357249383 -0.0595340025  0.3058945893  0.9339393832
##  [451] -0.8100639840 -0.2536204522  2.3039094358 -0.1858694693 -1.8480310653
##  [456] -2.1582173179  1.6566962590 -1.5558396952 -0.0963454255 -0.5787699643
##  [461] -2.1028935978  0.3582749986 -0.0302349506  0.8065050939 -0.2631642778
##  [466]  0.5502823974 -0.5908586043  2.6580136150 -0.4561430892 -1.4637787347
##  [471] -1.4102223741  0.6198559593  0.3762934722 -2.1745974014  0.1834934627
##  [476]  2.0217096387  0.1677510736  0.1735642648  0.0499834596  1.1300488592
##  [481]  1.1992098497  1.0483870242 -1.3800319508 -1.0752546201 -0.8395064270
##  [486] -0.3631365174  0.6214324507  0.3208482584  0.4878693130  1.0742418239
##  [491]  1.1465607699  0.1369979558 -0.2491245495  0.0671724801  2.1038093080
##  [496]  1.1854580009 -0.7068027134  0.4879895764  0.0407581689  0.6211853096
##  [501]  0.3625624964  0.9327306767  0.7455936101 -0.1700269426  0.7510696711
##  [506] -0.6520668017 -0.7093732359 -0.5224539410 -0.7547623311  1.1375380152
##  [511]  1.9376564045  1.1292128041 -1.3415004821  0.9315631595  1.2801033844
##  [516] -0.7762644298  1.8372267539 -1.0879882008  0.0842860498 -1.1859382630
##  [521]  0.3506995259  0.0138166687 -0.3440484298  0.1734894839 -0.8973319085
##  [526]  0.4411204965  0.7772015241 -0.4842702264  0.0276941960 -1.6264775762
##  [531]  0.5510677837 -1.3993937717 -0.7117006949 -0.6042986246  0.0993127761
##  [536] -1.0144804310  0.0523984686  0.6100626040 -2.8397638187  1.2008467846
##  [541] -0.4397281872 -0.5548972576 -0.0516636355 -0.6927099539 -0.0434456984
##  [546]  0.3902152639 -0.2551238916 -0.5128040769  0.3886034853 -0.7339266456
##  [551]  1.7119947679 -0.3409401040  0.3366177093  0.9388275419 -0.9053738581
##  [556] -0.9394470007 -0.0685781419  0.8399506315 -1.6211690756 -0.2935927913
##  [561]  0.5399090660 -0.1717004356 -0.2192936584 -0.2261401202  0.0372826503
##  [566] -0.2483951982  0.0259964518 -2.1045818961 -0.2595815585  0.4904306801
##  [571] -0.9995704120 -0.1059164829  0.8727722235  1.5817224653 -2.4148036510
##  [576] -0.3744030521 -1.3132690198  0.1433696252  1.0299215434 -0.7869189848
##  [581] -0.1491730581 -0.5221040292  0.0235466528  0.4113727408  2.3013096999
##  [586] -0.1494787879  0.8569974761 -0.6212589088 -1.1340652331  1.6333203513
##  [591] -0.1913717114 -2.4118160840 -0.5736726221  0.0770869903  0.6523229732
##  [596] -0.4716532701 -0.0376683653 -0.9460869429  0.4914207853 -0.4135264189
##  [601]  0.6528027004  0.2540445518 -0.2032116910  1.8070770584  1.4550136506
##  [606] -0.0053233175  0.0518906505 -0.4546570180  1.3281436312  0.0462262655
##  [611]  0.0147771903 -0.1919462679  0.4388678388  0.1523046079 -2.9067176981
##  [616] -2.4909182970  0.0994884329  0.1274256399 -2.4371692045 -1.8594485222
##  [621]  0.0813116292  1.4159292774  0.3486218998 -1.4207706902 -1.9215490998
##  [626] -0.7726821065 -0.0005012126 -0.2150471166 -0.2940481386  0.5957549172
```

```
##  [631]  0.0484794255 -1.4567558766  0.8549425813 -0.2952179128 -1.1708883058
##  [636] -1.5000093180 -0.0608286057 -0.2413332070 -0.2834647089 -0.0441799299
##  [641]  1.0459804018 -0.9576570239  1.0225385238  0.7439837178  0.5331668499
##  [646]  0.7060297393 -0.6852480465 -1.6466890665  1.4299137228  0.9483901181
##  [651] -0.3058155547  0.2987651448  0.5495639657 -0.7265111546 -0.0028806894
##  [656]  0.0186043226  1.1535949772  2.0648172409 -0.2554885201  0.8947314630
##  [661] -0.3770896729  0.0983002457  0.5137021504  1.3909160498 -1.5035961486
##  [666] -1.6003586976 -0.0013803676  1.0474044083  0.1251831535 -0.0265356640
##  [671]  1.1868421510  0.4439086050  0.0768176245 -2.1569803024 -0.8504961122
##  [676] -1.1232536799 -1.1699511080 -0.1452479038  0.8542500523 -0.6599792482
##  [681]  0.9319853742  0.0579472033 -0.9820438687  1.8802739764 -0.1008647899
##  [686] -0.0734158167  0.9868356933  0.9315324371 -1.3266730975  0.1577235524
##  [691] -0.1766484451 -2.5077265526  0.3322221448  2.3176644164  0.9823623344
##  [696]  0.4820656963 -2.1583800059 -0.3447182531  0.4356504053  0.7000435512
##  [701] -0.5550500595  0.8396129701 -3.0126760394  0.8932572784  0.4116521500
##  [706]  0.5927524494  1.4281123939  1.2148502332 -0.4414632105  0.2505976520
##  [711]  0.1602265552  0.0009086663  1.4064235397  1.0726732613 -0.2576698938
##  [716] -1.0971736924  1.7284095131  0.2527693136 -0.9039969088 -0.9995814735
##  [721]  0.9080467147 -1.6173018121  0.6797498558 -0.6946089162  0.5701026020
##  [726]  1.0525166689 -0.6804526971 -0.5213425671  1.4720774167  0.7366156850
##  [731]  1.9537317407  0.4216749305 -0.6537705656  1.5652562489 -2.2648436202
##  [736] -0.9640090172 -1.7587481583 -1.0608551543  1.3306306743 -0.2768349477
##  [741] -1.3791039085  0.0931646944 -0.9548161987 -0.7371503139  0.0689227656
##  [746] -0.6866724845  1.1393058863 -0.5972667457  0.3758184412 -1.6589496966
##  [751]  0.4746461459  0.5298100656 -0.4312393647 -0.9229301179  0.8387387609
##  [756] -1.1593941938 -0.4114716885  0.5895578954 -0.6429984301 -0.1560678185
##  [761] -0.0035626044  2.0070731473 -0.5203049806  0.1937508641  0.3818842491
##  [766] -0.0052266821  0.8244729570 -0.2811498335 -1.6988992704  0.4712659307
##  [771]  0.8581695390  0.2989241903 -0.3499983929 -0.2062306213 -0.9400259429
##  [776]  0.4410100153 -0.6352942324 -0.2786548951 -1.1897814308  0.7851655298
##  [781] -1.5950998714  1.5843817796  0.1407517798 -0.0214025196 -0.4379671203
##  [786]  0.8977842903 -0.2990392767  0.6718139973  0.5534570867  2.4724206881
##  [791]  0.1205916603  1.0456202019 -0.9659062950  0.7750355264  1.7055130224
##  [796]  0.1382093072 -0.4601148039  0.1422352084  0.2099944341  0.3637768760
##  [801]  0.8340233440  0.0335420359  1.3766338029 -1.9486577781 -1.1207747079
##  [806] -1.8314741393 -0.0281773806 -1.4210951208  1.1647010610  1.5531872205
##  [811]  0.9827898326 -0.1973057872  0.0280318314 -1.0454741363 -1.8136693957
##  [816]  0.0653278581  0.5278649665  0.1627075625 -0.8755234630  0.8531160203
##  [821] -0.0906961609 -0.9300298726  1.8167511198  1.5275041158  0.4032325419
##  [826] -0.7546789507 -1.7422831341 -0.8816924484  0.7235094646  1.1419528666
##  [831] -0.6552287030  1.2121221418  0.8943051736 -0.1129252332  1.5496022388
##  [836] -0.1130221699  0.0839146136  0.4486227943  0.0350383456  0.5471932345
##  [841] -0.2598685992 -1.4675073321  0.8182876709 -1.2301283516  0.1918708476
##  [846]  0.2919187293 -0.1043523814 -1.1127156872 -0.4347665885  1.0506105803
##  [851]  0.6881141715  0.0733442409 -1.2006284645  0.0206409712  2.0073643922
##  [856]  0.5839071433  1.4668370261  0.9858515025  0.7285240503 -0.1601254525
##  [861] -1.8973019980  0.3331529880  1.1781330055  0.4597983691  0.9820564572
##  [866]  0.2899441273 -0.8074136493  0.3725523283  1.0428911546 -0.1479755900
##  [871]  0.1562790529 -2.0779659265 -0.8866766464 -0.9501794831 -1.0165011328
##  [876]  0.6387416549 -0.1770200920 -0.7783150587  0.7822870094 -0.2785040906
##  [881]  0.3997581189 -0.6662015381 -1.3724456403 -0.3319717202 -0.0403081738
##  [886]  1.3354071324  1.1892709932 -0.2346283365 -1.5874091653  1.5866449758
##  [891] -0.9989376832  0.5640218976 -0.0892996120  1.0470006553  0.8135908856
##  [896] -0.9954765158 -0.3463619388 -0.5933120214 -0.3309285183  0.3121161818
```

