Learning R

To **install** R:

```
http://www.cran.r-project.org
```

To get R **manuals**:

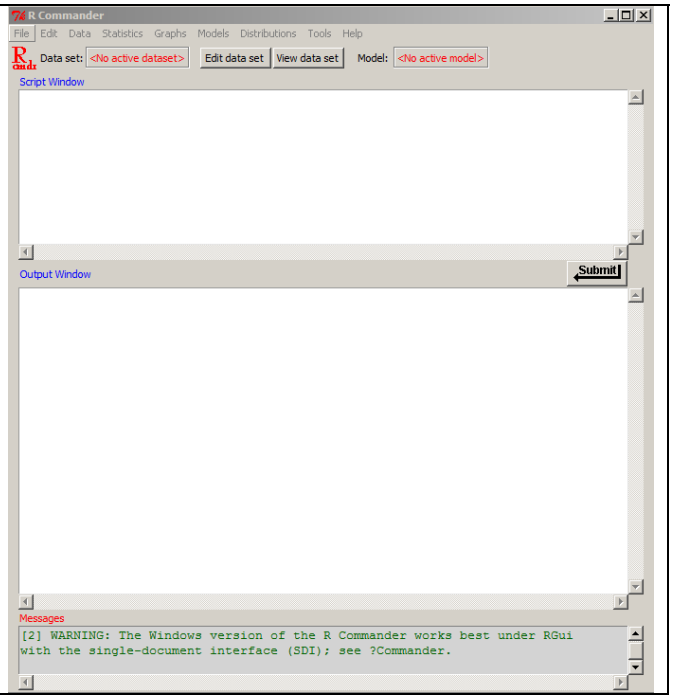
```
http://www.cran.r-project.org/manuals.html
```

To change **prompt**:

```
> options(prompt="R > ")
R >
```

To get **graphical interface**:

```
> library(Rcmdr)
```



To read in **csv** type data file (rownames are on the first column):

```
> data1 <- read.csv("Forbes2000.csv")
```

or

```
> data1 <- read.table("Forbes2000.csv", header=TRUE, sep=",", row.names=1)
```

To read in **txt** type data file:

```
> data1 <- read.table("Forbes2000.txt", header=TRUE)
```

To read in **csv** type data file and **assign** classes to variables:

```
> data1 <- read.table("Forbes2000.txt", header=TRUE, sep=",",
+ row.names=1,colClasses=c("character","integer","character","factor",
+ "factor","numeric","numeric","numeric","numeric"))
```

Simple calculation:

+, -, *, / (add, subtract, multiply, divide), ^ (power), %/% (integer quotient), %% (remainder)

```
> 3+5; 4/6; 4^2; 17%%6; 17%/%6
[1] 8
[1] 0.6666667
[1] 16
[1] 5
[1] 2
```

* sqrt (= $\sqrt{\ }$), abs, exp (=2.71828...), log (= $\log_n$), log10 (= $\log_{10}$), log2 (= $\log_2$), sin, cos, tan, asin (= $\sin^{-1}$), acos, atan, sinh, cosh, tanh, asinh (= $\sinh^{-1}$), acosh, atanh, gamma (= $(n-1)!$), lgamma (=log(gamma(x)), ceiling (=smallest integer $\geq x$), floor(=largest integer $\leq x$), trunc (=integer only), round (=rounding)

```
> log2(16); round(sin(pi), 2); asin(1);
[1] 4
[1] 0
[1] 1.570796
> pi / 2
[1] 1.570796
> gamma(4); ceiling(6.789); floor(6.789); trunc(6.789); round(6.789,2)
[1] 6
[1] 7
[1] 6
[1] 6
[1] 6.79
```

Insertion (i.e., assignment):

<-, ->, =

```
> x <- 1:4
> x
[1] 1 2 3 4
> 1:5 -> y
> y
[1] 1 2 3 4 5
> z = 1:5
> z
[1] 1 2 3 4 5
> (z <- 1:5)   #insert and "display"
[1] 1 2 3 4 5

> x <- 0.8-4
> ( y <- c(1,2,3,4,5) )
[1] 1 2 3 4 5
> ( z <- c(y, x) )
[1]  1.0  2.0  3.0  4.0  5.0 -3.2
```

objects:

```
> objects()
[1] "data1"    "TREATMNT"
> rm(data1)
> rm(list=ls(all=TRUE))  #to remove all objects
```

* This is same as using “List objects” from “drop down” menu “MISC.”

Parameters: these are used inside options:

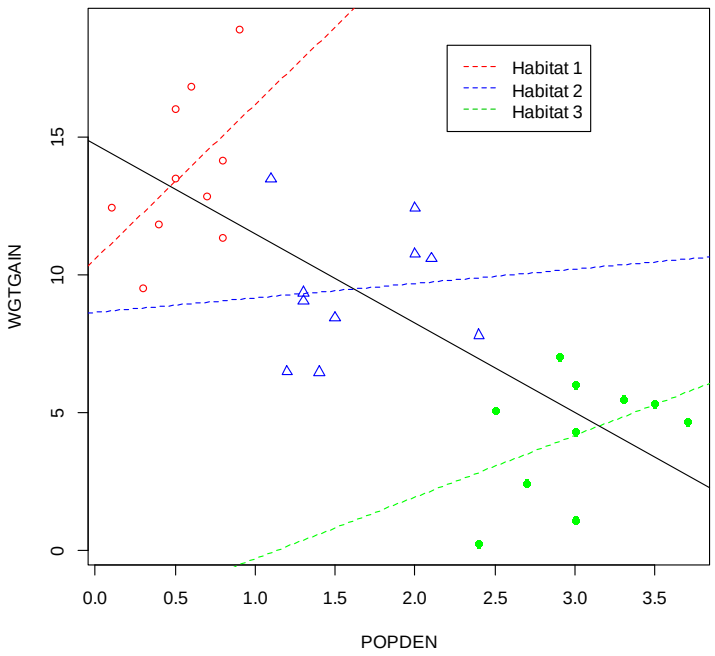
Parameters	What do they do?
continue	assigns “prompt” (default = +)
device	assigns device ('x11', 'windows', 'gtk')
digits	number of digits to display (default = 7)
expressions	number of lines to evaluate (default = 500)
prompt	assigns prompt (default = >)
scipen	to display in exponential form or not (default = 0)
show.error.messages	to display error message or not (default = TRUE)
timeout	time limit to connect to the internet (default = 60 seconds)
warn	how to handle “warnings” (default = 0). All warnings are ignored if you enter “negative value.” “1” will keep the warning alive and print. Numbers greater than 1 will convert all warning to errors.
width	numbers of characters per line (default = 80)
CRAN	assigns the URL of CRAN (default = http://cran.r-project.org)

```
> 2/9
[1] 0.2222222
> options(digits=10)
> 2/9
[1] 0.2222222222
> options(scipen=0);  print(1e5)
[1] 1e+05
> options(scipen=1);  print(1e5)
[1] 100000
```

```
> x <- runif(6)
> x
[1] 0.6833957110 0.3477469841 0.9696522753 0.2036655385 0.9043799203 0.4716767156
> options(width=50)
> x
[1] 0.6833957110 0.3477469841 0.9696522753
[4] 0.2036655385 0.9043799203 0.4716767156
> options(width=10)
> x
[1] 0.6833957110
[2] 0.3477469841
[3] 0.9696522753
[4] 0.2036655385
[5] 0.9043799203
[6] 0.4716767156
```

Plots with a symbols and regression lines overlaid:

```
> data1 <- read.csv("D:\\Stat333\\data1.csv")
> dim(data1)
[1] 30  3
> head(data1)
  HABITAT POPDEN WGTGAIN
1       1    0.1   12.41
2       1    0.3    9.48
3       1    0.4   11.83
4       1    0.5   15.98
5       1    0.5   13.48
6       1    0.6   16.81
> attach(data1)
> sPOPDEN <- split(POPDEN, HABITAT)
> sWGTGAIN <- split(WGTGAIN, HABITAT)
> plot(POPDEN, WGTGAIN, type="n")
> points(sPOPDEN[[1]], sWGTGAIN[[1]], pch=1,col="red")
> points(sPOPDEN[[2]], sWGTGAIN[[2]], pch=2,col="blue")
> points(sPOPDEN[[3]], sWGTGAIN[[3]], pch=16,col="green")
> abline(lm(WGTGAIN[HABITAT==1]~POPDEN[HABITAT==1]), lty=2, col="red")
> abline(lm(WGTGAIN[HABITAT==2]~POPDEN[HABITAT==2]), lty=2, col="blue")
> abline(lm(WGTGAIN[HABITAT==3]~POPDEN[HABITAT==3]), lty=2, col="green")
> abline(lm(WGTGAIN~POPDEN))
> legend(locator(1),c("Habitat 1","Habitat 2","Habitat 3"),lty=c(2,2,2),col=c(2,4,3))
```



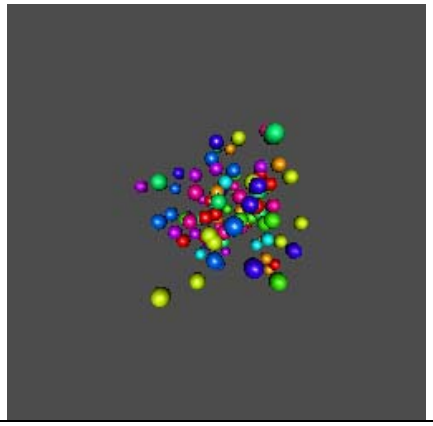
Packages:

Package names	Description
base	R's basic package
boot	Package about bootstrap
class	Package about classification
cluster	Package about cluster analysis
Foreign	Reads data files created by other stat packages (like SAS)
grid	Grid graphics package
KernSmooth	Package for density estimation by kernel function
lattice	Lattice graphics Package
MASS	Package and datasets that are shown in "Modern Applied Statistics with S"
methods	Methods and Classes defined as R's objects. Programming tool package
mgcv	Package for multivariate smoothing parameter estimation (GCV,GAM)
nlme	Package for linear mixed effects model and nonlinear mixed effects model
nnet	Package for Feed, forward, neural net and polynomial loglinear model
rpart	Package about classification by recursive algorithm and regression trees
spatial	Package for Krieging and point pattern interpretation
splines	Spline regression package
stats	R's statistics package
stats4	S4-class statistical functions
survival	Survival time analysis package including penalized likelihood
tcltk	Interface package for Tcl/Tk
tools	Package for development and maintenance of packages
utils	R's utility package

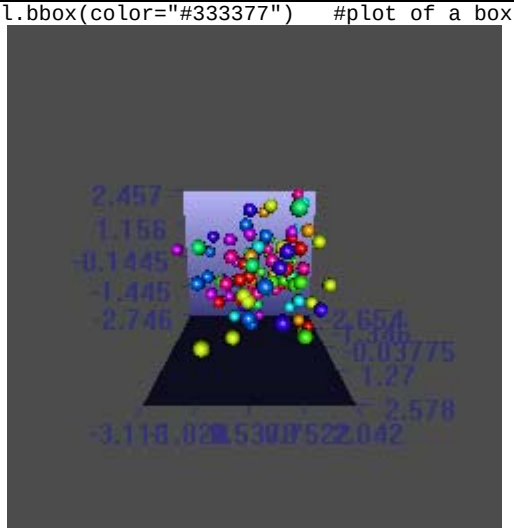
* In addition to the standard packages, there are over 700 packages.

```
> library(help="stats")     #for detailed explanation about a package
> help(package="stats")
> library()                 #shows available packages
> search()                  #shows list of packages that are currently read
```

```
> library(rgl)              #calling a library.
You may have to > install.packages("rgl") first
> example(rgl.surface)      #just to show off

> n <- 100
> rgl.clear()
> rgl.bg(color=c("white", "black")) #background
color
> rgl.spheres(rnorm(n), rnorm(n), rnorm(n),
radius=0.2, color=rainbow(10))  #plot of a
sphere
```



```
> rgl.bbox(color="#333377")   #plot of a box
```



```
> detach("package:rgl")     #to unload a
package
```

Data vector 1:

```
> x <- c(1, 2, 3, 4, 5)
> x
[1] 1 2 3 4 5
> length(x)
[1] 5
> 1:5
[1] 1 2 3 4 5
> 3:-3
[1]  3  2  1  0 -1 -2 -3
> rep(2:5, times = 3)
[1] 2 3 4 5 2 3 4 5 2 3 4 5
> rep(1:3, length=5)
[1] 1 2 3 1 2
> seq(1,10,length=6)
[1]  1.0  2.8  4.6  6.4  8.2 10.0
```

Command	Function
`a:b`	From a to b (with increment 1)
`seq(a, b, length = n)`	From a to b with n equal intervals
`seq(a, b, by = c)`	From a to b (with increment c)
`rep(a:b, times = c)`	From a to b, "c" times
`rep(a:b, length = c)`	From a to b, of length "c"
`unique(x)`	Returns a vector after removing replicates
`numeric(n)`	Creates a vector of 0's of length "n"

Data vector 2:

```
> x <- c(1, 2, 3, 4, 5)
> x
[1] 1 2 3 4 5
> c(x, 10)
[1]  1  2  3  4  5 10
> x[2] <- 99
> x
[1]  1 99  3  4  5

> c(1, 2, 3) + c(4, 5, 6)
[1] 5 7 9
> 1 / (2:5)
[1] 0.5000000 0.3333333 0.2500000 0.2000000
> sqrt(1:5)
[1] 1.000000 1.414214 1.732051 2.000000 2.236068

> x <- 1:5
> rev(x)
[1] 5 4 3 2 1
> y <- cumsum(x)
> y
[1]  1  3  6 10 15
> mean(y, 0.5)        #trimmed mean after removing 50% of each end
[1] 6
> mean(y, 0.2)        #trimmed mean after removing 20% of each end
[1] 6.333333
> y <- c(0, 1, 3, 6, 10);
> sort(y)
[1]  0  1  3  6 10
> order(y)
[1] 1 2 3 4 5
> rank(y)
[1] 1 2 3 4 5
> k <- c(10, 20, 20, 40, 50)
> order(k)
[1] 1 2 3 4 5
> rank(k)
[1] 1.0 2.5 2.5 4.0 5.0
> z <- c(100, 20, 4, 2, 1)
> pmax(x, y, z)
[1] 100  20   4   6  10
> pmin(x, y, z)
[1] 0 1 3 2 1
```

```
> x <- c(1, 2, 3)
> y <- c(10, 11, 12)
> prod(x,y)
[1] 7920
> prod(x)
[1] 6
> prod(y)
[1] 1320
```

`sum()`, `mean()`, `var()`, `median()`, `cor()`, `max()`, `min()`, `prod()`, `cumsum()`, `sd()`, `sort()`, `rev()`(=reverse sort), `pmax()`, `pmin()`(=max & min for each place), `range()`, `match()`, `diff()`, `rank()`, `order()`(=original place of elements)

Data vector 3:

```
> x <- 1:5;  y <- 1:4
> intersect(x, y)
[1] 1 2 3 4
> union(x, y)
[1] 1 2 3 4 5
> setequal(x, y)
[1] FALSE
> is.element(x, y)
[1]  TRUE  TRUE  TRUE  TRUE FALSE
```

Commands	Meaning
`union(x, y)`	Union of two sets
`intersect(x, y)`	Intersection of two sets
`setdiff(x, y)`	Difference of two sets
`setequal(x, y)`	Are the two sets the same?
`is.element(x, y)`	Are the elements of "x" also elements of "y"?