```
##  [901]  0.7452905642 -0.4276471429  2.7793908680  1.0206158319  2.2532827062
##  [906] -1.1756605366 -0.6839750347 -0.5319610886 -0.5564445108  1.0673607102
##  [911]  0.6920239465 -0.0029114938  1.1581824384 -0.4930121712 -1.0276111523
##  [916] -0.8492165439 -0.1807653499 -0.1750079387 -1.3202893511  0.3249313711
##  [921]  0.0088643935 -0.0973182768  0.6575433881  0.3630981205 -0.6009487788
##  [926]  0.7906476458  0.4156405981 -0.9086425739  1.4002090887 -0.8347679498
##  [931] -0.7608005604 -1.3740274693  1.6302546539 -1.3428767439  0.4501170731
##  [936] -0.9245984023 -0.3679011715  0.5585920186  0.4462699727  0.7506643364
##  [941]  0.3640865814  0.3939266095 -0.3298141917  0.9926653608  0.8936095778
##  [946] -1.7666248398  1.0631974822  0.3966203546  2.2214320993 -0.0654818174
##  [951] -0.6862264486 -0.4437560056 -0.4626628299 -0.7620189389  0.0633896593
##  [956]  0.0553337951  0.3599691186 -0.4893066579  1.3655698356  1.0665707930
##  [961] -0.5342244420 -0.0382079474 -0.8663823251 -0.6322328415 -0.8708468134
##  [966]  0.2371176074 -0.1174438250 -0.7638960137 -0.7877759840 -0.9419285099
##  [971]  0.7886895826  1.0540125484 -2.3116646312  0.0417254140  1.4316062844
##  [976]  0.9629673523 -1.7548757654  0.0588601866 -0.7861373398 -0.0442734031
##  [981]  0.1987368535 -0.6053162548 -0.8075372192  1.4326539650  0.0540148218
##  [986]  1.7939045857  0.6346471813 -1.2422044083 -0.0555730726  0.0829397507
##  [991]  0.1665541147  0.8086654576 -0.1366476158 -1.6932806546 -0.9177512378
##  [996] -1.2480091310  2.2352144987 -1.0876223086  1.9978945980  0.9418505676
```

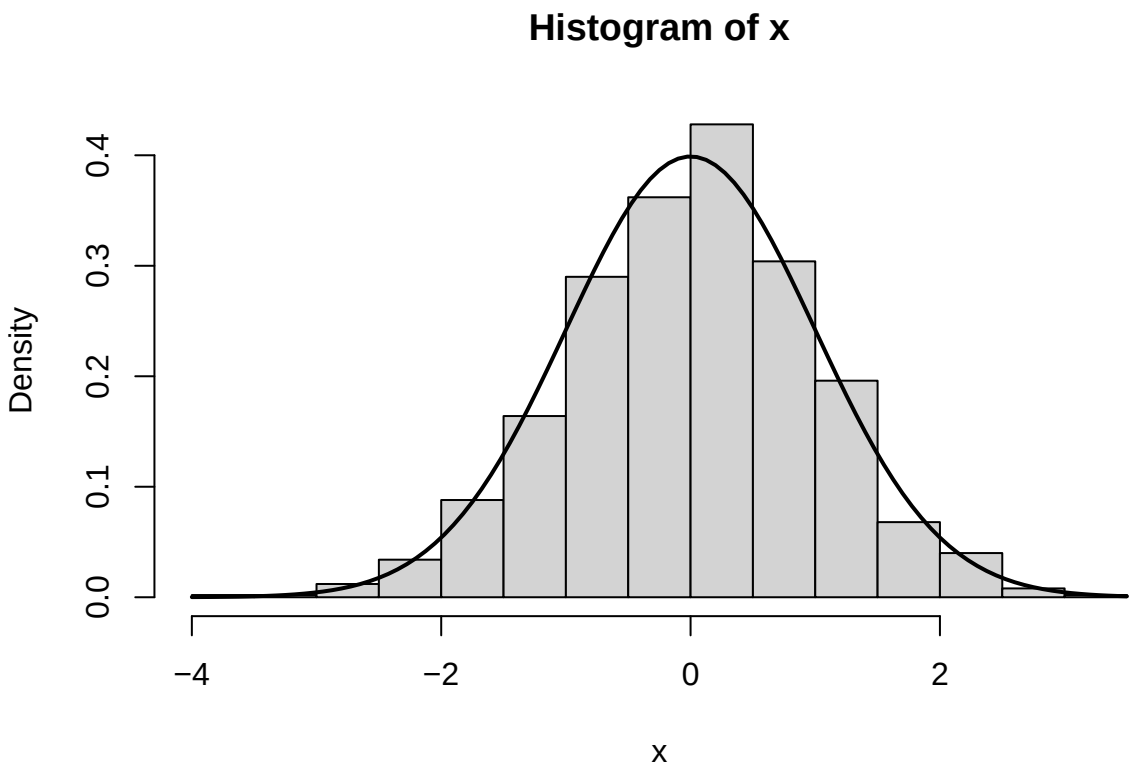
```
mean(x)
```

```
## [1] 0.01007682
```

```
sd(x)
```

```
## [1] 0.9975375
```

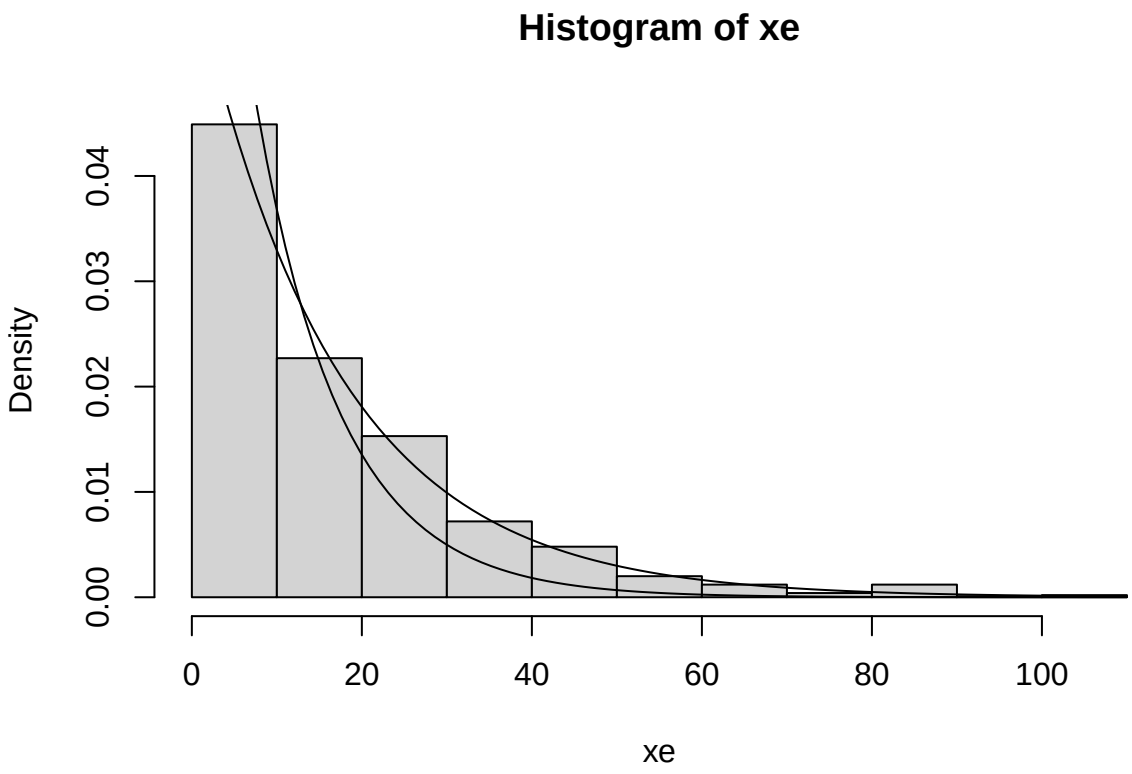
```
hist(x,prob=TRUE)
curve(dnorm(x,mean=0,sd=1),add=TRUE,lwd=2) #to add on histogram
```



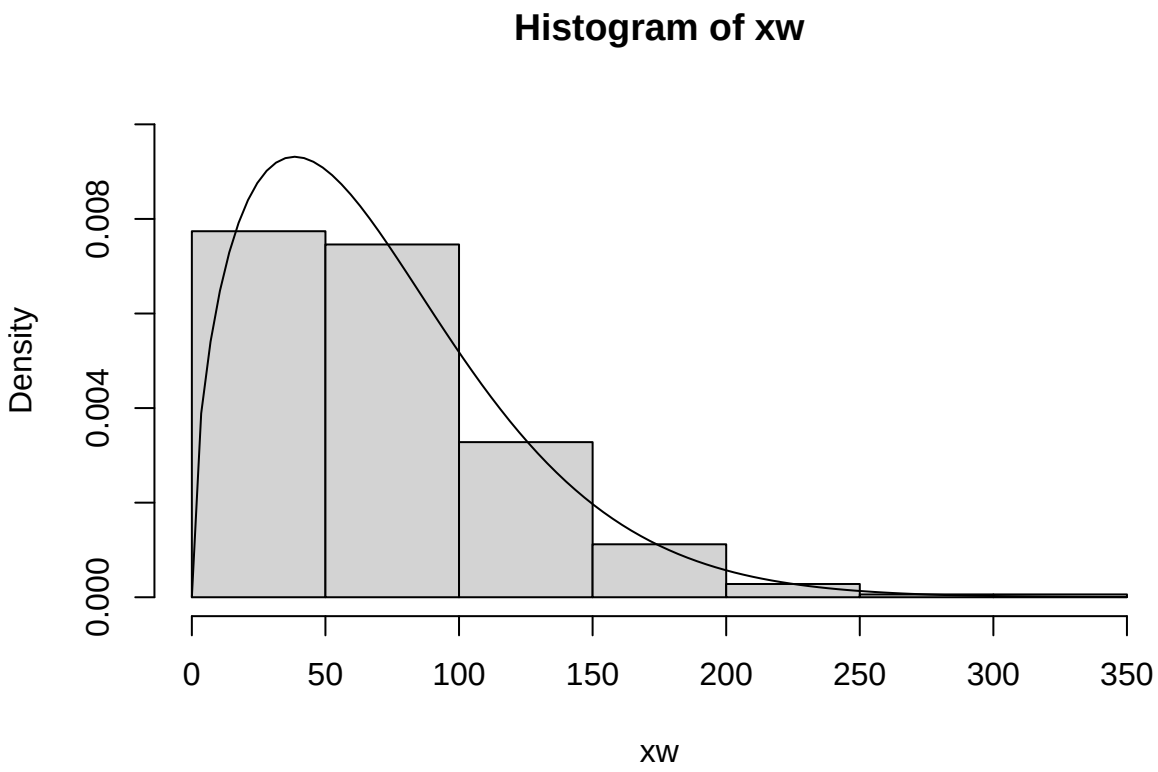

```
# An example with exponential
xe=rexp(1000,rate=0.06)
mean(xe)
```

```
## [1] 17.07742
```

```
hist(xe, prob=TRUE)
curve(dexp(x,rate=0.06),add=TRUE)
curve(dexp(x,rate=0.10),add=TRUE)
```

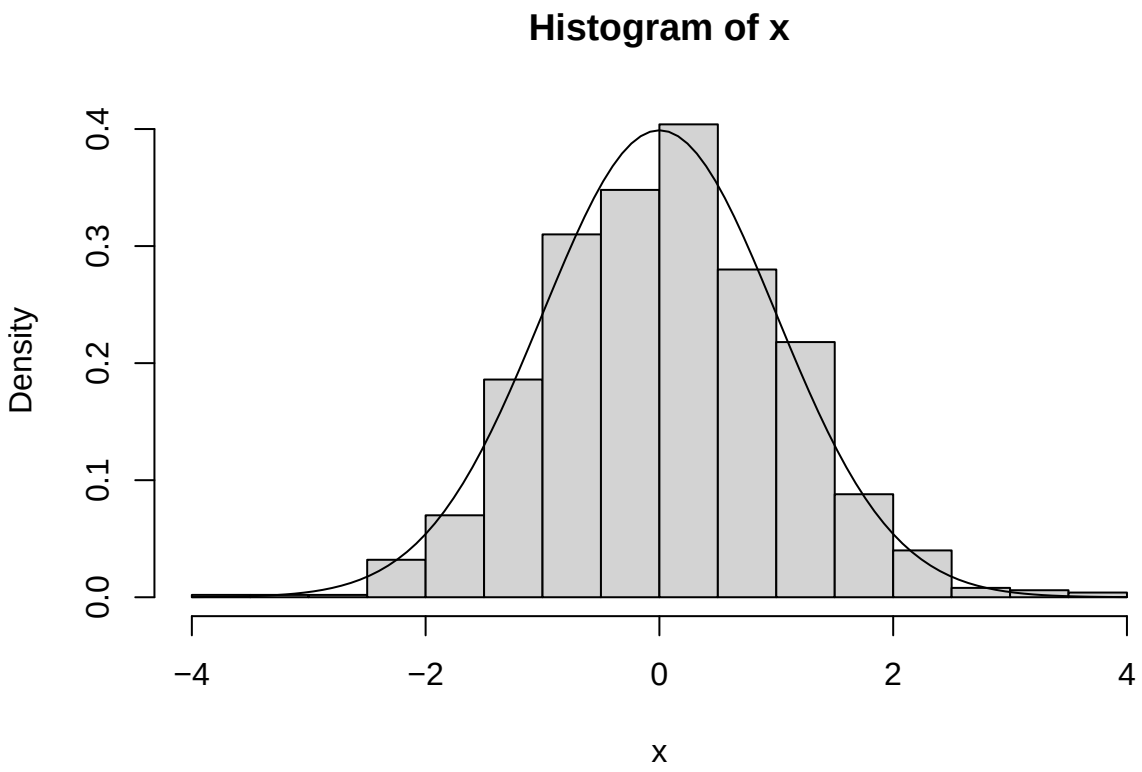



```
# Simulate 1000 random values from Weibull(shape=1.2,scale=16)
xw=rweibull(n=1000,shape=1.5, scale=80)
hist(xw,prob=TRUE,ylim=c(0,.01))
curve(dweibull(x,shape=1.5,scale=80),add=TRUE)
```



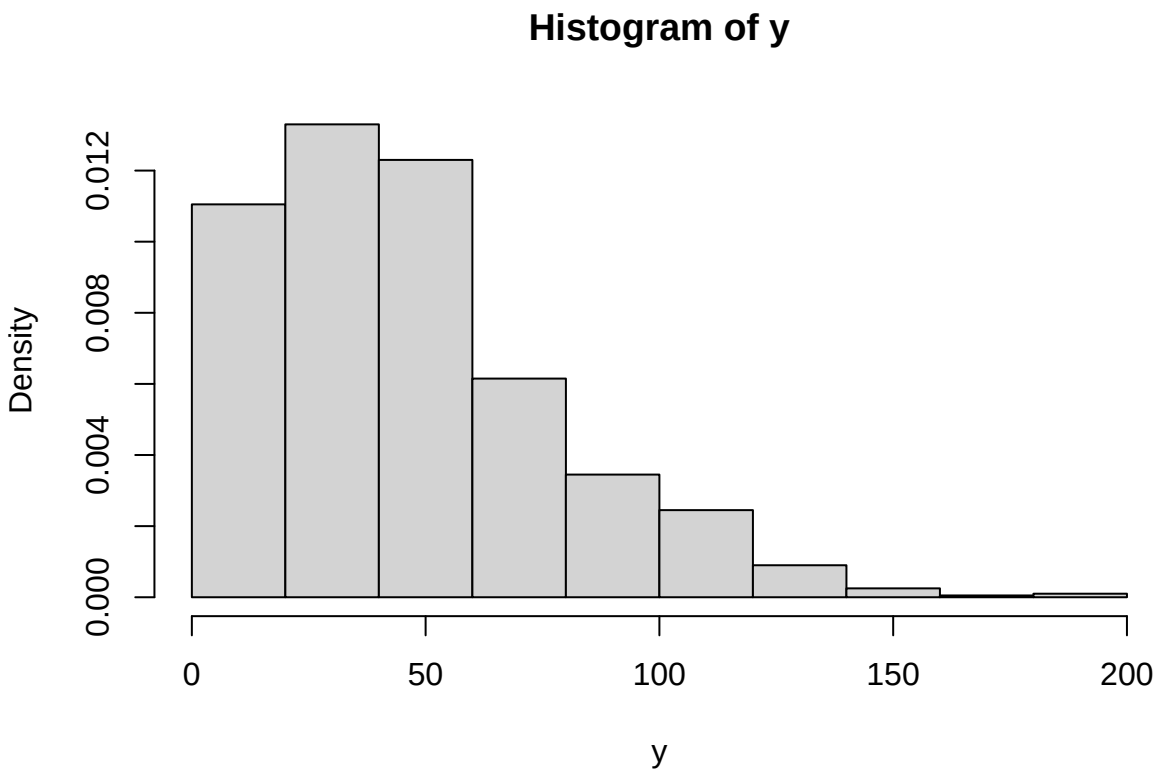
Details of histogram