Matrix:

```
> x <- c(1, 2, 3, 4, 5, 6)
> A <- matrix(x, nrow=2,ncol=3)
> A
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> A[1,]
[1] 1 3 5
> A[,1]
[1] 1 2
> A[1,2]
[1] 3
> A[c(1,2), 3]
[1] 5 6
> A[c(1,2), c(2,3)]
     [,1] [,2]
[1,]    3    5
[2,]    4    6
> x <- c(1,2,3,4)
> A <- matrix(x, 2, 2)
> A
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> A*A
     [,1] [,2]
[1,]    1    9
[2,]    4   16
> A%*%A
     [,1] [,2]
[1,]    7   15
[2,]   10   22
> t(A)
     [,1] [,2]
[1,]    1    2
[2,]    3    4
> eigen(A)
$values
```

```
[1]  5.3722813 -0.3722813

$vectors
           [,1]       [,2]
[1,] -0.5657675 -0.9093767
[2,] -0.8245648  0.4159736

> det(A)
[1] -2
> solve(A)
     [,1] [,2]
[1,]   -2  1.5
[2,]    1 -0.5
```

Commands	Meaning
solve(A)	A^{-1}
chol(A)	Cholesky decomposition
svd(A)	Singular-value decomposition of a rectangular matrix
ginv(A)	Generalized inverse
qr(A)	QR decomposition
t(A)	Transpose of A

Solving for a system of equations:

```
> a <- matrix(c(0,1,2,3,4,5,6,7,9), 3,3)       #      3y + 6z =  1
> b <- matrix(c(1,0,-2))                       #  x + 4y + 7z =  0
> solve(a, b)                                  # 2x + 5y + 9z = -2
          [,1]
[1,] -2.333333
[2,]  2.333333
[3,] -1.000000
```

Assign names to rows and columns:

```
> x <- matrix(1:6, nrow=2, ncol=3)
> rownames(x) <- c("up", "down")
> colnames(x) <- c("left", "center", "right")
> x
     left center right
up      1      3     5
down    2      4     6
> x["up","right"]
[1] 5
```

List

```
> x <- list(57, "173", c(1,9,0,4,5,0,7))
> names(x) <- c("weight", "height", "BMI")
> x
$weight
[1] 57

$height
[1] "173"

$BMI
[1] 1 9 0 4 5 0 7

> temp <- "height"
>  x[[temp]]
[1] "173"
> list(sequence=1:5, "letters"="abc")
$sequence
[1] 1 2 3 4 5

$letters
[1] "abc"
```

Functions:

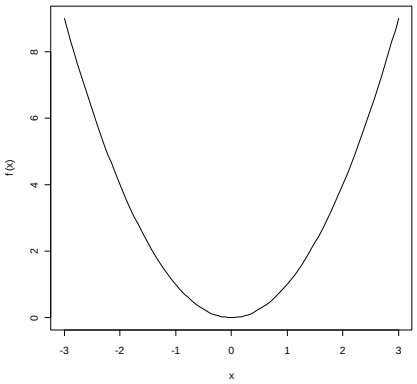
```
> date()
```

```
> myfunc <- function(x, y) {
+ return(x*y)
```

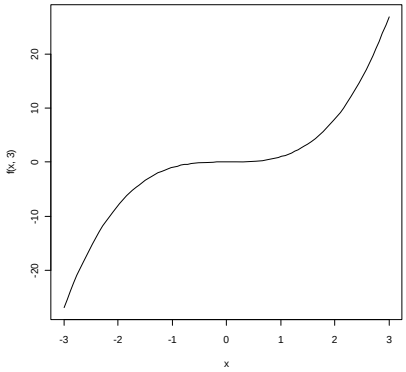
```
> mydouble <- function(x) 2*x
>
> mydouble <- function(x) {
+    return( 2*x )
+ }
> mydouble(2)
[1] 4
```

```
+ }
> myfunc(2,7)
[1] 14

> myprod <- function(n) {
+    if (n==0) 1
+    else      prod(1:n)
+  }
> myprod(0)
[1] 1
> myprod(5)
[1] 120
```

```
> f <- function(x) {
+ return(x^2)
+ }
> curve(f, -3, 3)
```




```
> f <- function(x, a) {
+ return(x^a)
+ }
> curve(f(x, 3), -3, 3)
```



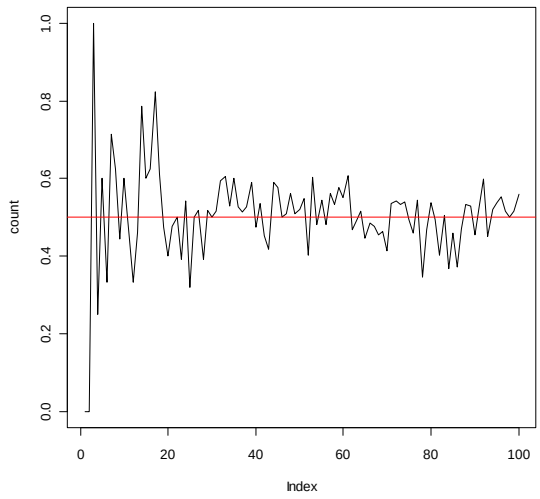

```
> myfunc00 <- function(x) {
+ if (x > 1) return(1)
+ else       return(0)
+ }
> myfunc00(78)
[1] 1
```

```
> myfunc05 <- function() {
+ if (runif(1) > 0.5) return(1)
+ else                return(0)
+ }
> myfunc05()
[1] 1
> myfunc05()
[1] 0
```

```
> myfunc <- function(x) ifelse(x > 0, 1, -1)
> myfunc(1)
[1] 1
> myfunc(-2)
[1] -1
```

```
> x <- 2
>  if (x > 0) {
+    sum(1:x)
+  } else {
+    x <- -x
+    sum(1:x)
+  }
[1] 3
```

```
> mymonte02 <- function(n) {
+ count <- 0
+ for (i in 1:n) {
+ if (runif(1) > 0.5) count <- count+1  #flip a
fair coin
+ }
+ return(count / n)
+ }
> mymonte03 <- function(n) {
+ count <- c()
+ for (i in 1:n) {
+ count <- c(count, mymonte02(i))
+ }
+ plot(count, type="l")
+ }
> mymonte02(100)
[1] 0.48
> mymonte02(100)
[1] 0.52
> mymonte03(100)
> abline(h=0.5, col="red")
```

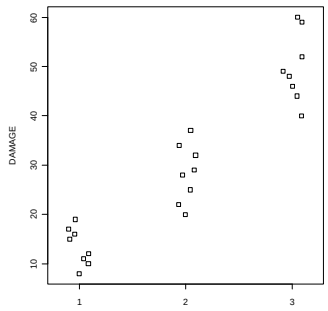



```
> data1 <- read.csv("D:\\Stat333\\hw10-1.csv")
> data1[1:3,]
```

```
>
with(data1,stripchart(DAMAGE~WDKLR,vertical=T,m
```

```
   WDKLR DAMAGE
1      1     12
2      1      8
3      1     15
> stderr <- function(x) sqrt(var(x)/length(x))
> se <- tapply(data1$DAMAGE, data1$WDKLR,
stderr)
> se
       1        2        3
1.349603 2.078268 2.469456
```

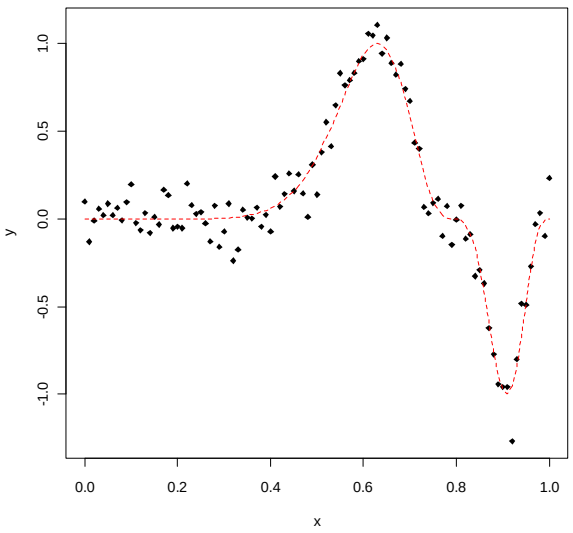
```
ethod="jitter")) #use "stack" to not "jitter"
```




```
> choose(5,2)
[1] 10
> factorial(3)
[1] 6
> round(2.51)
[1] 3
> round(0.5); round(1.5); round(2.5);
round(3.5); round(4.5); round(5.5); round(6.5)
[1] 0
[1] 2
[1] 2
[1] 4
[1] 4
[1] 6
[1] 6
```

```
> round(7.5); round(8.5); round(9.5);
round(10.5); round(11.5)
[1] 8
[1] 8
[1] 10
[1] 10
[1] 12

> r2norm <- function(n, mu, sigma, rho) {
+    tmp <- rnorm(n)
+    x   <- mu+sigma*tmp
+    y   <- rho*x + sqrt(1-rho^2)*rnorm(n)
+    return(data.frame(x=x,y=y))
+  }
>  mydata <- r2norm(100, 0, 1, 0.5)
>  cor(mydata$x,mydata$y)
[1] 0.3077361
>  plot(mydata$x,mydata$y)
```

```
> funk <- function(x) (sin(2*pi*x^3))^3
> x <- seq(0,1,by=0.01)
> y <- funk(x)+(0.1*rnorm(101))
>
matplot(x,cbind(y,funk(x)),type="pl",ylab="y",p
ch=18,lty=2)
```



* gamma function: $\Gamma(t) = \int_0^\infty x^{t-1} e^{-x} dx$, and $\Gamma(1) = 1$, $\Gamma\left(\frac{1}{2}\right) = \sqrt{\pi}$, $\Gamma(t) = (t-1)\Gamma(t-1)$, $t > 1$, $\Gamma(n) = (n-1)!$

`beta(a, b)`	beta function, B(a, b) = (gamma(a) × gamma(b)) / gamma(a+b).
`lbeta(a, b)`	log(beta(a, b))
`gamma(x)`	gamma function
`lgamma(x)`	log(gamma(x))
`digamma(x)`	first derivative of lgamma
`trigamma(x)`	second derivative of lgamma
`tetragamma(x)`	third derivative of lgamma

`pentagamma(x)`	fourth derivative of lgamma
`choose(n, k)`	binomial coefficient
`lchoose(n, k)`	log(choose(n, k))

Loop:

```
> x <- 0
> for (i in 1:5) {
+    x <- x + 1
+ }
> x
[1] 5

> x <- 0
> while (x <= 5) {
+    x <- x + 1
+ }
> x
[1] 6
```

If:

```
> x <- 2
>  if (x > 0) {
+    sum(1:x)
+  } else {
+    x <- -x
+    sum(1:x)
+  }
[1] 3

> x <- c(-2:4)
> sqrt(ifelse(x >= 0, x, NA))
[1]       NA       NA 0.000000 1.000000 1.414214 1.732051 2.000000

> myfactorial <- function(n) {
+   if (n <= 1) return(1)
+   else        return( n * myfactorial(n-1) )
+ }
> myfactorial(5)
[1] 120

> myfactorial <- function(n) {
+   if (n <= 1) return(1)
+   else        return( n * Recall(n-1) )
+ }
> myfactorial(5)
```

Local vs. Global:

```
> x <- 10
> myfunc <- function(y) {
+   x <- y + 999
+   return(x)
+ }
> x
[1] 10
> myfunc(1)
[1] 1000
> x
[1] 10
>
> x <- 10
> myfunc <- function () {
+   x <- 777
+   x <<- 99
+   print(x)
+ }
> myfunc()
[1] 777
> x
[1] 99
```

Numerical methods:

```
> f <- function (x) x^3 - 2                 #solution by Newton method
> uniroot(f, c(0, 2))
$root
[1] 1.259934

$f.root
[1] 6.088618e-05

$iter
[1] 5

$estim.prec
[1] 6.103516e-05

> polyroot(c(5, 4, 1))                      #x^2 + 4x + 5 = 0
[1] -2+1i -2-1i
> round( polyroot(c(1, 2, 1)), digits=3 )   #x^2 + 2x + 1 = 0
[1] -1+0i -1+0i

> f <- expression( a*x^4 )                  #define f
> D(f, "x")                                 #differentiate f wrt x
a * (4 * x^3)

> DD <- function(expr, name, order = 1) {
+   if(order < 1) stop("'order' must be >= 1")
+   if(order == 1) D(expr, name)
+   else DD(D(expr, name), name, order - 1)
+ }
> DD(f, "x", 3)                             #differentiate 3 times
a * (4 * (3 * (2 * x)))

> f <- function(x) x^2                      #define f
> integrate(f, 0, 1)                        #integrate between 0 and 1
0.3333333 with absolute error < 3.7e-15

> integrate(sin, 0, pi)
2 with absolute error < 2.2e-14

> integrate(dnorm, -Inf, 1.96)
0.9750021 with absolute error < 1.3e-06
```

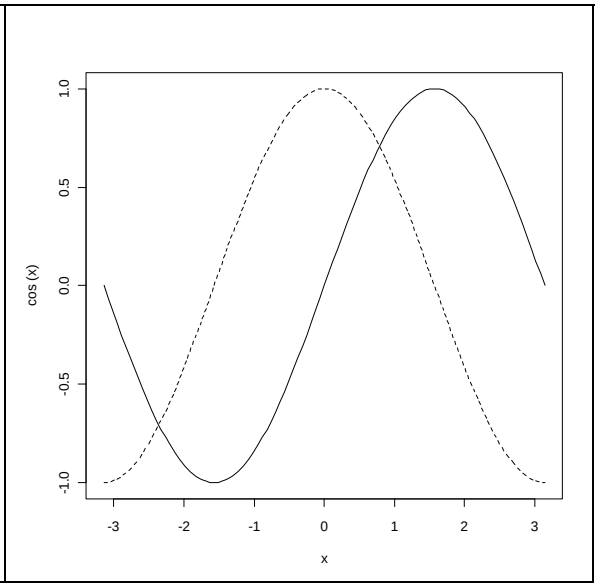
Dataframe:

```
> sex    <- c("F","F","M","M","M")
> height <- c(158,162,177,173,166)
> weight <- c(51,55,72,57,64)
> ( x    <- data.frame(SEX=sex, HEIGHT=height, WEIGHT=weight) )
  SEX HEIGHT WEIGHT
1   F    158     51
2   F    162     55
3   M    177     72
4   M    173     57
5   M    166     64

> mean( x$WEIGHT )
[1] 59.8
> x[,2]
[1] 158 162 177 173 166

> summary(x)
 SEX       HEIGHT          WEIGHT
 F:2   Min.   :158.0   Min.   :51.0
 M:3   1st Qu.:162.0   1st Qu.:55.0
       Median :166.0   Median :57.0
       Mean   :167.2   Mean   :59.8
       3rd Qu.:173.0   3rd Qu.:64.0
       Max.   :177.0   Max.   :72.0
> by(x, x$SEX, summary)
x$SEX: F
 SEX       HEIGHT        WEIGHT
 F:2   Min.   :158   Min.   :51
 M:0   1st Qu.:159   1st Qu.:52
       Median :160   Median :53
```

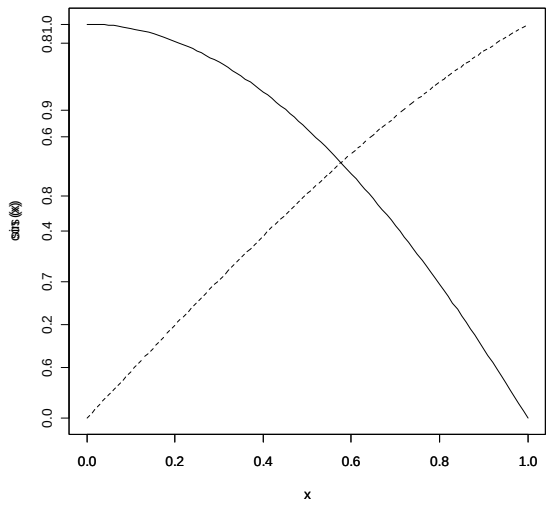
```
       Mean   :160   Mean   :53
       3rd Qu.:161   3rd Qu.:54
       Max.   :162   Max.   :55
------------------------------------------------------------
x$SEX: M
 SEX       HEIGHT          WEIGHT
 F:0   Min.   :166.0   Min.   :57.00
 M:3   1st Qu.:169.5   1st Qu.:60.50
       Median :173.0   Median :64.00
       Mean   :172.0   Mean   :64.33
       3rd Qu.:175.0   3rd Qu.:68.00
       Max.   :177.0   Max.   :72.00

> ( x <- read.csv("D:\\Stat333\\sample1.csv", header=F, col.names=c("SEX","HEIGHT","WEIGHT")) )
  SEX HEIGHT WEIGHT
1   F    158     51
2   F    162     55
3   M    177     72
4   M    173     57
5   M    166     64

> data(sleep)
> with(sleep, {
+   x <- extra[group==1]
+   y <- extra[group==2]
+   var.test(x, y)
+ })

        F test to compare two variances

data:  x and y
F = 0.7983, num df = 9, denom df = 9, p-value = 0.7427
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.198297 3.214123
sample estimates:
ratio of variances
         0.7983426
```

Plot:

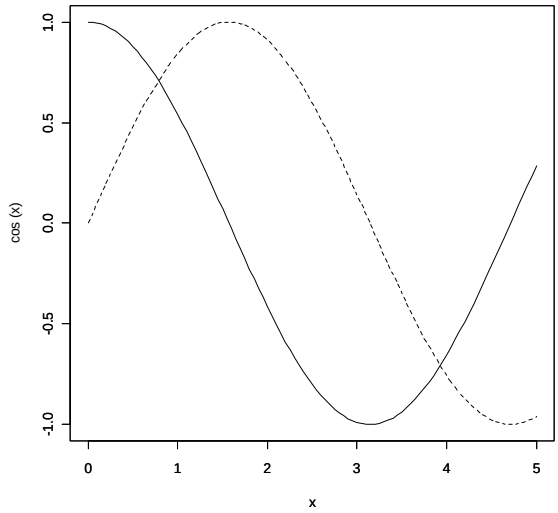
```
> plot(cos, -pi, pi, lty=2)
> curve(sin, add=T)
```



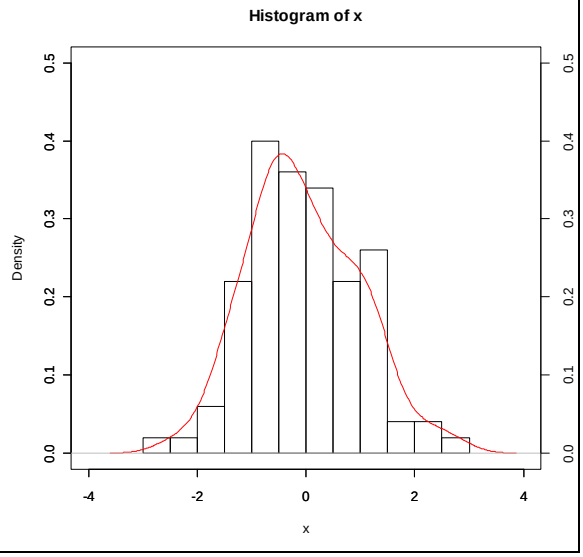
```
> plot(sin, lty=2)
> par(new=T)            #overwriting
> plot(cos)
```



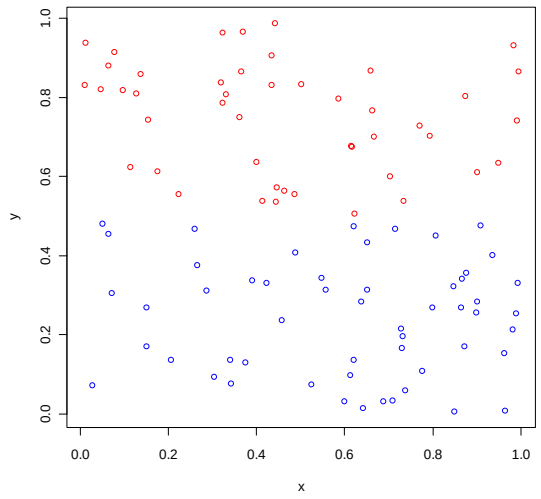

```
> plot(sin, xlim=c(0,5), ylim=c(-1,1), ylab="",
lty=2)   #ann=F works the same
>  par(new=T)
>  plot(cos, xlim=c(0,5), ylim=c(-1,1))
```



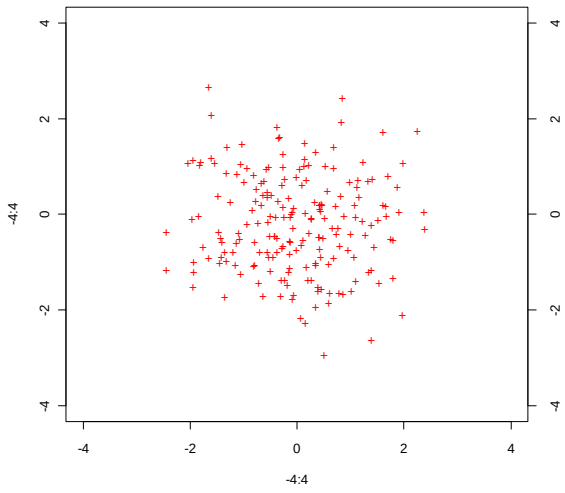

```
> x <- rnorm(100)
> hist(x, xlim=c(-4,4), ylim=c(0,0.5), prob=T,
ann=T)
> par(new=T)
> plot(density(x),xlim=c(-4,4),ylim=
c(0,0.5),xlab="",ylab="",main="",col="red")
> axis(side=4)
> #1=below, 2=left, 3=above, 4=right
```



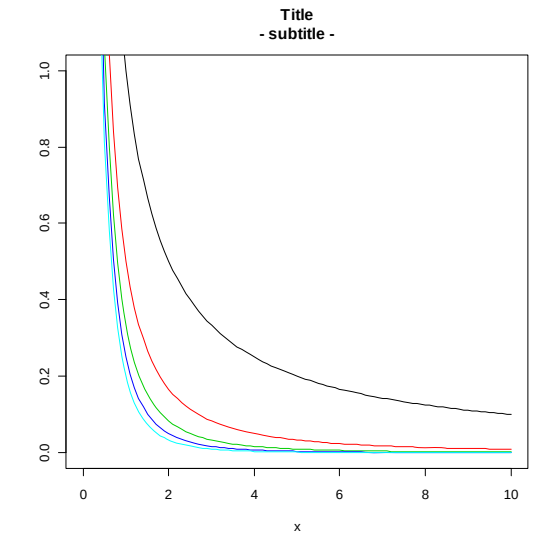
```
> x <- runif(100)
> y <- runif(100)
> plot(x, y, col = ifelse(y>0.5, "red", "blue"))
#red if y > 0.5, blue otherwise
>
```



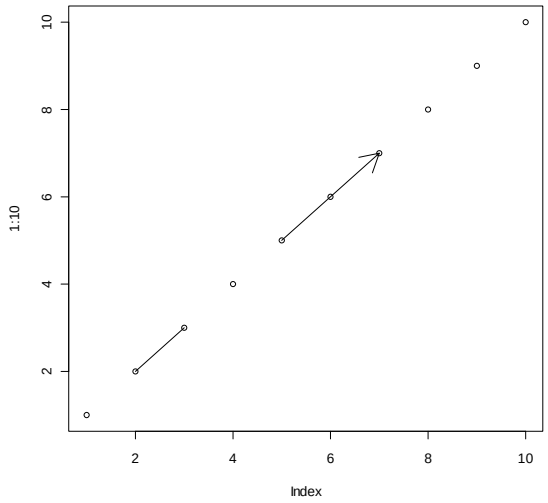

```
> plot(-4:4, -4:4, type = "n")
>  points(rnorm(200), rnorm(200), pch="+", col =
"red")
> axis(side=4)
```



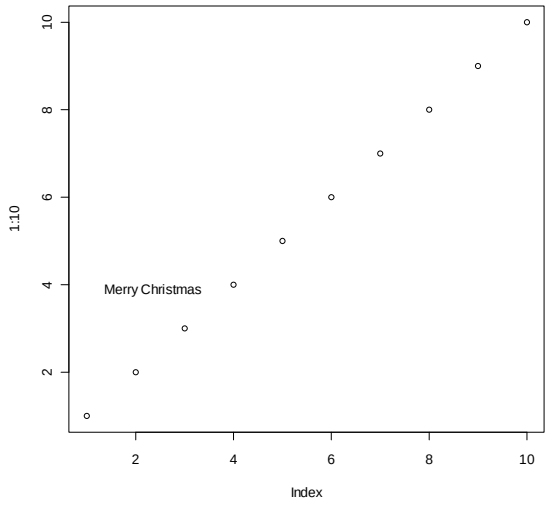

```
> x <- seq(0, 10, by=0.1)
>  y <- seq(0, 1, by=0.01)
>  plot(x, y, ylab="", type='n', main = "Title\n-
subtitle -")
>  for(i in 1:5) lines(x, beta(x,i), col = i)
>
```



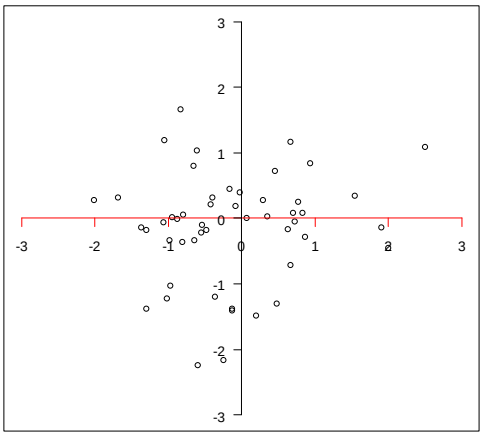
```
> plot(1:10)
>  segments(2,2,3,3)
   #line that connects (2,2) and (3,3)
>  arrows(5,5,7,7)
   #arrow that connects (5,5) and (7,7)
```



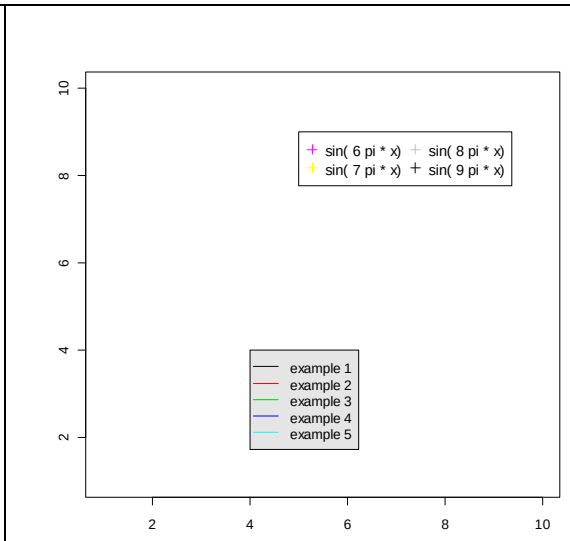

```
> plot(1:10)
>  text(locator(1), labels = "Merry Christmas")
```



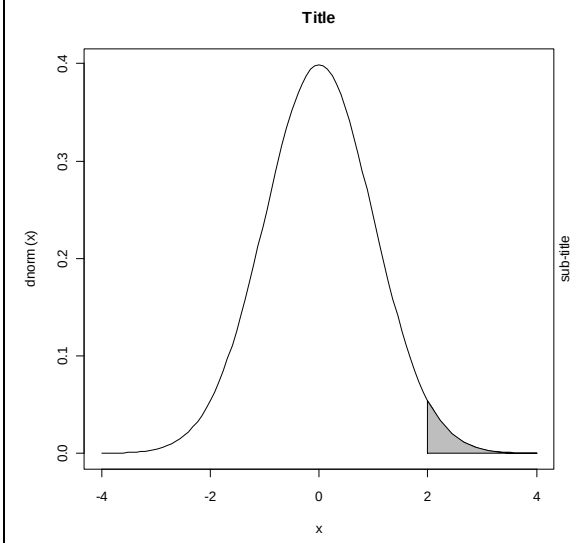

```
> plot(rnorm(50), rnorm(50), xlim=c(-3,3), ylim=c(-
3,3), axes = F, ann=F)
>  axis(1, pos = 0, at = -3:3, adj = 0, col = 2)
#x-axis with red
>  axis(2, pos = 0, at = -3:3, adj = 1, las = 2)
#y-axis with black
>  box()
```



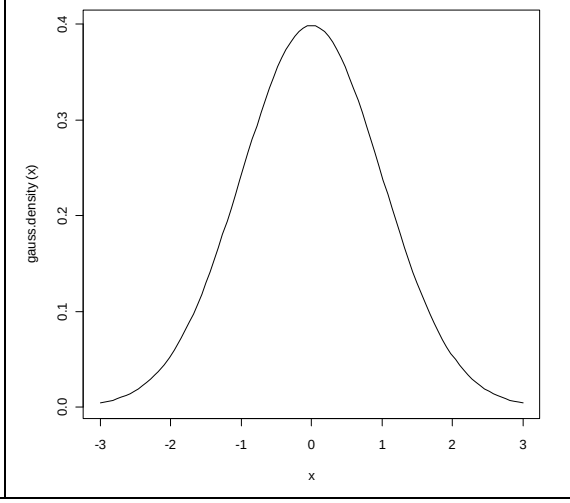
```
> plot(1:10, 1:10, ann=F, type='n')
> legend(4, 4, paste("example",c(1:5)), col =
c(1:5),
+ lwd=1, merge = TRUE, bg='gray90')
> legend(5, 9, paste("sin(",6:9,"pi * x)"),
col=6:9,
+ pch=3, ncol=2, cex=1.1, pt.bg="pink")
```




```
> plot(dnorm, -4, 4)
> xvals <- seq(2, 4, length=10)
> dvals <- dnorm(xvals)
> polygon(c(xvals,rev(xvals)),
+ c(rep(0,10),rev(dvals)),col="gray")
> title("Title")
> mtext("sub-title", side=4)
>
```

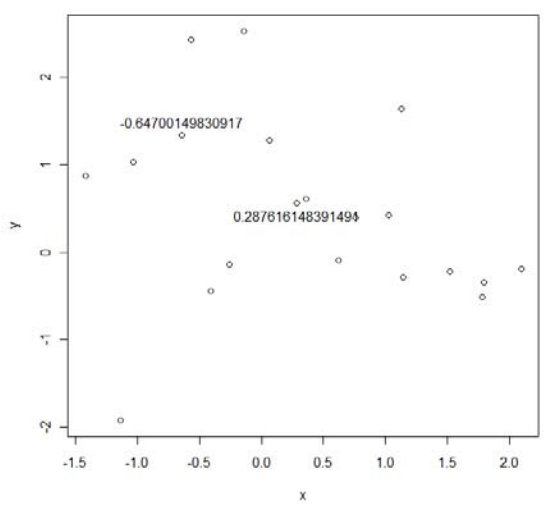



```
> gauss.density <- function(x) 1/sqrt(2*pi)*exp(-
x^2/2)
> plot(gauss.density,-3,3)
```

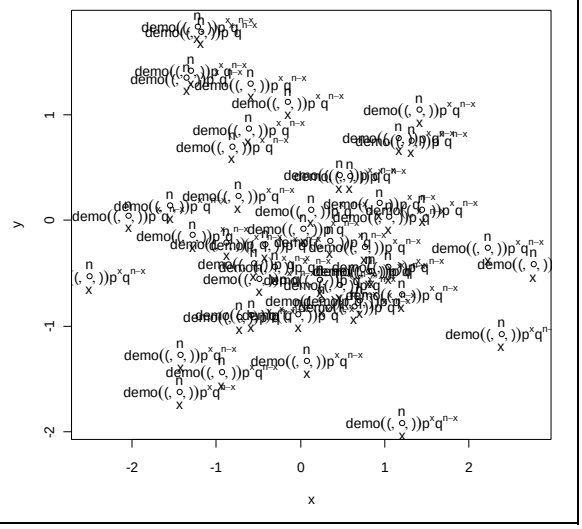


```
> x <- rnorm(20);  y <- rnorm(20)
> plot(x, y)
> identify(x, y, x) #click a point, shows x
coordinate

> identify(x, y, y) #click a point, shows y
coordinate

> identify(x, y) #click a point, shows observation
number
```

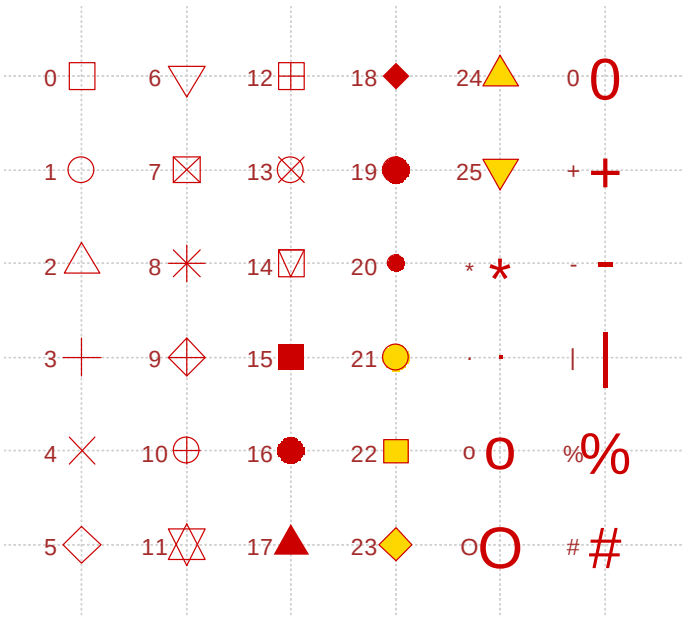
* Hit “Esc” key to terminate.




```
> x <- rnorm(50)
> y <- rnorm(50)
> plot(x, y)
>
text(x,y,expression(paste(demo("(",atop(n,x),")"),p
^x, q^{n-x})))
```



pch symbols:

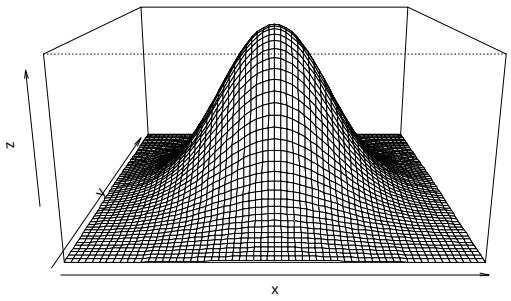


Lines & Types:

parameters	Function
`lwd = 1`	(line width) bigger number means thicker line.
`lty=0 , lty="blank"`	transparent line
`lty=1 , lty="solid"`	solid line
`lty=2 , lty="dashed"`	dashed line
`lty=3 , lty="dotted"`	dotted line
`lty=4 , lty="dotdash"`	
`lty=5 , lty="longdash"`	
`lty=6 , lty="twodash"`	
`type = "p"`	plots points (default)
`type = "l"`	plots lines
`type = "b"`	plots points and lines
`type = "c"`	same as "b" except points
`type = "o"`	plots points and lines
`type = "h"`	plots vertical lines between points and x-axis
`type = "s"`	plots steps based on left-hand-side values
`type = "S"`	plots steps based on right-hand-side values
`type = "n"`	does not plot points (in case of continuing with low-level plots)

3D graph:

```
> x <- seq(-3,3,0.1)
> y <- seq(-3,3,0.1)
> f <- function(x,y) 1/(2*pi)*exp(-(x^2+y^2)/2)
> z <- outer(x, y, f)
> persp(x,y,z,expand=0.5)
```

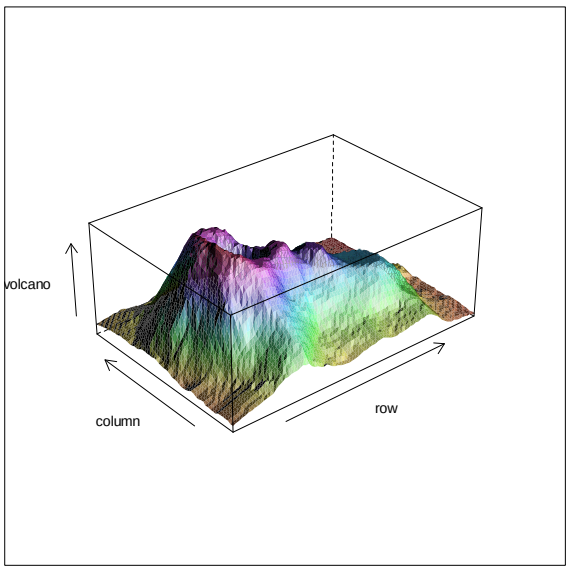


Other functions available in lattice

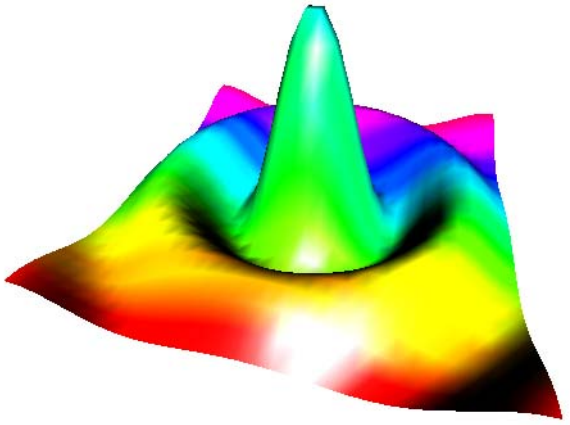
	function names	
1-dimensional plots	barchart()	bar graph
	bwplot()	boxplot
	densityplot()	estimation of density function
	dotplot()	dotplot
	histogram()	histogram
	qqmath()	plot of distribution function
	stripplot()	1-dimensional scatterplot
model fitting	rfs()	residual plot
2-dimensional plots	qq()	2-dimensional q-q plot
	xyplot()	trellis-version plot()
contour plots	levelplot()	contour plot
	contourplot()	contour plot
3-dimensional plots	cloud()	3D scatterplot (points)
	wireframe()	3D scatterplot (plane)
higher-dimensional plots	splom()	scatterplot of a matrix
	parallel()	parallel plot

library(lattice):

```
> library(lattice)
> data(volcano)
> wireframe(volcano, shade = TRUE, aspect =
c(61/87, 0.4), light.source = c(10,0,10))
```

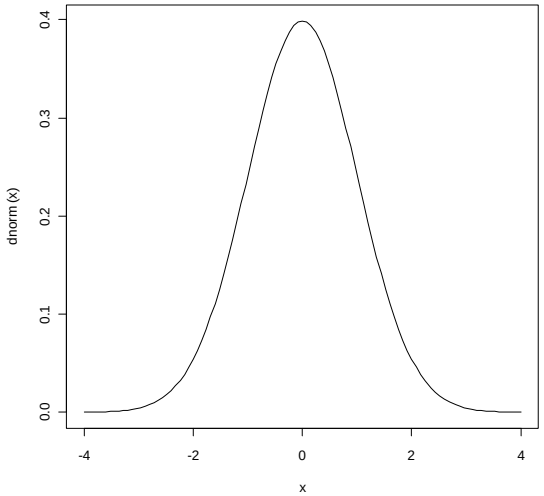



```
> library(rgl)
> rgl.bg(color=c("white", "black"))
> rgl.light(theta = 5, phi = 5)
> x <- seq(-10, 10, length = 30)
> y <- x
> f <- function(x,y) {
+ r <- sqrt(x^2+y^2); 10 * sin(r)/r
+ }
> z <- outer(x, y, f)
> z[is.na(z)] <- 1
> ylim <- range(y)
> ylen <- ylim[2] - ylim[1] + 1
> colorlut <- rainbow(ylen)
> col <- colorlut[ y-ylim[1]+1 ]
> rgl.surface(x, y, z, color=col)
```

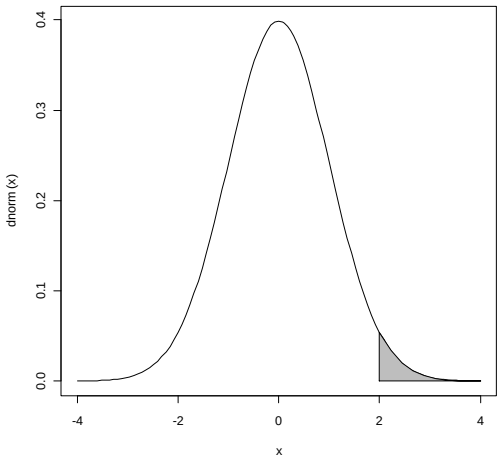


More Graphs:

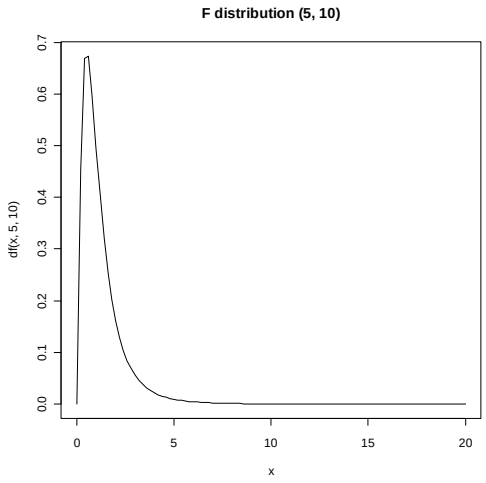
```
> curve(dnorm, -4, 4, type="l")
```



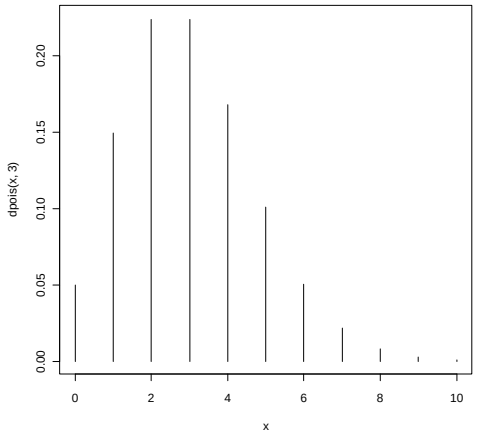

```
> curve(dnorm, -4, 4, type="l")
> xvals <- seq(2, 4, length=10)
> dvals <- dnorm(xvals)
> polygon(c(xvals,rev(xvals)),
+ c(rep(0,10),rev(dvals)),col="gray")
```



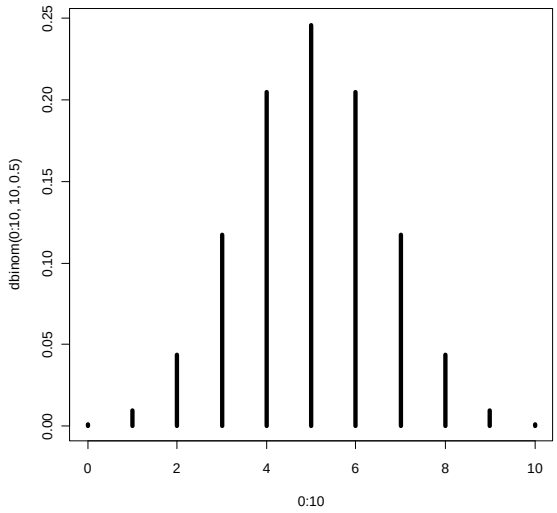
```
> x  <- seq(0,20,0.1)
> curve(df(x,5,10), from=0,to=20,main="F
distribution (5, 10)")
```



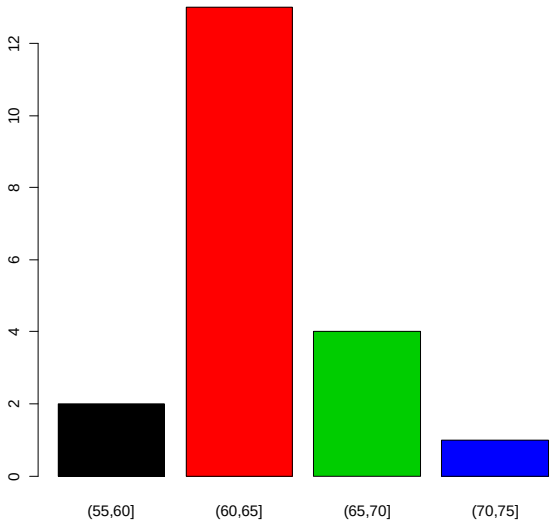

```
> x  <- seq(0,10,1)
> plot(x, dpois(x,3),type="h")
```



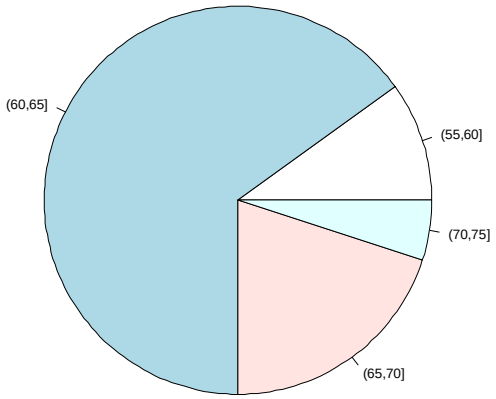

```
> plot(0:10, dbinom(0:10, 10, 0.5),
type="h", lwd=5)
```



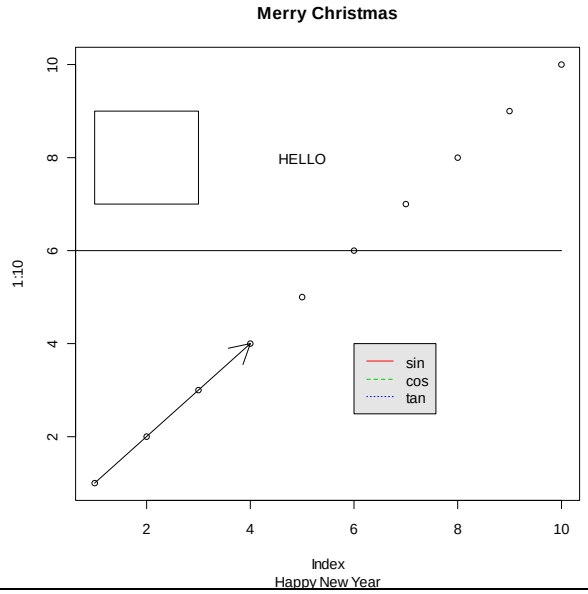

```
> x <- c(59, 60, 61, 62, 62, 63, 63, 63,
63, 64,
+  64, 64, 65, 65, 65, 66, 66, 67, 68, 72)
> y <- table( cut(x, c(55, 60, 65, 70,
75)) )
> y

(55,60] (60,65] (65,70] (70,75]
      2      13       4       1
> barplot(y, col=1:4)
```



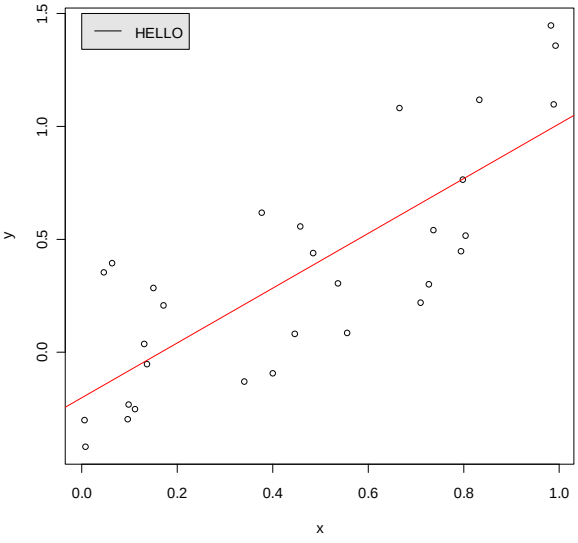
```
> pie(y, r=1.0)
```



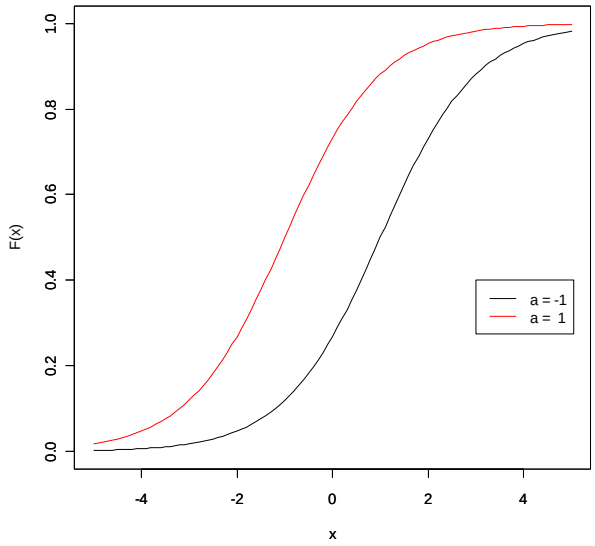

```
> plot(1:10)
> lines(c(0,10), c(6,6))
> rect(1,7, 3,9)
> text(5,8,"HELLO")
> arrows(1,1, 4,4)
> title("Merry Christmas","Happy New Year")
> legend(6, 4,
+ c("sin", "cos", "tan"),
+ col=2:4, lty = 1:3,
+ bg='gray90')
```



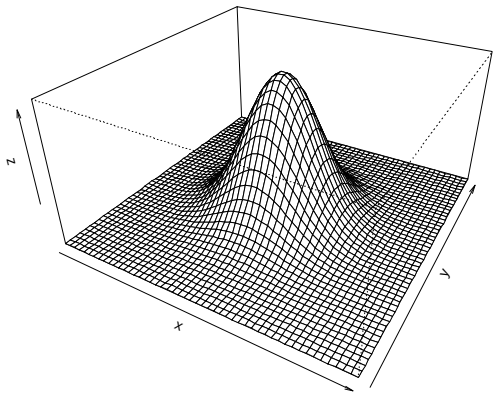

```
> x <- runif(30)
> y <- jitter(x, amount=0.5)
> plot(x,y)
> abline(lm(y~x), col="red")
> legend(0, 1.5, col=1, lty = 1, "HELLO",
bg='gray90')
```