```
x=rnorm(1000)
hist(x,prob=TRUE)
curve(dnorm(x),add=TRUE)
```

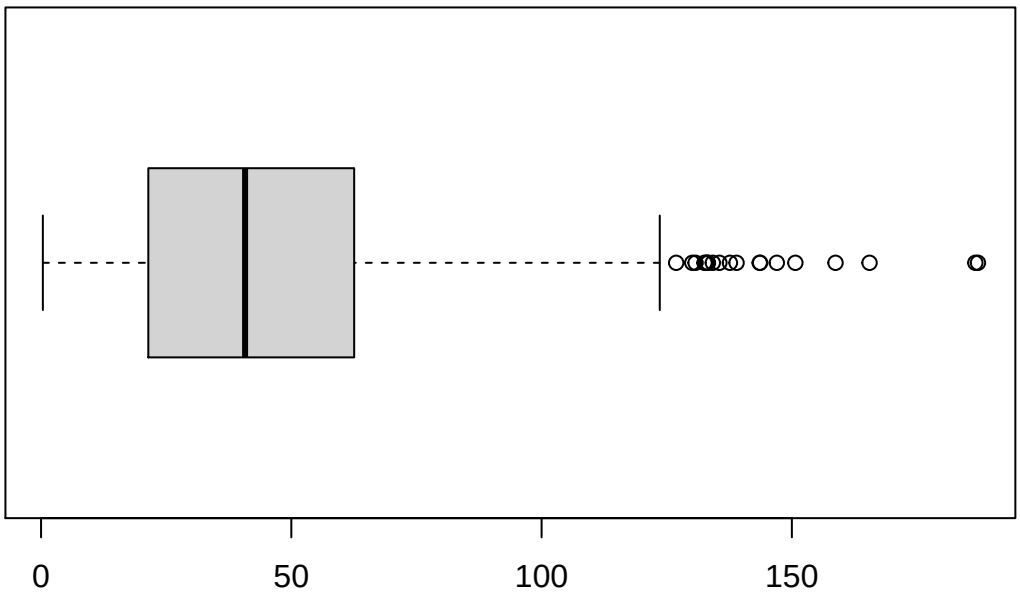


Box plot—boxplot()

```
y=rweibull(1000,shape=1.5,scale=50)
hist(y,prob=TRUE)
```




```
boxplot(y,horizontal=TRUE)
```




```
median(y)
```

```
## [1] 40.76161
```

```
quantile(y,prob=c(0.25,.50,.75))
```

```
##      25%      50%      75%
## 21.43287 40.76161 62.50444
```

```
min(y)
```

```
## [1] 0.3527543
```

```
max(y)
```

```
## [1] 187.1773
```

```
IQR(y)
```

```
## [1] 41.07156
```

```
LowerWhisker=max(min(y),20.449-1.5*IQR(y))
LowerWhisker
```

```
## [1] 0.3527543
```

```
UpperWhisker=min(max(y),60.734+1.5*IQR(y))
UpperWhisker
```

```
## [1] 122.3413
```

Box plot of 'Iris' data

```
names(iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

```
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

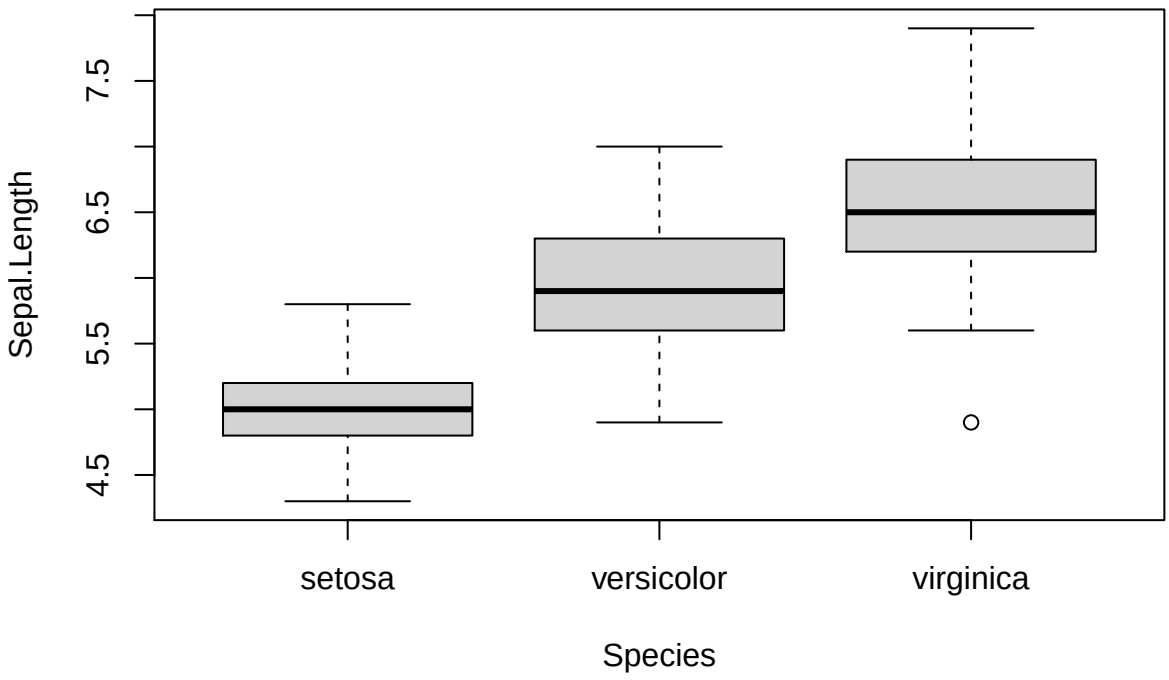
```
tail(iris)
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 145          6.7         3.3          5.7         2.5 virginica
## 146          6.7         3.0          5.2         2.3 virginica
## 147          6.3         2.5          5.0         1.9 virginica
## 148          6.5         3.0          5.2         2.0 virginica
## 149          6.2         3.4          5.4         2.3 virginica
## 150          5.9         3.0          5.1         1.8 virginica
```

```
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
boxplot(Sepal.Length~Species,data=iris)
```



Scatter plot—'plot()'

```
x= rnorm(100) # assigns 100 random normal observations to x
y=rpois(100,30)
mean(y)
```

```
## [1] 29.68
```

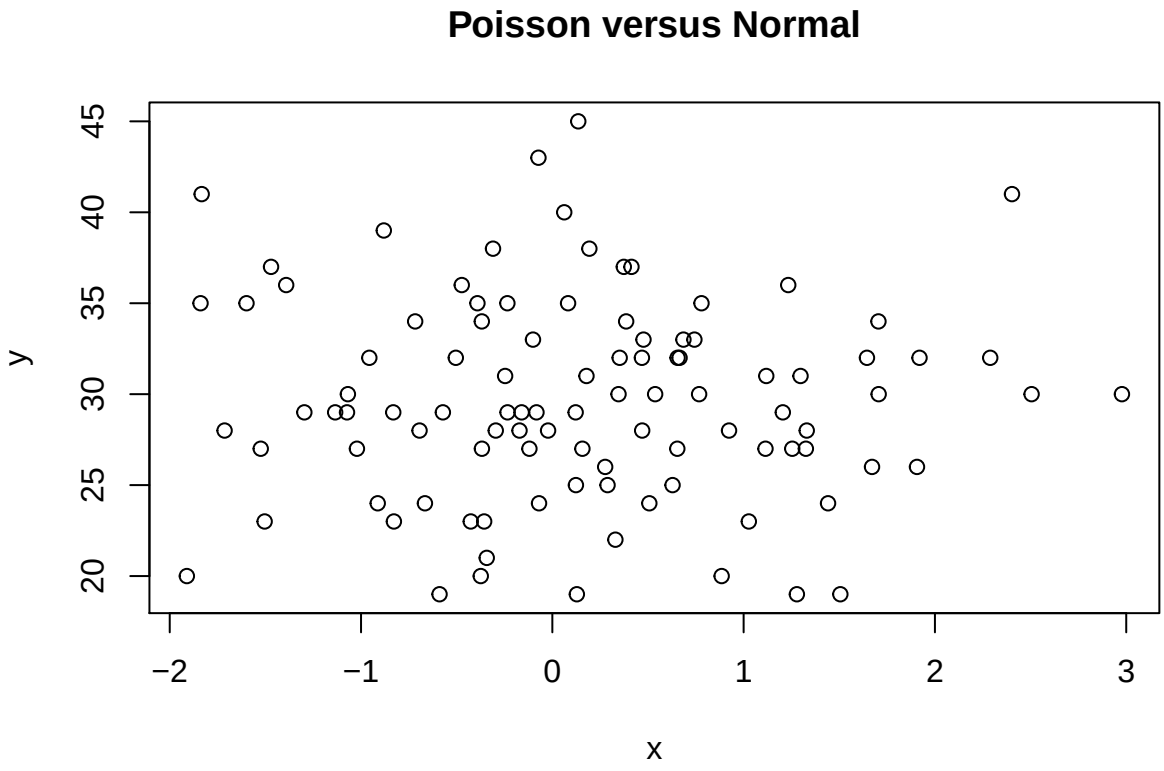
```
mean(x)
```

```
## [1] 0.1589246
```

```
sd(x)
```

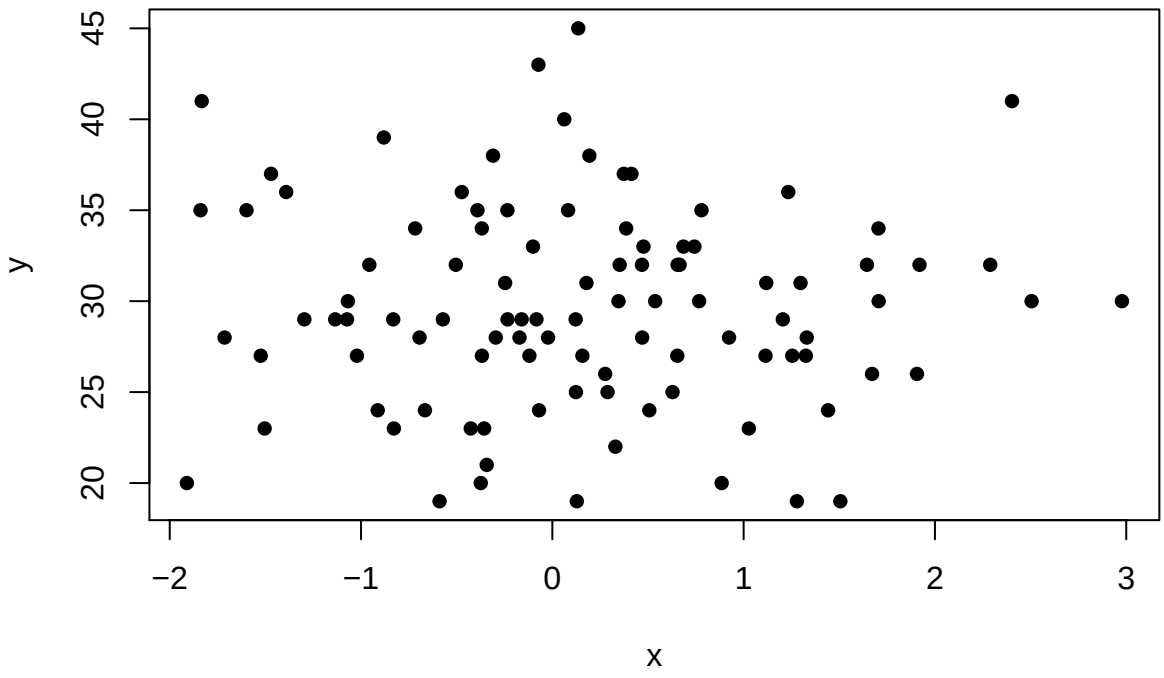
```
## [1] 1.045671
```

```
plot(x,y,main="Poisson versus Normal")
```

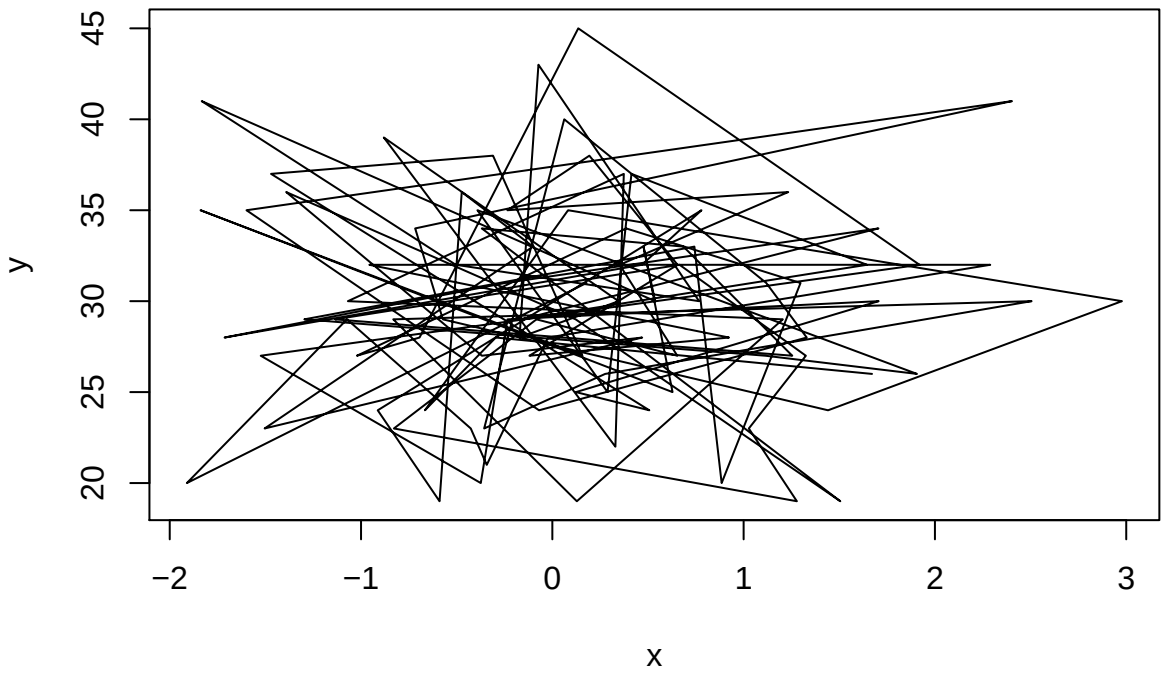



```
plot(x,y,pch=16,type="points")
```

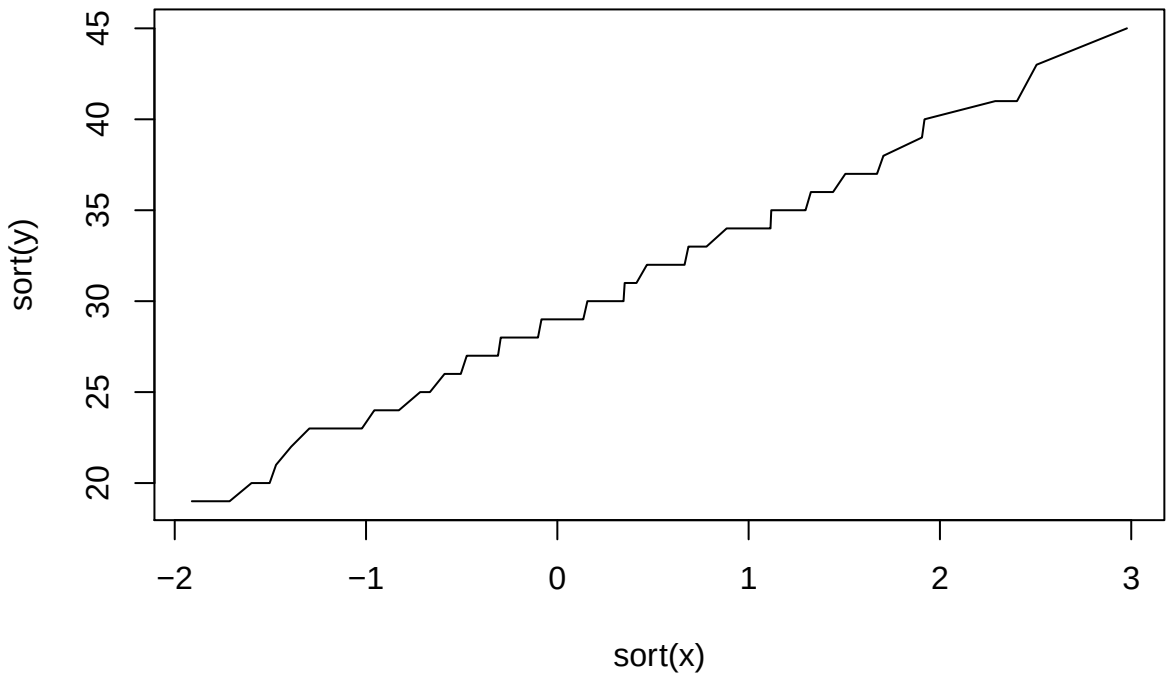
```
## Warning in plot.xy(xy, type, ...): plot type 'points' will be truncated to first
## character
```




```
plot(x,y,type="l") #lines
```




```
plot(sort(x),sort(y),type="l")
```



Plotting From a data frame

```
data(Orange)
names(Orange)
```

```
## [1] "Tree"          "age"           "circumference"
```

```
head(Orange)
```

```
##   Tree  age circumference
## 1    1  118            30
## 2    1  484            58
## 3    1  664            87
## 4    1 1004           115
## 5    1 1231           120
## 6    1 1372           142
```

```
tail(Orange)
```

```
##    Tree  age circumference
## 30    5  484            49
## 31    5  664            81
## 32    5 1004           125
## 33    5 1231           142
## 34    5 1372           174
## 35    5 1582           177
```