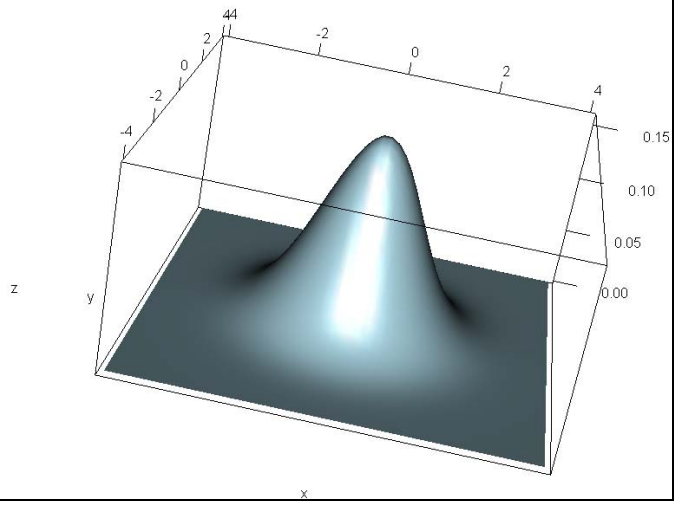
```
> F <- function(x, a) { 1/(1+exp(-a-x)) }
> curve(F(x,-1), col=1, xlim=c(-5,5),
ylim=c(-0,1), ylab="")
> par(new=T)
> curve(F(x, 1), col=2, xlim=c(-5,5),
ylim=c(-0,1), ylab="F(x)")
> legend(3, 0.4, c("a = -1","a =  1"),
lty=1, col=1:2)
```




```
> f <- function(x, y)
return( 1/(2*pi)*exp(-(x^2+y^2)/2) )
> x <- seq(-4, 4, length= 50)
> y <- x
> z <- outer(x, y, f);
> persp(x, y, z, theta = 30, phi = 30,
expand = 0.5)
```



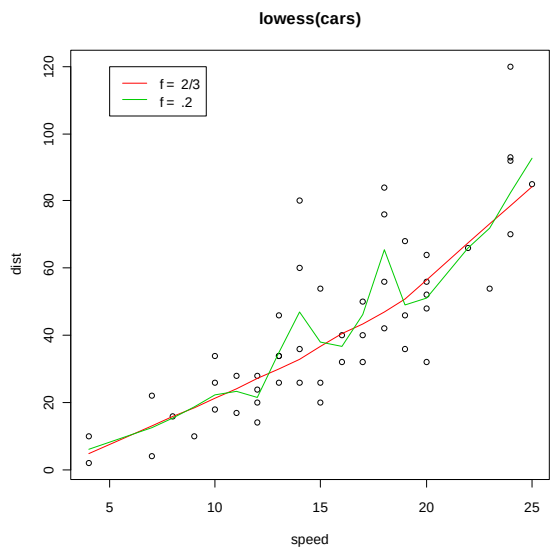

```
> library(rgl)
> open3d()
[1] 2
> persp3d(x, y, z, aspect=c(1,1,0.5),
col="lightblue")
```



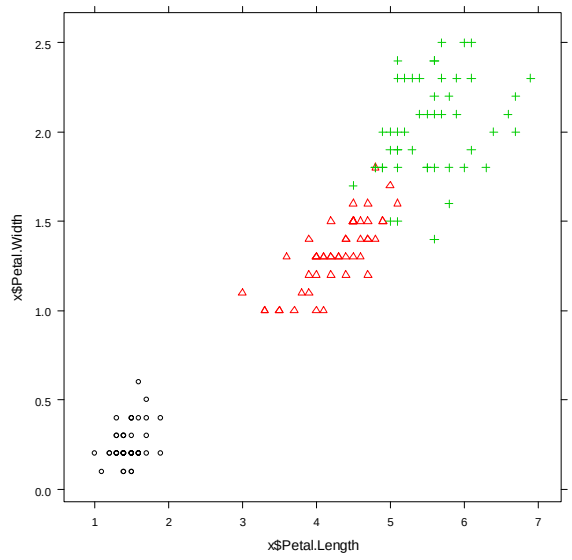
Statistics:

```
> qf(0.95, 2, 3)
[1] 9.552094

> runif(5)
[1] 0.9067795 0.9827270 0.5315387 0.8131461 0.8858163
> runif(5)
[1] 0.9943650 0.9217619 0.7635328 0.1109955 0.2654058
> set.seed(1001); runif(5)
[1] 0.9856888 0.4126285 0.4295392 0.4191722 0.4265066
> set.seed(1001); runif(5)
[1] 0.9856888 0.4126285 0.4295392 0.4191722 0.4265066

> x <- c(0.7,-1.6,-0.2,-1.2,-0.1,3.4,3.7,0.8,0.0,2.0)
> breaks <- seq(-2, 4, 2)
> ( result <- table(cut(x, breaks)) )

(-2,0]  (0,2]  (2,4]
     5      3      2
```

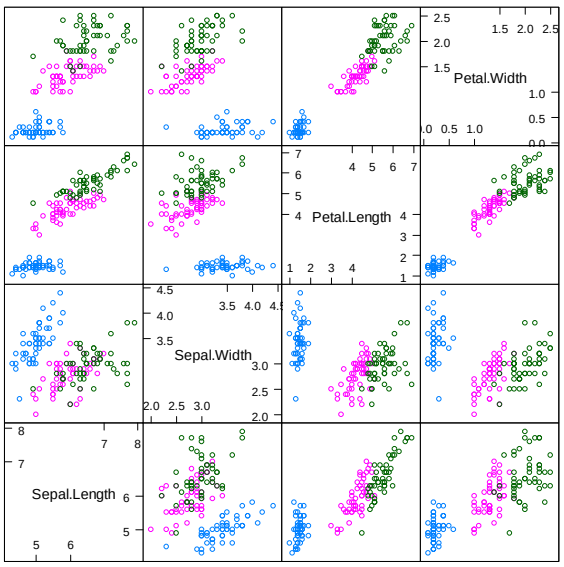
```
> data(cars)
> plot(cars, main = "lowess(cars)")
> lines(lowess(cars), col = 2)
> lines(lowess(cars, f = 0.2), col = 3)
> legend(5, 120, c(paste("f = ", c("2/3",
".2"))), lty = 1, col = 2:3)
```




```
> data(iris)
> library(lattice)
> x <- iris
> xyplot(x$Petal.Width ~ x$Petal.Length,
+ col=as.integer(x$Species),
pch=as.integer(x$Species), ps=20)
```

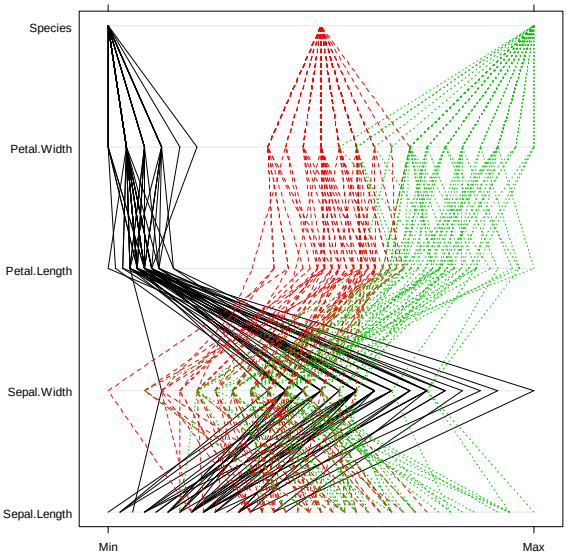


```
> data(iris)
> library(lattice)
> x <- iris
> splom(~ x[,1:4], groups=Species, data=x)
```



Scatter Plot Matrix

```
> data(iris)
> library(lattice)
> x <- iris
> parallel(x, col=as.integer(x$Species),
+ lty=as.integer(x$Species))
```




```
> t.test(extra ~ group, data = sleep, var.equal=T)

        Two Sample t-test

data:  extra by group
t = -1.8608, df = 18, p-value = 0.07919
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -3.3638740  0.2038740
sample estimates:
mean in group 1 mean in group 2
           0.75            2.33

> wilcox.test(extra ~ group, data = sleep)

        Wilcoxon rank sum test with continuity correction

data:  extra by group
W = 25.5, p-value = 0.06933
alternative hypothesis: true location shift is not equal to 0

Warning message:
In wilcox.test.default(x = c(0.7, -1.6, -0.2, -1.2, -0.1, 3.4, 3.7,  :
  cannot compute exact p-value with ties
```

```
> var.test(extra ~ group, data = sleep)

        F test to compare two variances

data:  extra by group
F = 0.7983, num df = 9, denom df = 9, p-value = 0.7427
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.198297 3.214123
sample estimates:
ratio of variances
         0.7983426

> bartlett.test(extra ~ group, data = sleep)

        Bartlett test of homogeneity of variances

data:  extra by group
Bartlett's K-squared = 0.1079, df = 1, p-value = 0.7426
```

```
>  x <- c(70, 72, 62, 64, 71, 76, 60, 65, 74, 72)
>  y <- c(70, 74, 65, 68, 72, 74, 61, 66, 76, 75)

> cor(x, y, method="pearson")
[1] 0.9496011

> cor.test(x, y, method="pearson")

        Pearson's product-moment correlation

data:  x and y
t = 8.5685, df = 8, p-value = 2.656e-05
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.7957471 0.9883181
sample estimates:
      cor
0.9496011

> cor(x, y, method="spearman")
[1] 0.929878

> cor.test(x, y, method="spearman")

        Spearman's rank correlation rho

data:  x and y
S = 11.5701, p-value = 9.713e-05
alternative hypothesis: true rho is not equal to 0
sample estimates:
     rho
0.929878

Warning message:
In cor.test.default(x, y, method = "spearman") :
  Cannot compute exact p-values with ties
```

```
>  power.t.test(n=100, delta=1.0, sd=2.5)

     Two-sample t test power calculation

              n = 100
          delta = 1
             sd = 2.5
      sig.level = 0.05
          power = 0.8036466
    alternative = two.sided

 NOTE: n is number in *each* group

> power.t.test(power=0.9, delta=1.0, sd=2.5)

     Two-sample t test power calculation

              n = 132.3106
```

```
          delta = 1
             sd = 2.5
      sig.level = 0.05
          power = 0.9
    alternative = two.sided

 NOTE: n is number in *each* group
```

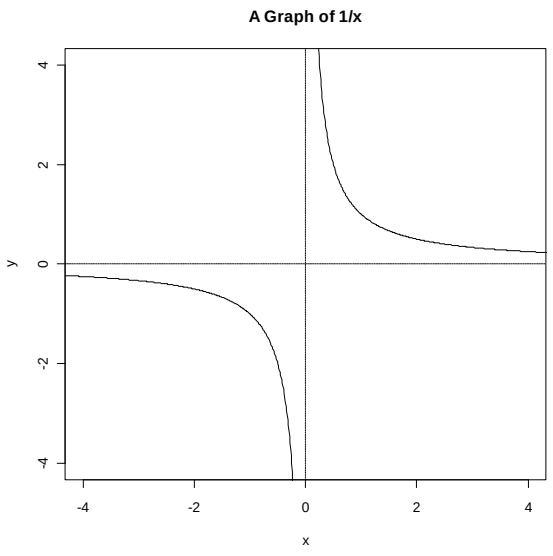
```
> data(trees)
> result1 <- lm(log(Volume) ~ log(Girth), data = trees)
>  summary(result1)

Call:
lm(formula = log(Volume) ~ log(Girth), data = trees)

Residuals:
      Min        1Q    Median        3Q       Max
-0.205999 -0.068702  0.001011  0.072585  0.247963

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.35332    0.23066  -10.20 4.18e-11 ***
log(Girth)   2.19997    0.08983   24.49  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.115 on 29 degrees of freedom
Multiple R-squared: 0.9539,     Adjusted R-squared: 0.9523
F-statistic: 599.7 on 1 and 29 DF,  p-value: < 2.2e-16

> result2 <- update(result1, ~ . + log(Height), data = trees)
>  summary(result2)

Call:
lm(formula = log(Volume) ~ log(Girth) + log(Height), data = trees)

Residuals:
      Min        1Q    Median        3Q       Max
-0.168561 -0.048488  0.002431  0.063637  0.129223

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -6.63162    0.79979  -8.292 5.06e-09 ***
log(Girth)   1.98265    0.07501  26.432  < 2e-16 ***
log(Height)  1.11712    0.20444   5.464 7.81e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.08139 on 28 degrees of freedom
Multiple R-squared: 0.9777,     Adjusted R-squared: 0.9761
F-statistic: 613.2 on 2 and 28 DF,  p-value: < 2.2e-16

> result3 <- step(result2)
Start:  AIC=-152.69
log(Volume) ~ log(Girth) + log(Height)

              Df Sum of Sq    RSS      AIC
<none>                     0.1855 -152.685
- log(Height)  1    0.1978 0.3832 -132.185
- log(Girth)   1    4.6275 4.8130  -53.743
>  summary(result3)

Call:
lm(formula = log(Volume) ~ log(Girth) + log(Height), data = trees)

Residuals:
      Min        1Q    Median        3Q       Max
-0.168561 -0.048488  0.002431  0.063637  0.129223

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -6.63162    0.79979  -8.292 5.06e-09 ***
log(Girth)   1.98265    0.07501  26.432  < 2e-16 ***
log(Height)  1.11712    0.20444   5.464 7.81e-06 ***
---
```

```
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.08139 on 28 degrees of freedom
Multiple R-squared: 0.9777,     Adjusted R-squared: 0.9761
F-statistic: 613.2 on 2 and 28 DF,  p-value: < 2.2e-16

> library(wle)
> result <- mle.cp(log(Volume) ~ log(Girth) + log(Height), data = trees)
>  summary(result)
>  result <- mle.cv(log(Volume) ~ log(Girth) + log(Height), data = trees)
>  summary(result)
>  result <- mle.aic(log(Volume) ~ log(Girth) + log(Height), data = trees)
>  summary(result)

Call:
mle.aic(formula = log(Volume) ~ log(Girth) + log(Height), data = trees)

Akaike Information Criterion (AIC):
     (Intercept) log(Girth) log(Height)      aic
[1,]           1          1           1  -64.560
[2,]           1          1           0  -36.700
[3,]           0          1           1    2.196
[4,]           0          1           0  169.000
[5,]           1          0           1  632.100
[6,]           0          0           1  976.300
[7,]           1          0           0 1158.000
Printed the first  7  best models
```

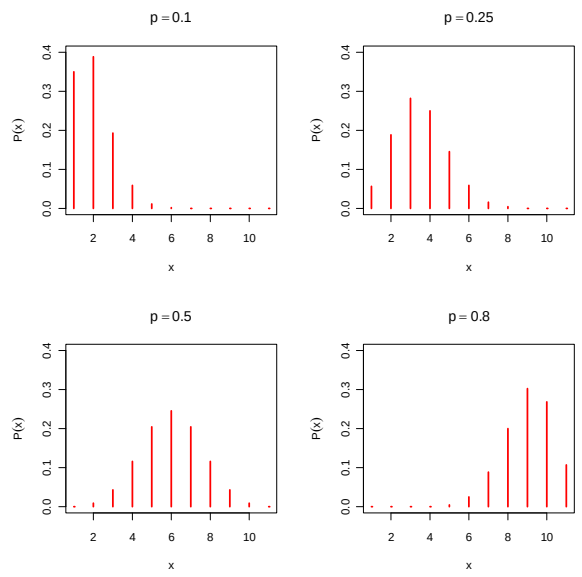
```
> chisq.test(c(8, 12, 10, 9, 5, 6))

        Chi-squared test for given probabilities

data:  c(8, 12, 10, 9, 5, 6)
X-squared = 4, df = 5, p-value = 0.5494

> chisq.test(c(20,8,5,2), p=c(4, 3, 2, 1)/10)

        Chi-squared test for given probabilities

data:  c(20, 8, 5, 2)
X-squared = 4.381, df = 3, p-value = 0.2232

Warning message:
In chisq.test(c(20, 8, 5, 2), p = c(4, 3, 2, 1)/10) :
  Chi-squared approximation may be incorrect
> chisq.test(matrix(c(20, 15, 16, 4, 15, 7, 9, 4), ncol=4, byrow=T))

        Pearson's Chi-squared test

data:  matrix(c(20, 15, 16, 4, 15, 7, 9, 4), ncol = 4, byrow = T)
X-squared = 1.1981, df = 3, p-value = 0.7535

Warning message:
In chisq.test(matrix(c(20, 15, 16, 4, 15, 7, 9, 4), ncol = 4, byrow = T)) :
Chi-squared approximation may be incorrect


> x <- matrix(c(14, 8, 4, 17), ncol=2, byrow=T)
> x
     [,1] [,2]
[1,]   14    8
[2,]    4   17
> chisq.test(x)

        Pearson's Chi-squared test with Yates' continuity correction

data:  x
X-squared = 7.0406, df = 1, p-value = 0.007968

> fisher.test(x)

        Fisher's Exact Test for Count Data

data:  x
p-value = 0.005089
```

```
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
  1.567890 39.549785
sample estimates:
odds ratio
  7.051895

> mcnemar.test(x, correct=F)

        McNemar's Chi-squared test

data:  x
McNemar's chi-squared = 1.3333, df = 1, p-value = 0.2482

> prop.test(c(245, 157), c(300,250), correct=F)

        2-sample test for equality of proportions without continuity
        correction

data:  c(245, 157) out of c(300, 250)
X-squared = 24.6789, df = 1, p-value = 6.772e-07
alternative hypothesis: two.sided
95 percent confidence interval:
 0.1144583 0.2628750
sample estimates:
   prop 1    prop 2
0.8166667 0.6280000

> test <- matrix(c(157,250,88,50),nrow=2,ncol=2)
> test
     [,1] [,2]
[1,]  157   88
[2,]  250   50
> chisq.test(test,correct=F)

        Pearson's Chi-squared test

data:  test
X-squared = 26.4331, df = 1, p-value = 2.728e-07
```