```
str(Orange)
```

```
## Classes 'nfnGroupedData', 'nfGroupedData', 'groupedData' and 'data.frame':   35 obs. of  3 variables
##  $ Tree         : Ord.factor w/ 5 levels "3"<"1"<"5"<"2"<..: 2 2 2 2 2 2 2 4 4 4 ...
##  $ age          : num  118 484 664 1004 1231 ...
##  $ circumference: num  30 58 87 115 120 142 145 33 69 111 ...
##  - attr(*, "formula")=Class 'formula'  language circumference ~ age | Tree
##   .. ..- attr(*, ".Environment")=<environment: R_EmptyEnv>
##  - attr(*, "labels")=List of 2
##   ..$ x: chr "Time since December 31, 1968"
##   ..$ y: chr "Trunk circumference"
##  - attr(*, "units")=List of 2
##   ..$ x: chr "(days)"
##   ..$ y: chr "(mm)"
```

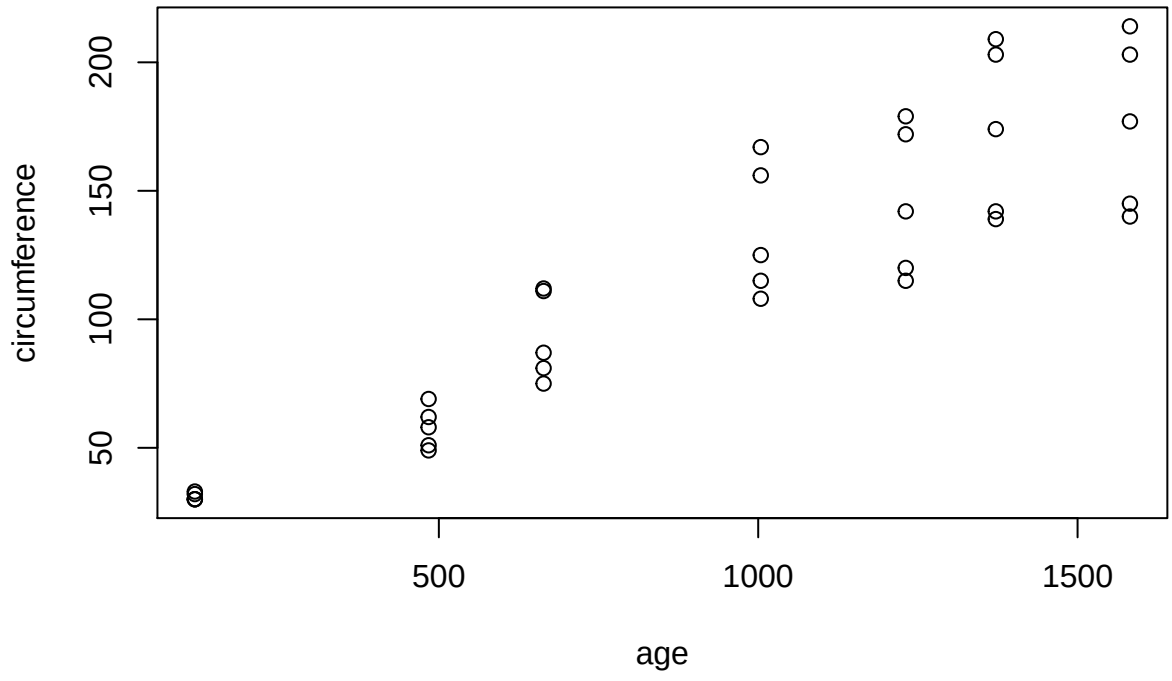
```
Orange$Tree #extract column of Tree
```

```
##  [1] 1 1 1 1 1 1 1 2 2 2 2 2 2 2 3 3 3 3 3 3 3 4 4 4 4 4 4 4 5 5 5 5 5 5 5
## Levels: 3 < 1 < 5 < 2 < 4
```

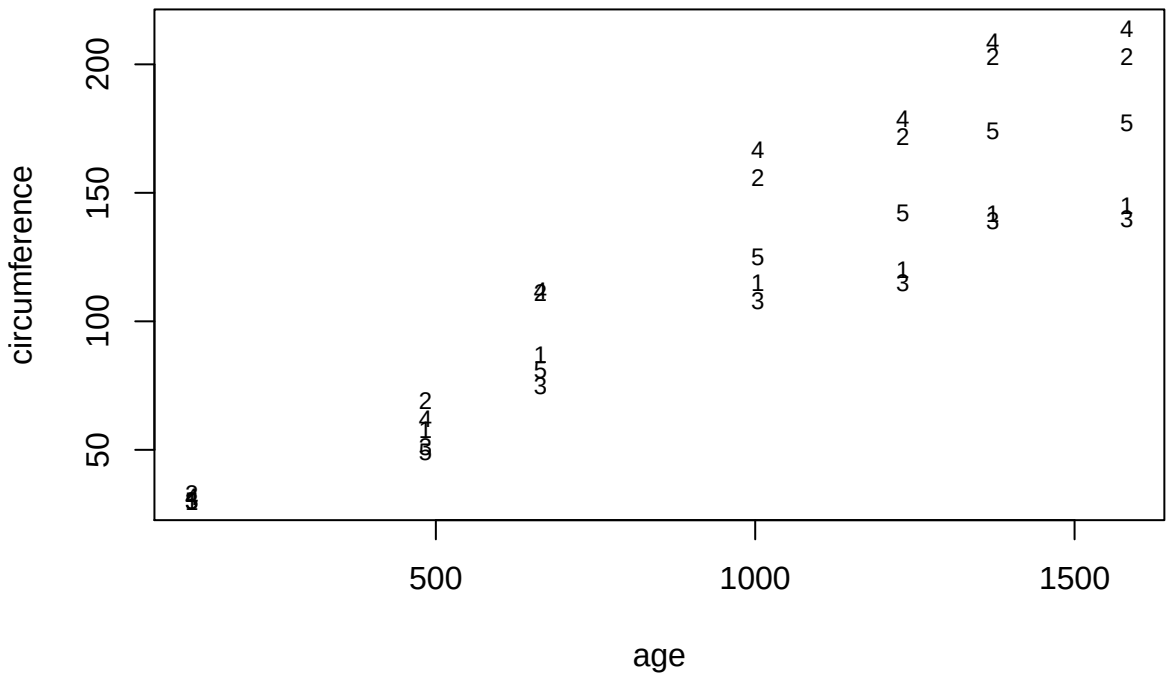
```
unique(as.character(Orange$Tree))
```

```
## [1] "1" "2" "3" "4" "5"
```

```
plot(circumference~age, data = Orange)
```




```
plot(circumference~age, data = Orange, pch = as.character(Tree),
cex = 0.75)
```

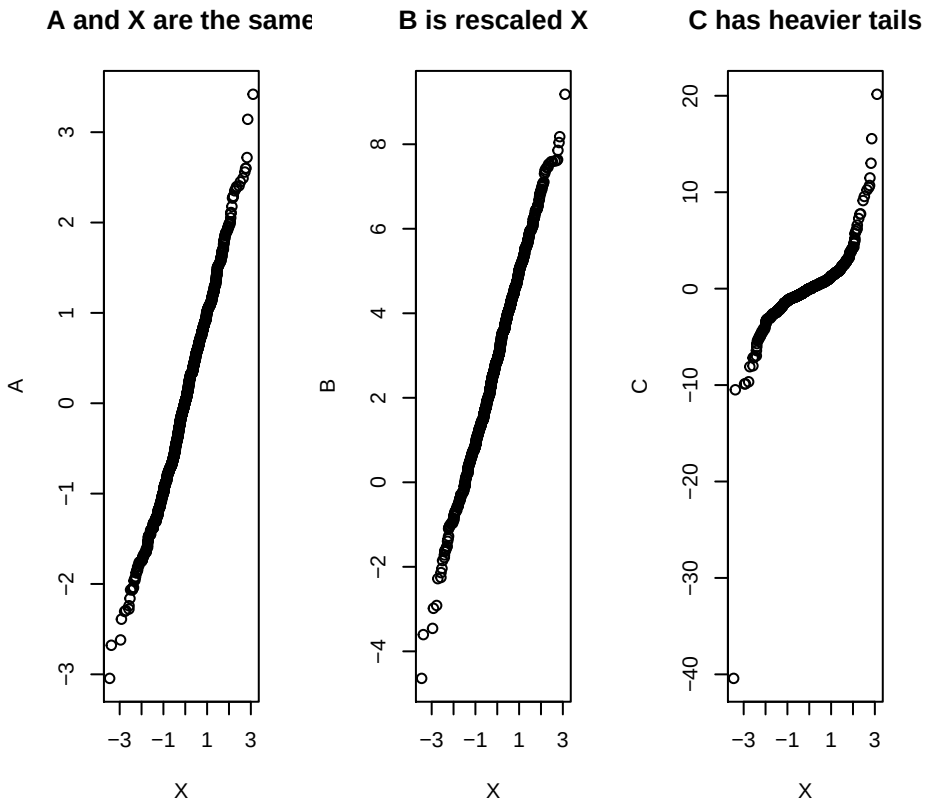


QQ plot

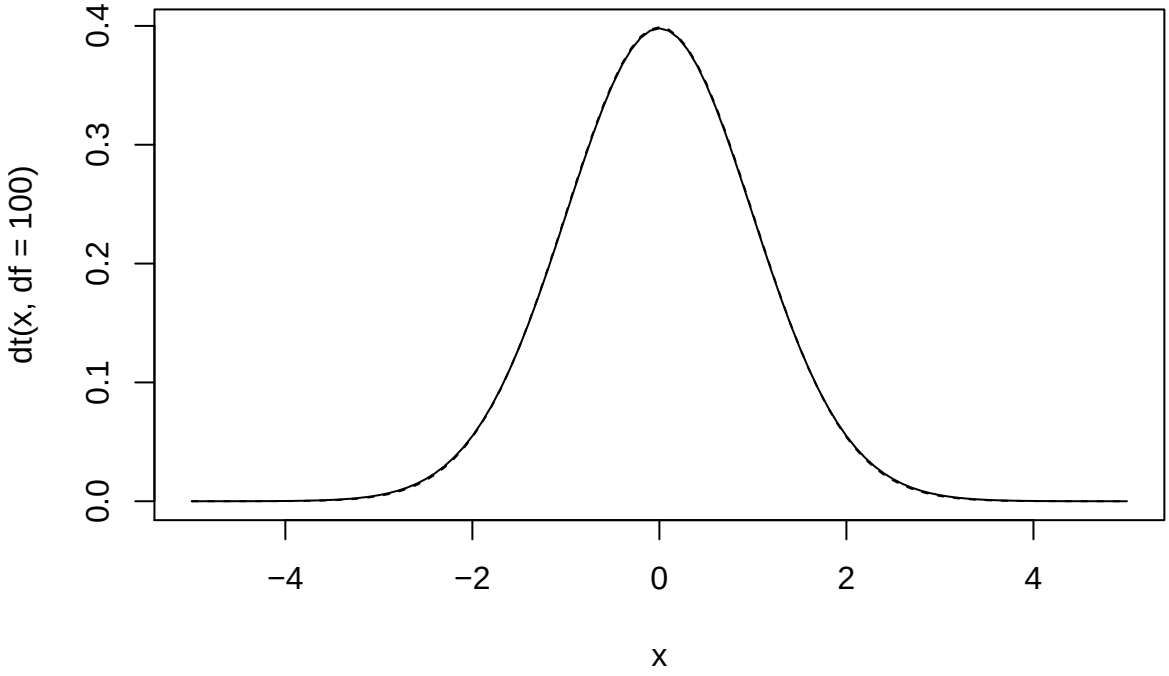
```
z<-rnorm(10000,mean=0,sd=1)
quantile(z,prob=c(0.25,.50,.75))
```

```
##          25%          50%          75%
## -0.683965522  0.008889652  0.676895665
```

```
par(mfrow = c(1,4))
X=rnorm(1000)
A=rnorm(1000)
qqplot(X,A,main="A and X are the same")
B=rnorm(1000,mean=3,sd=2)
qqplot(X,B,main="B is rescaled X")
C=rt(1000,df=2)
qqplot(X,C,main="C has heavier tails")
par(mfrow=c(1,1))
```




```
curve(dt(x,df=100),from=-5,to=5)
curve(dnorm(x),add=TRUE,lty=2)
```



Low level graphic functions

Adding to plots

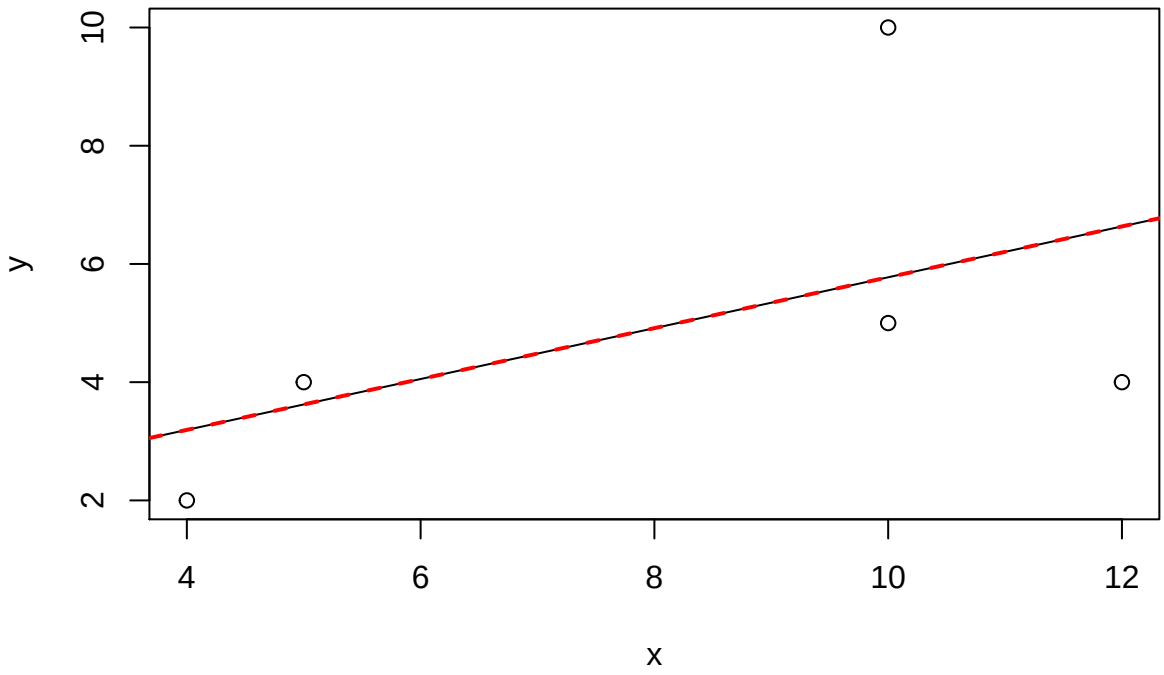
- points(x, y, . . .) # adds point
- lines(x, y, . . .) # adds line segments
- text(x, y, labels, . . .) # adds text into the graph
- abline(a, b, . . .) # adds the line $y = a + bx$
- abline(h = y, . . .) # adds a horizontal line
- abline(v = x, . . .) # adds a vertical line
- polygon(x, y, . . .) # adds a closed and possibly filled polygon
- segments(x0, y0, x1, y1, . . .) # draws line segments
- arrows(x0, y0, x1, y1, . . .) # draws arrows
- symbols(x, y, . . .) # draws circles, squares, thermometers, etc. legend(x, y, legend, . . .) # draws a *legend

Linear Models— ‘lm()’

```
y=c(2,4,5,4,10)
x=c(4,5,10,12,10)
plot(y~x)
# y=beta0+beta1*x+error
M1=lm(y~x)
coef(M1)
```

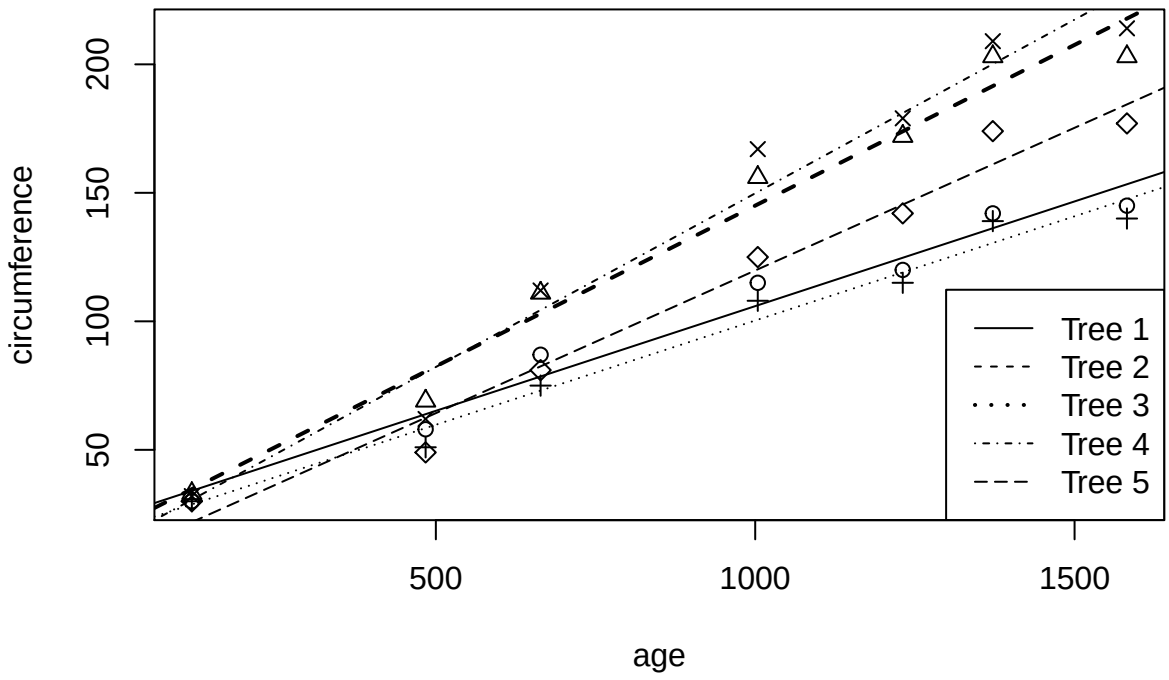
```
## (Intercept)           x
##   1.4713115   0.4303279
```