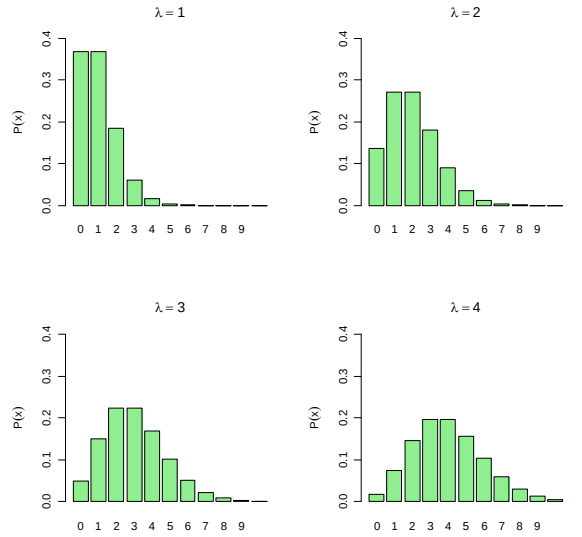
```
> x = seq(-5, 5, length=1001)
> y = 1/x
> plot.new()
> plot.window(xlim=c(-4, 4), ylim=c(-4, 4))
> lines(x, y)
> abline(h=0, v=0, lty="11")
> axis(1)
> axis(2)
> box()
> title(main="A Graph of 1/x", xlab="x",
ylab="y")
```



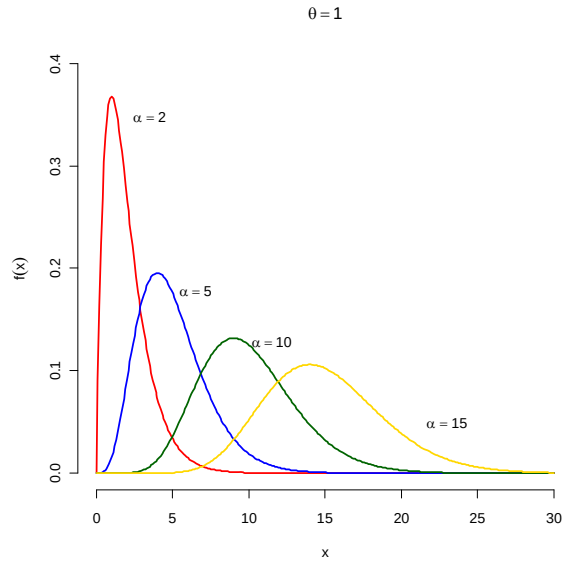
```
> par(mfrow=c(2,2))
> n <- 10
> p <- c(0.1, 0.25, 0.5, 0.8)
> x <- 0:10
> for (i in 1:4)
+ {
+ x1 <- dbinom(x, n, p[i])
+ plot(x1, type="h", xlab="x",
ylab=expression(P(x)),
+ ylim=c(0,.4),lwd=2, col="red")
+ j <- p[i]
+ title(main = substitute(p == j,list(j=j)))
+ }
```




```
> par(mfrow=c(2,2))
> lambda <- 1:4
> x <- 0:10
>
> for (i in lambda)
+ {
+ x1 <- dpois(x, lambda[i])
+ barplot(x1, names.arg=c("0", "1", "2", "3",
"4", "5", "6", "7", "8", "9", "10"),
ylab=expression(P(x)), ylim=c(0,.4),
col="lightgreen")
+ title(main = substitute(lambda ==
i,list(i=i)))
+ }
```

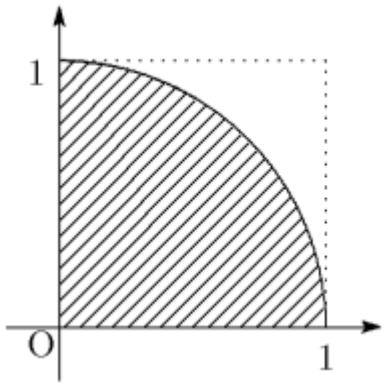



```
> x <- seq(0, 30, length=300)
> hx <- dgamma(x, shape=1, scale=1)
>
> gshape <- c(2, 5, 10, 15)
> colors <- c("red", "blue", "darkgreen",
"gold")
>
> plot(x, hx, type="n", lty=2, lwd=2, xlab="x",
ylab=expression(f(x)), main=expression(theta ==
1), ylim=c(0,0.4), frame.plot=F)
>
> for (i in 1:4){
+ lines(x, dgamma(x,shape=gshape[i], scale=1),
lwd=2, col=colors[i])
+ }
>
> # Inserting mathematical expressions
>
> text(3.5,.35 , expression(paste(alpha==2)))
> text(6.5,.18 , expression(paste(alpha==5)))
> text(11.5,.13 , expression(paste(alpha==10)))
> text(23,.05 , expression(paste(alpha==15)))
```

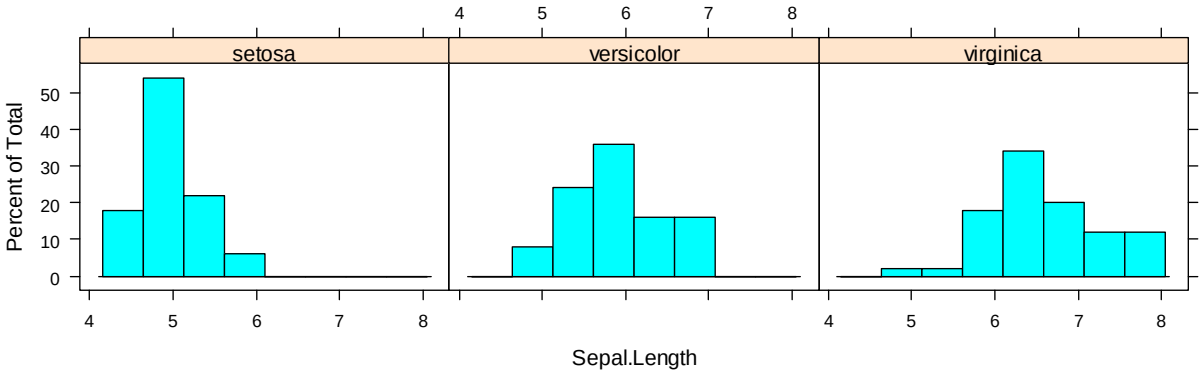


```
> pi.montecarlo <- function(n) {
+ a <- 0
+ for (i in 1:n) {
+ b <- runif(2)
+ if (sqrt(b[1]^2 + b[2]^2) < 1) {
+ a <- a + 1
+ }
+ }
+ return(4 * a / n)
+ }
> pi.montecarlo(50)
[1] 3.2
> pi.montecarlo(5000)
[1] 3.176
> pi.montecarlo(500000)
[1] 3.144072
```

Calculating π



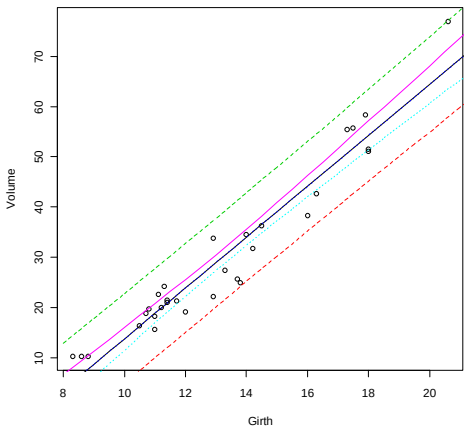

```
> data(iris)
> library(lattice)
> histogram(~ Sepal.Length | Species, data=iris)
```



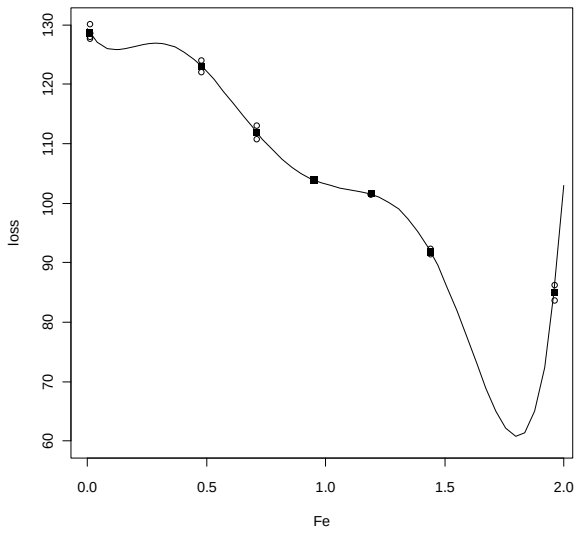
Regression, prediction and plot:

```
> x <- rnorm(30)
> y <- rnorm(30)
> model1 <- lm(y~x)
> a <- c(0.5,1.2,1.9)
> b <- numeric(3)
> for (i in 1:3) {
+ b[i] <- model1$coef[[1]]+(a[i]*model1$coef[[2]]) }
> b
[1] -0.2108714 -0.2158105 -0.2207497
```

```
> library(faraway)
> data(trees)
> plot(Volume ~ Girth, data = trees)
> result <- lm(Volume ~ Girth, data = trees)
> abline(result)
> summary(result)

> new <- data.frame(Girth = seq(8, 24, 0.5))
> result.pre <- predict(result, new,
interval="prediction")
> result.con <- predict(result, new,
interval="confidence")
> matplot(new$Girth, cbind(result.pre,
result.con), lty=c(1,2,2,3,3), type="l",
add=T)
```

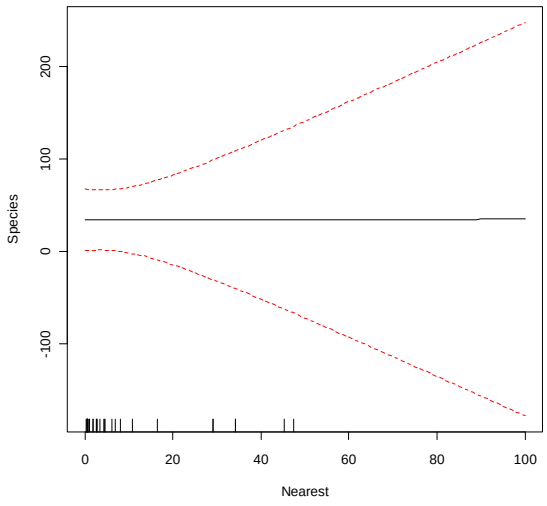


```
> library(faraway)
> data(corrosion)
> model1 <-
lm(loss~Fe+I(Fe^2)+I(Fe^3)+I(Fe^4)+I(Fe^5)+I(Fe
^6),corrosion)
> model2 <- lm(loss~factor(Fe),corrosion)
> plot(loss~Fe,data=corrosion,ylim=c(60,130))
> points(corrosion$Fe,fitted(model2), pch=15)
> grid <- seq(0,2,length=50)
> lines(grid, predict(modelp,
data.frame(Fe=grid)))
```



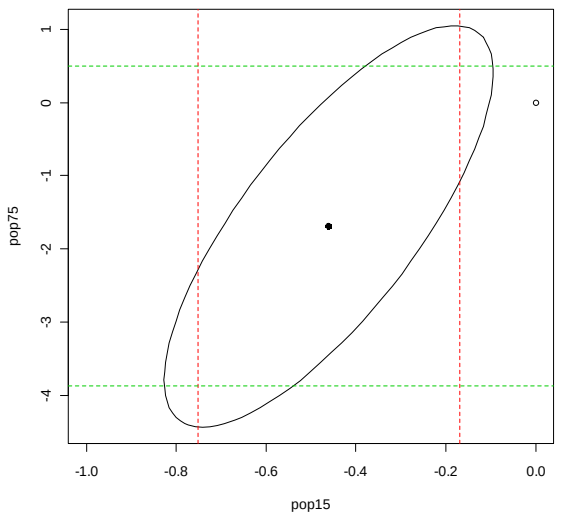
Prediction band on a single variable:

```
> library(faraway)
> data(gala)
> dim(gala)
[1] 30  7
> gala[1:3,]
          Species Endemics  Area Elevation
Nearest Scruz Adjacent
Baltra         58       23 25.09       346
0.6   0.6     1.84
Bartolome      31       21  1.24       109
0.6  26.3   572.33
Caldwell        3        3  0.21       114
2.8  58.7     0.78
> model1 <-
lm(Species~Area+Elevation+Nearest+Scruz+Adjacen
t, data=gala)
> grid <- seq(0,100,by=1)
> p <- predict(model1,
data.frame(Area=0.08,Elevation=93,Nearest=grid,
Scruz=12,
+ Adjacent=0.34),se=T,interval="confidence")
>
matplot(grid,p$fit,lty=c(1,2,2),type="l",col=c(
1,2,2),xlab="Nearest",ylab="Species")
> rug(gala$Nearest)
```



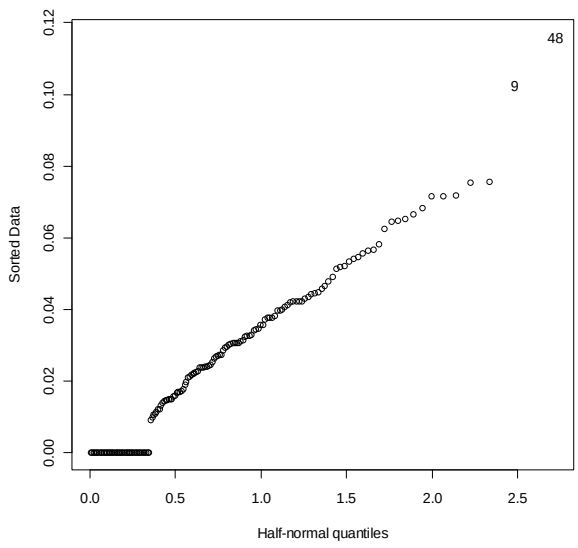
Confidence ellipse of regression coefficients:

```
> library(faraway)
> data(savings)
> dim(savings)
[1] 50  5
> savings[1:3,]
             sr pop15 pop75     dpi ddpi
Australia 11.43 29.35  2.87 2329.68 2.87
Austria   12.07 23.32  4.41 1507.99 3.93
Belgium   13.17 23.80  4.43 2108.47 3.82
> model1 <- lm(sr ~., savings)
> confint(model1)
                   2.5 %       97.5 %
(Intercept) 13.753330728 43.378842354
pop15       -0.752517542 -0.169868752
pop75       -3.873977955  0.490982602
dpi         -0.002212248  0.001538444
ddpi         0.014533628  0.804856227
> library(ellipse)
> model1

Call:
lm(formula = sr ~ ., data = savings)

Coefficients:
(Intercept)        pop15        pop75
dpi         ddpi
 28.5660865   -0.4611931   -1.6914977   -
0.0003369    0.4096949

> plot(ellipse(model1,c(2,3)),type="l",xlim=c(-
1,0))
> points(0,0)
>
points(coef(model1)[2],coef(model1)[3],pch=19)
> abline(v=confint(model1)[2,],lty=2, col=2)
> abline(h=confint(model1)[3,],lty=2, col=3)
```



Multicollinearity and influence statistics:

```
> library(faraway)
> data(airquality)
> dim(airquality)
[1] 153   6
> airquality[1:3,]
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
> model1 <-
lm(Ozone~Solar.R+Wind+Temp,airquality,na.action
=na.exclude)

> vif(model1)
 Solar.R     Wind     Temp
1.095253 1.329070 1.431367
> halfnorm(lm.influence(model1)$hat)
```




```
> library(faraway)
> data(seatpos)
> dim(seatpos)
[1] 38  9
> seatpos[1:3,]
  Age Weight HtShoes    Ht Seated  Arm Thigh  Leg hipcenter
1  46    180   187.2 184.9   95.2 36.1  45.3 41.3  -206.300
2  31    175   167.5 165.5   83.8 32.9  36.5 35.9  -178.210
3  23    100   153.6 152.2   82.9 26.0  36.6 31.0   -71.673
> model1 <- lm(hipcenter ~., seatpos)
> x <- model.matrix(model1)[,-1]
```

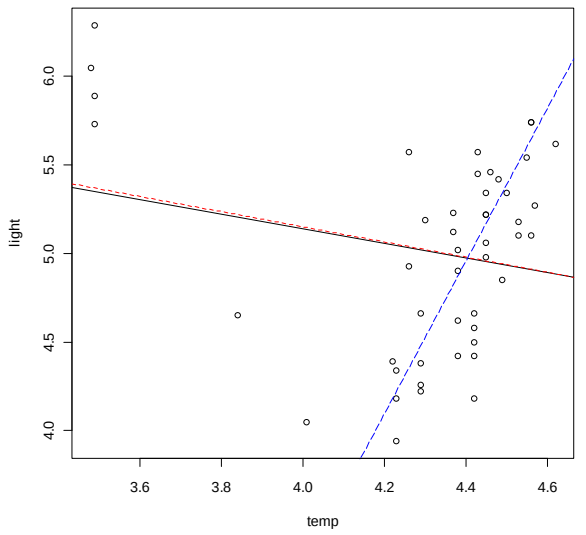
```
> e <- eigen(t(x) %*% x)
> round(e$val, 3)
[1] 3653671.363   21479.480   9043.225    298.953    148.395     81.174    53.362     7.298
> sqrt(e$val[1]/e$val)
[1]   1.00000  13.04226  20.10032 110.55123 156.91171 212.15650 261.66698 707.54911
> #Here is how to compute VIF number
> 1/(1-(summary(lm(x[,1]~x[,-1]))$r.squared))
[1] 1.997931
> vif(x)
       Age     Weight    HtShoes         Ht     Seated        Arm      Thigh        Leg
  1.997931   3.647030 307.429378 333.137832   8.951054   4.496368   2.762886   6.694291
```

or

```
> vif(model1)
       Age     Weight    HtShoes         Ht     Seated        Arm      Thigh        Leg
  1.997931   3.647030 307.429378 333.137832   8.951054   4.496368   2.762886   6.694291
```

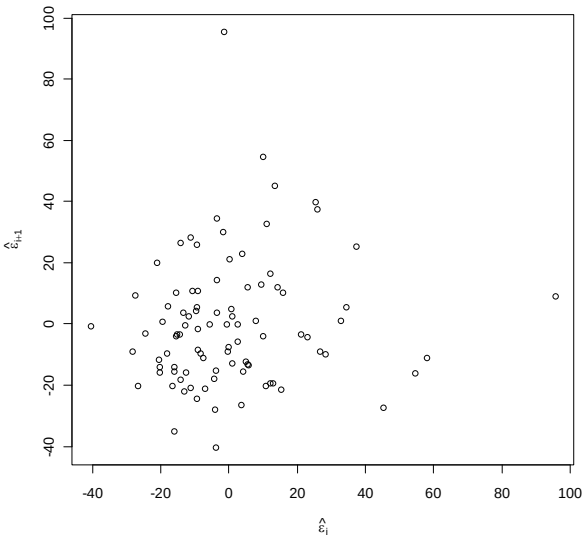
Robust regression, Least Trimmed Squared regression:

```
> library(faraway)
> library(MASS)
> data(star)
> plot(light~temp,star)
> model1 <- lm(light~temp, star)
> abline(coef(model1))
> model2 <- rlm(light~temp, star)
> abline(coef(model2), lty=2, col=2)
> model3 <- ltsreg(light~temp, star,
nsamp="exact")
> abline(coef(model3), lty=5, col=4)
```



Durbin-Watson test & checking residuals for autocorrelation:

```
> library(faraway)
> data(airquality)
> dim(airquality)
[1] 153   6
> airquality[1:3,]
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
> model1 <-
lm(Ozone~Solar.R+Wind+Temp,airquality,na.action
=na.exclude)
> plot(residuals(model1)[-
153],residuals(model1)[-1],xlab=
+
expression(hat(epsilon)[i]),ylab=expression(hat
(epsilon)[i+1]))

> summary(lm(residuals(model1)[-1] ~
residuals(model1)[-153])
+ )

Coefficients:
                        Estimate Std. Error t
value Pr(>|t|)
(Intercept)              -0.1439     2.1905  -
0.066    0.948
residuals(model1)[-153]   0.1093     0.1067
1.025    0.308
```




```
> library(lmtest)
>
dwtest(Ozone~Solar.R+Wind+Temp,data=na.omit(air
quality))
```

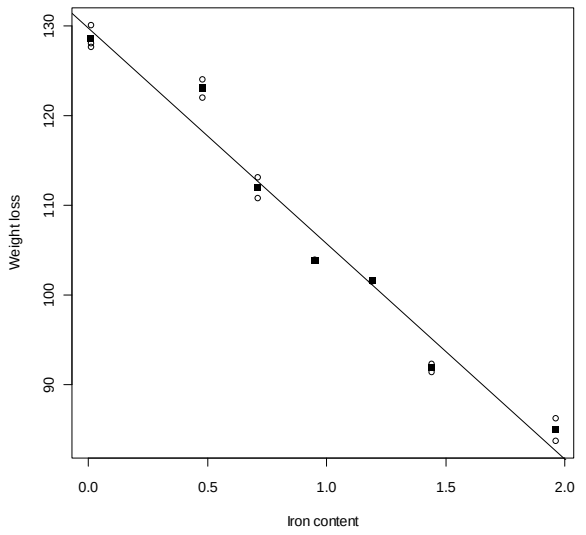
```
Residual standard error: 21 on 90 degrees of freedom
  (60 observations deleted due to missingness)
Multiple R-squared: 0.01153,    Adjusted R-squared: 0.0005466
F-statistic:  1.05 on 1 and 90 DF,  p-value: 0.3083
```

```
        Durbin-Watson test

data:  Ozone ~ Solar.R + Wind + Temp
DW = 1.9355, p-value = 0.3347
alternative hypothesis: true autocorrelation is greater than 0
```

```
> library(faraway)
> library(nlme)
> data(longley)
> dim(longley)
[1] 16  7
> longley[1:3,]
     GNP.deflator     GNP Unemployed Armed.Forces Population Year Employed
1947         83.0 234.289      235.6        159.0    107.608 1947   60.323
1948         88.5 259.426      232.5        145.6    108.632 1948   61.122
1949         88.2 258.054      368.2        161.6    109.773 1949   60.171
> model1 <- gls(Employed~GNP+Population, correlation=corAR1(form=~Year),data=longley)
> summary(model1)
Generalized least squares fit by REML
  Model: Employed ~ GNP + Population
  Data: longley
       AIC      BIC    logLik
  44.66377 47.48852 -17.33188

Correlation Structure: AR(1)
 Formula: ~Year
 Parameter estimate(s):
      Phi
0.6441692

Coefficients:
               Value Std.Error   t-value p-value
(Intercept) 101.85813 14.198932  7.173647  0.0000
GNP           0.07207  0.010606  6.795485  0.0000
Population   -0.54851  0.154130 -3.558778  0.0035

 Correlation:
           (Intr) GNP
GNP         0.943
Population -0.997 -0.966

Standardized residuals:
       Min         Q1        Med         Q3        Max
-1.5924564 -0.5447822 -0.1055401  0.3639202  1.3281898

Residual standard error: 0.689207
Degrees of freedom: 16 total; 13 residual
> intervals(model1)
Approximate 95% confidence intervals

 Coefficients:
                  lower         est.       upper
(Intercept) 71.18320460 101.85813305 132.5330615
GNP          0.04915865   0.07207088   0.0949831
Population  -0.88149053  -0.54851350  -0.2155365
attr(,"label")
[1] "Coefficients:"

 Correlation structure:
      lower      est.    upper
Phi -0.4432383 0.6441692 0.964504          #So there is no serial correlation.
attr(,"label")
[1] "Correlation structure:"

 Residual standard error:
    lower      est.     upper
0.2477527 0.6892070 1.9172599
```

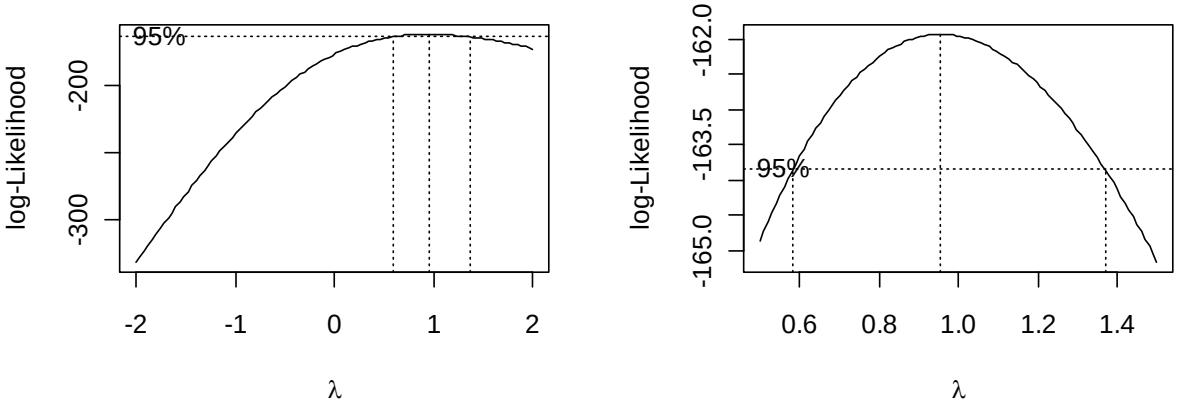
Lack of fit:

```
> library(faraway)
> data(corrosion)
> dim(corrosion)
[1] 13  2
> corrosion[1:3,]
    Fe  loss
1 0.01 127.6
2 0.48 124.0
3 0.71 110.8
> model1 <- lm(loss~Fe, corrosion)
> model2 <- lm(loss~factor(Fe), corrosion)
> plot(loss~Fe,corrosion,xlab="Iron
content",ylab="Weight loss")
> abline(coef(model1))
> points(corrosion$Fe, fitted(model2),pch=15)

> anova(model1, model2)
Analysis of Variance Table

Model 1: loss ~ Fe
Model 2: loss ~ factor(Fe)
  Res.Df     RSS Df Sum of Sq      F   Pr(>F)
1     11 102.850
2      6  11.782  5    91.069 9.2756 0.008623
**
```



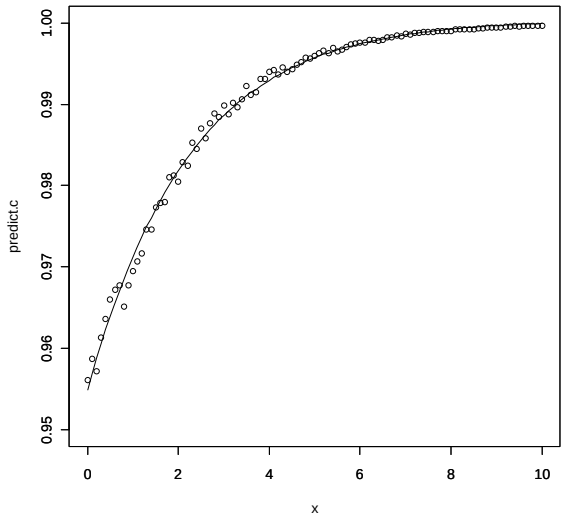
Boxcox method to find optimal transformation of y:

```
> library(faraway)
> library(MASS)
> data(savings)
> dim(savings)
[1] 50  5
> savings[1:3,]
             sr pop15 pop75     dpi ddpi
Australia 11.43 29.35  2.87 2329.68 2.87
Austria   12.07 23.32  4.41 1507.99 3.93
Belgium   13.17 23.80  4.43 2108.47 3.82
> model1 <- lm (sr ~ pop15+pop75+dpi+ddpi, savings)
> par(mfrow=c(1,2))
> boxcox(model1, plotit=T)
> boxcox(model1, plotit=T,lambda=seq(0.5,1.5, by=0.1))
```



Nonlinear Regression:

```
> f <- function(x) 1/(1+exp(-3-0.5*x))
> x <- seq(0,10,0.1); y <- f(jitter(x,
amount=0.2))
> ( result <- nls(y ~ a/(1+exp(-b-c*x)),
start=c(a=2,b=1,c=0.1)) )
Nonlinear regression model
  model:  y ~ a/(1 + exp(-b - c * x))
   data:  parent.frame()
     a      b      c
1.0004 3.0449 0.4617
 residual sum-of-squares: 6.689e-05

Number of iterations to convergence: 7
Achieved convergence tolerance: 5.922e-06
> predict.c <- predict(result)
> plot(x, y, ann=F, xlim=c(0,10),
ylim=c(0.95,1)); par(new=T)
> plot(x, predict.c, type="l", xlim=c(0,10),
ylim=c(0.95,1))
```



C_p, CV, and AIC:

```
> install.packages("wle")
> library(wle)
> data(trees)
> result1 <- mle.cp(log(Volume) ~ log(Girth) + log(Height), data = trees)
> result2 <- mle.cv(log(Volume) ~ log(Girth) + log(Height), data = trees)
> result3 <- mle.aic(log(Volume) ~ log(Girth) + log(Height), data = trees)
> summary(result1)
> summary(result2)
> summary(result3)
```

update:

```
> result1 <- lm(log(Volume) ~ log(Girth), data = trees)
> result2 <- update(result1, ~ . + log(Height), data = trees)
> summary(result2)
> result3 <- step(result2)
```

To get **tally** of a categorical variable:

```
> library(HSAUR)
> data(Forbes2000) #demo dataset
> table(Forbes2000[,4])
```

How many **levels** of a categorical variable and what are they?

```
> library(HSAUR)
> data(Forbes2000) #demo dataset
> nlevels(Forbes2000[,4])
[1] 27
> levels(Forbes2000[,4])
```

To select **subset** of data with "assests" >1000 only:

```
> library(HSAUR)
> data(Forbes2000)
> Forbes2000[Forbes2000$assets > 1000, c("names","sales","profits",
+ "assets")]
```

or

```
> table(Forbes2000$assets > 1000)
```

or

```
> UK <- subset(Forbes, country=="United Kingdom")
```

or

```
> boxplot(log(marketvalue)~country, data=subset(Forbes2000, country %in% c("United Kingdom",
"Germany", "India", "Turkey")), varwidth=TRUE)
```

par(mfrow=c(3,3)):

```
> par(mfrow=c(3,3))
```

To get **summary** statistics:

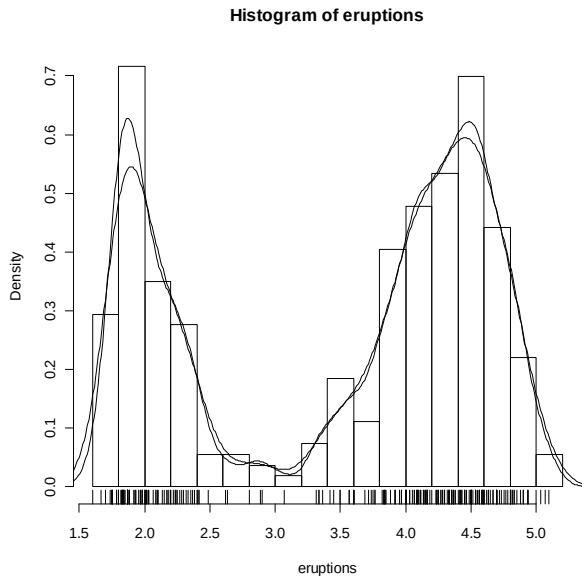
```
> library(HSAUR)
> data(Forbes2000)
> lapply(Forbes2000, summary)
```

or

```
> mprofits  <- tapply(Forbes2000$profits, Forbest2000$category, median, na.rm=TRUE)
```