```
abline(lm(y~x))
abline(M1,lwd=2,lty=2,col="red")
```

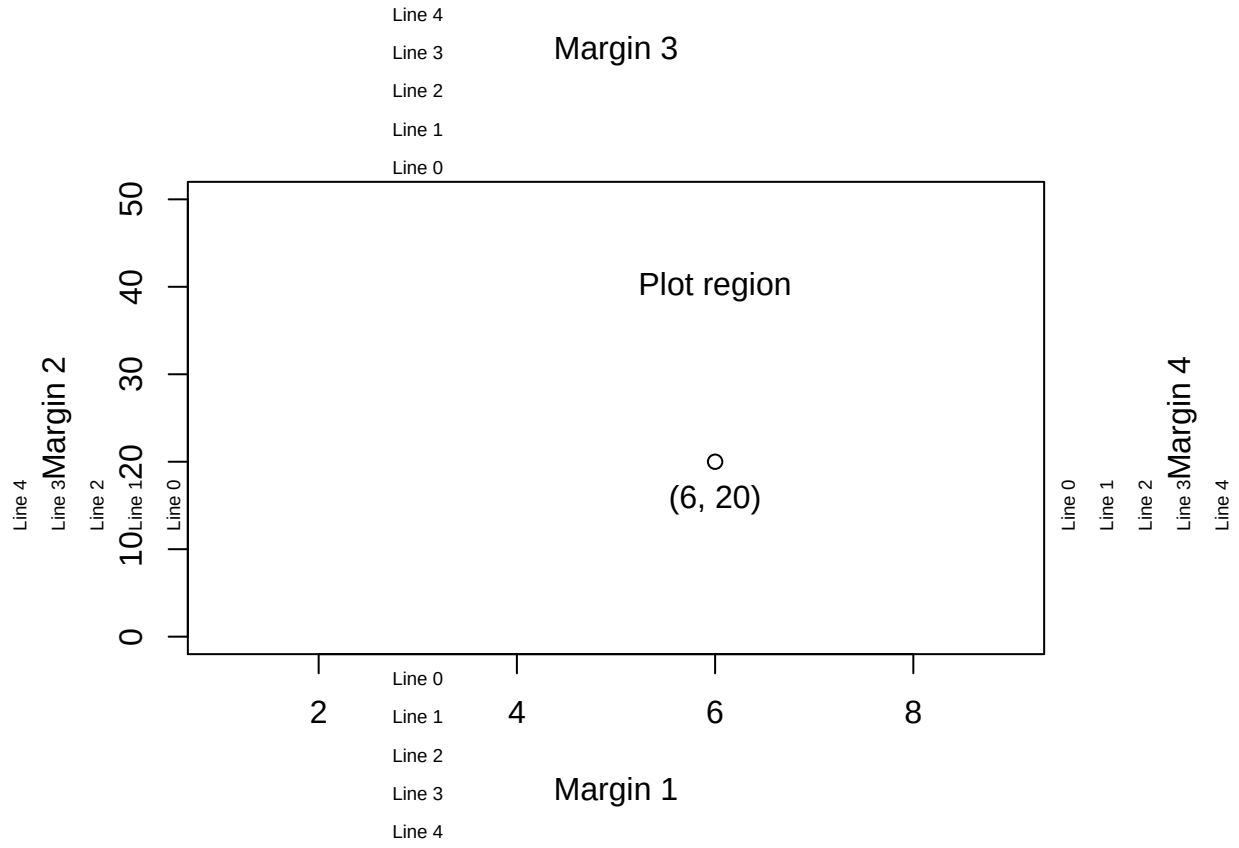


Example Orange data

```
plot(circumference~age, pch=as.numeric(as.character(Tree)),data=Orange)
abline(lm(circumference~age,data=Orange,subset=Tree=="1"))
abline(lm(circumference~age,data=Orange,subset=Tree=="2"),lty=2,lwd=2)
abline(lm(circumference~age,data=Orange,subset=Tree=="3"),lty=3)
abline(lm(circumference~age,data=Orange,subset=Tree=="4"),lty=4)
abline(lm(circumference~age,data=Orange,subset=Tree=="5"),lty=5)
legend("bottomright",legend = paste("Tree",1:5),lty=1:5,lwd=c(1,1,2,1,1))
```

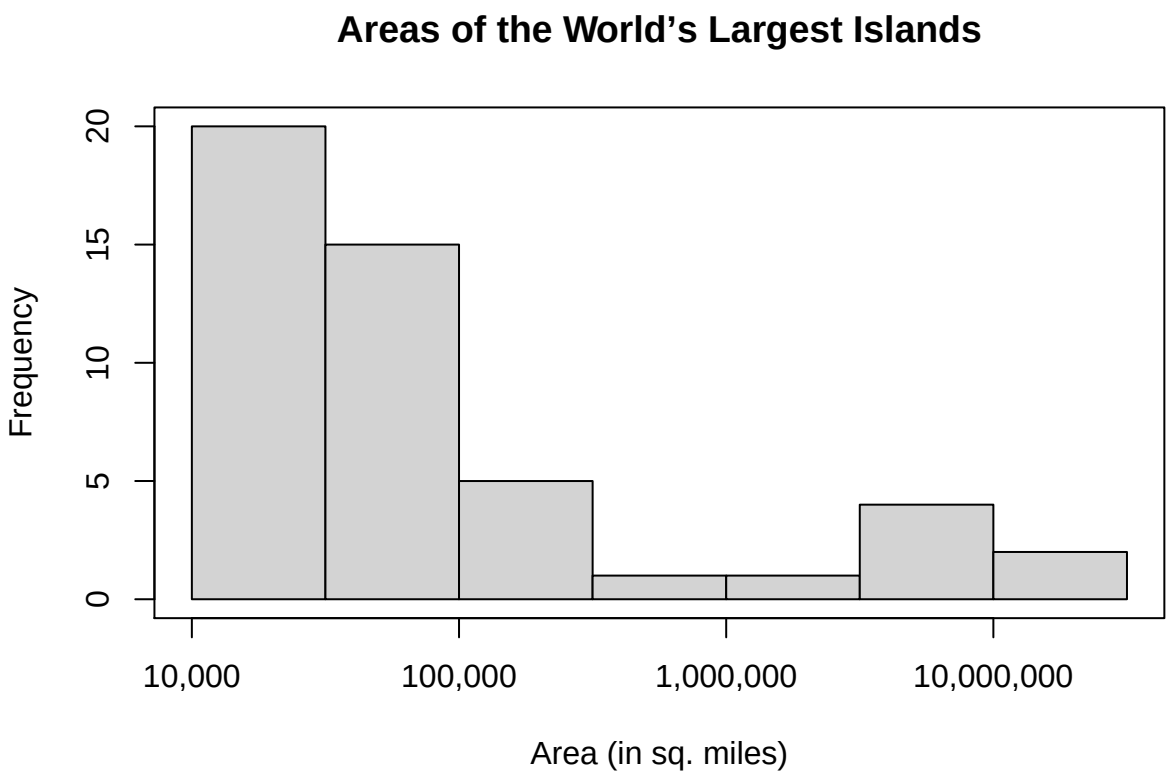



```
par(mar=c(5,5,5,5)+0.1)
plot(c(1,9),c(0,50),type='n' ,xlab="", ylab="")
text(6, 40, "Plot region")
points(6, 20)
text(6, 20, "(6, 20)", adj = c(0.5, 2))
mtext(paste("Margin", 1:4), side = 1:4, line = 3)
mtext(paste("Line", 0:4), side = 1, line = 0:4, at = 3, cex = 0.6)
mtext(paste("Line", 0:4), side = 2, line = 0:4, at = 15, cex = 0.6)
mtext(paste("Line", 0:4), side = 3, line = 0:4, at = 3, cex = 0.6)
mtext(paste("Line", 0:4), side = 4, line = 0:4, at = 15, cex = 0.6)
```



Axis and Tricks

```
hist(log(1000*islands, 10), axes = FALSE, xlab = "Area (in sq. miles)",
main = "Areas of the World's Largest Islands")
box()
axis(side = 1, at = 4:7, labels = c("10,000", "100,000", "1,000,000",
"10,000,000"))
axis(side = 2)
```



Boxplot with some additional features

motor ## Brand 1 Brand 2 Brand 3 Brand 4 Brand 5 ## 1 13.1 16.3 13.7 15.7 13.5 ## 2 15.0 15.7 13.9 13.7 13.4 ## 3 14.0 17.2 12.4 14.4 13.2 ## 4 14.4 14.9 13.8 16.0 12.7 ## 5 14.0 14.4 14.9 13.9 13.4 ## 6 11.6 17.2 13.3 14.7 12.3