Basic statistics:

```
> library(HSAUR)
> attach(faithful)
> fivenum(eruptions)
> stem(eruptions)

> hist(eruptions)
> hist(eruptions, seq(1.6, 5.2, 0.2))

> hist(eruptions, seq(1.6, 5.2, 0.2),
prob=TRUE)
> lines(density(eruptions, bw=0.1))
> lines(density(eruptions, bw="SJ"))
> rug(eruptions)

> plot(ecdf(eruptions), do.points=FALSE,
verticals=TRUE)
> long <- eruptions[eruptions>3]
> plot(ecdf(long), do.points=FALSE,
verticals=TRUE)
> x <- seq(3, 5.4, 0.01)
> lines(x, pnorm(x, mean=mean(long),
sd=sqrt(var(long))), lty=3)
>
> par(pty="s") #for square plotting region
> qqnorm(long); qqline(long)
```



QQ plot of t numbers:

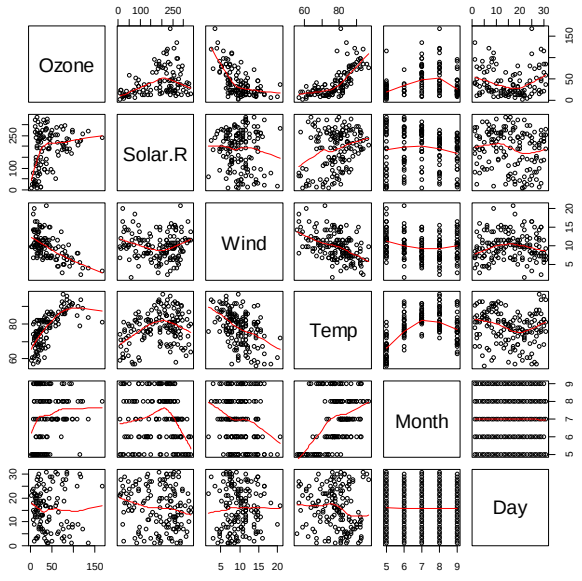
```
> qqplot(qt(ppoints(250), df=5), x, xlab="Q-Q plot for t distribution")
> qqline(x)

> qqnorm(residuals(g), ylab="Residuals") #after fitting a model "g". Use datax=T to reverse axes.
> qqline(residuals(g))
```

Variance test (I):

```
> var.test(residuals(g)[savings$pop15 > 35], residuals(g)[savings$pop15 < 35])
```

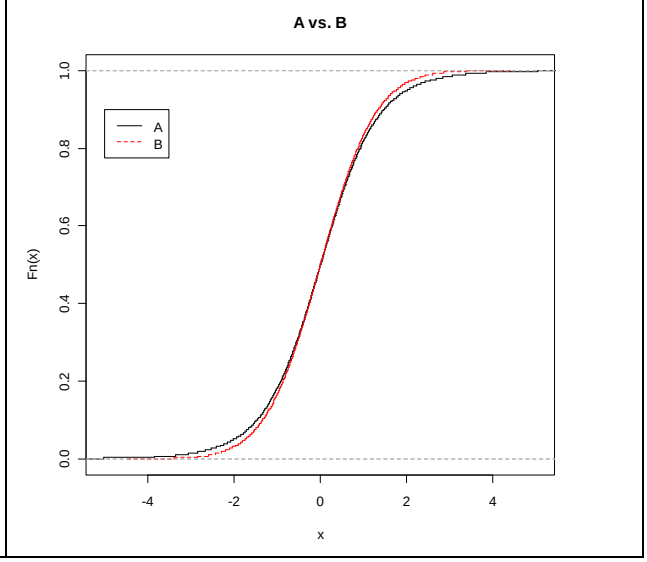
Matrix plot with smooth curve on top

```
> library(faraway)
> data(airquality)
> pairs(airquality, panel=panel.smooth)
```



Comparing A & B, Shapiro test, KS test, Variance test (II):

```
> A <- c(qt(ppoints(250), df=5))
> B <- c(qt(ppoints(250), df=15))
> var.test(A,B)
> t.test(A,B)
> wilcox.test(A, B)
> ks.test(A, B)

> plot(ecdf(A), do.points=FALSE,
verticals=TRUE, xlim=range(A,B), main="A vs.
B")
> plot(ecdf(B), do.points=FALSE,
verticals=TRUE, col=2, lty=2,add=TRUE)
> legend(-5,0.9,c("A","B"),col=1:2,lty=1:2)
```



To **sort** in reverse order (i.e., from big to small):

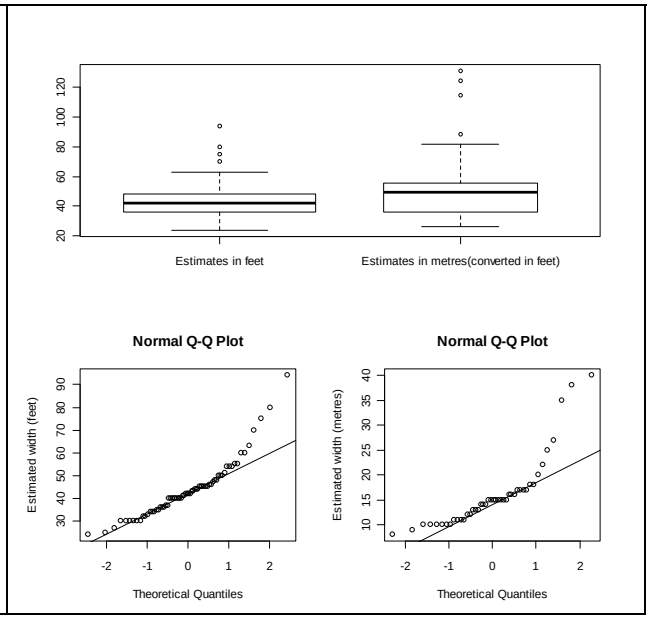
```
> rev(sort(mprofits))[1:3]
```

If data are in meters, multiply **3.28,** but leave it if it's in "feet":

```
> library(HSAUR)
> data(roomwidth) #demo dataset

> convert <- ifelse(roomwidth$unit == "feet", 1, 3.28)
> tapply(roomwidth$width*convert, roomwidth$unit, sd)
```

Layout and **boxplot** for each case:

```
> layout(matrix(c(1,2,1,3),nrow=2,ncol=2,
byrow=FALSE))
> boxplot(I(width*convert)~unit,
data=roomwidth, varwidth=TRUE,
names=c("Estimates in feet", "Estimates in
metres(converted in feet)"))
> feet <- roomwidth$unit == "feet"
> qqnorm(roomwidth$width[feet], ylab="Estimated
width (feet)")
> qqline(roomwidth$width[feet])
> qqnorm(roomwidth$width[!feet],
ylab="Estimated width (metres)")
> qqline(roomwidth$width[!feet])
```

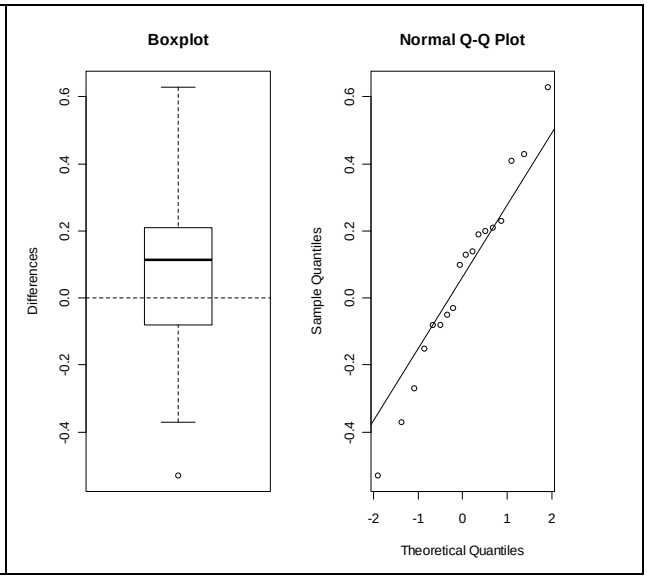


Two-sample t-test and **Wilcoxon rank sum test** (grouped data):

```
> wilcox.test(I(width*convert)~unit, data=roomwidth, conf.int=TRUE)
```

Boxplot, **t-test**, and **Wilcoxon signed rank test** for **paired data**:

```
> library(HSAUR)
> data(waves)
> data(waves)
> dim(waves)
[1] 18  2
> waves[1:3,]
  method1 method2
1    2.23    1.82
2    2.55    2.42
3    7.99    8.26
> diff <- waves$method1 - waves$method2
> layout(matrix(1:2, ncol=2))
> boxplot(diff, ylab="Differences",
main="Boxplot")
> abline(h=0, lty=2)
> qqnorm(diff); qqline(diff)

> wilcox.test(diff)
> t.test(diff, mu=0)
```



Legend of a plot:

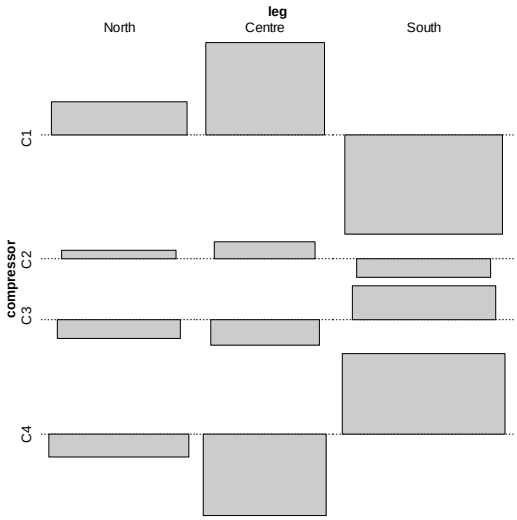
```
> nf <- layout(matrix(c(2,0,1,3),2,2,byrow=TRUE))
> psymb <- as.numeric(water$location)
> plot(mortality~hardness, data=water, pch=psymb)
> abline(lm(mortality~hardness, data=water))
> legend("topright", legend=levels(water$location),pch=c(1,2),bty="n")
> hist(water$hardness)
> boxplot(water$mortality)
```

Correlation test:

```
> cor.test(~mortality+hardness, data=water)
```

4×3 table & χ^2 test:

```
> library(HSAUR)
> data(pistonrings)
> dim(pistonrings)
[1] 4 3
> pistonrings[1:3,]
          leg
compressor North Centre South
        C1    17     17    12
        C2    11      9    13
        C3    11      8    19
> str(pistonrings)
 table [1:4, 1:3] 17 11 11 14 17 9 8 7 12 13 ...
 - attr(*, "dimnames")=List of 2
  ..$ compressor: chr [1:4] "C1" "C2" "C3" "C4"
  ..$ leg       : chr [1:3] "North" "Centre" "South"
> chisq.test(pistonrings)

        Pearson's Chi-squared test

data:  pistonrings
X-squared = 11.7223, df = 6, p-value = 0.06846
> chisq.test(pistonrings)$residuals
          leg
compressor      North     Centre      South
        C1  0.6036154  1.6728267 -1.7802243
        C2  0.1429031  0.2975200 -0.3471197
        C3 -0.3251427 -0.4522620  0.6202463
        C4 -0.4157886 -1.4666936  1.4635235
> library(vcd)
> assoc(pistonrings)
```



2×2 table **McNemar** test, **binomial** test:

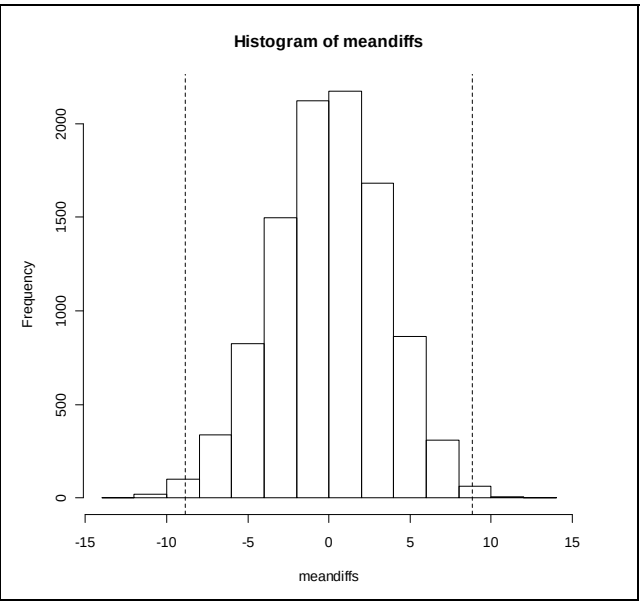
```
> rearrests
             Juvenile court
Adult court   Rearrest No rearrest
  Rearrest         158         515
  No rearrest      290        1134
> mcnemar.test(rearrests, correct=TRUE)

        McNemar's Chi-squared test with continuity correction

data:  rearrests
McNemar's chi-squared = 62.3304, df = 1, p-value = 2.904e-15

> binom.test(rearrests[2],n=sum(rearrests[c(2,3)]))
        Exact binomial test

data:  x[2] and sum(x[c(2, 3)])
number of successes = 290, number of trials = 805, p-value = 1.918e-15
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.3270278 0.3944969
sample estimates:
probability of success
             0.3602484
```

Randomization test (I), histogram, **p-value** and **confidence interval**:

```
> library(HSAUR)
> data(roomwidth)
> dim(roomwidth)
[1] 113   2
> roomwidth[1:3,]
    unit width
1 metres     8
2 metres     9
3 metres    10
> convert <-
ifelse(roomwidth$unit=="feet",1,3.28)
> feet <- roomwidth$unit == "feet"
> metre <- !feet
> y <- roomwidth$width*convert
> T <- mean(y[feet]) - mean(y[metre]) # test
statistic
> T
[1] -8.858893

> greater <- abs(meandiffs) > abs(T)
> mean(greater)
[1] 0.00950095
> binom.test(sum(greater),
length(greater))$conf.int
```

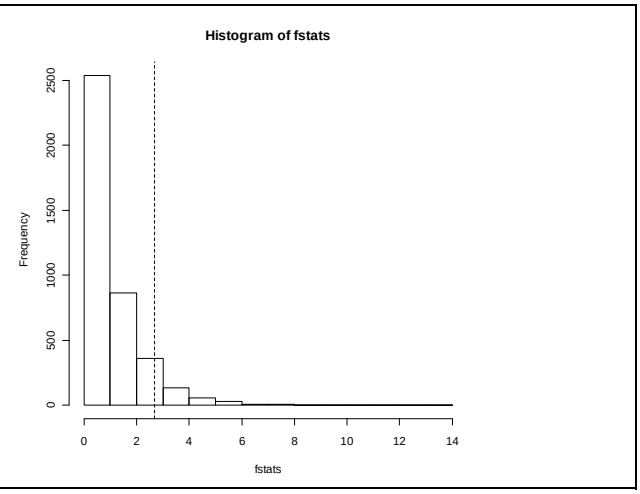
```
> meandiffs <- double(9999)
> for (i in 1:length(meandiffs)) {
+ sy <- sample(y); meandiffs[i] <-
mean(sy[feet]) - mean(sy[metre])}
> hist(meandiffs)
> abline(v=T, lty=2); abline(v=-T, lty=2)
```

```
[1] 0.00950446 0.01378945
attr(,"conf.level")
[1] 0.95
```



Randomization test (II), histogram, **p-value**:

```
> library(faraway)
> data(savings)
> dim(savings)
[1] 50  5
> savings[1:3,]
            sr pop15 pop75     dpi ddpi
Australia 11.43 29.35  2.87 2329.68 2.87
Austria   12.07 23.32  4.41 1507.99 3.93
Belgium   13.17 23.80  4.43 2108.47 3.82
> fstats <- numeric(4000)
> for (i in 1:4000) {
+ ge <- lm(sample(sr) ~ pop75+dpi,
data=savings)
+ fstats[i] <- summary(ge)$fstat[1] }
> length(fstats[fstats >2.6796])/4000
[1] 0.084
> hist(fstats)
> abline(v=2.6796, lty=2)
```



Tally:

```
> library(HSAUR)
> data(Lanza)
> dim(Lanza)
[1] 198   3
> Lanza[1:3,]
  study   treatment classification
1     I Misoprostol              1
2     I Misoprostol              1
3     I Misoprostol              1
> xtabs(~treatment+classification+study, data=Lanza)
, , study = I

             classification
treatment      1  2  3  4  5
  Misoprostol 21  2  4  2  0
  Placebo      2  2  4  9 13
.........
```

Converting to a **table**:

```
> anomalies <- c(235,23,3,0,41,35,8,0,20,11,11,1,2,1,3,1)
> anomalies <- as.table(matrix(anomalies, ncol=4, dimnames=list(MD=0:3, RA=0:3)))
> anomalies
   RA
MD    0   1   2   3
  0 235  41  20   2
  1  23  35  11   1
```

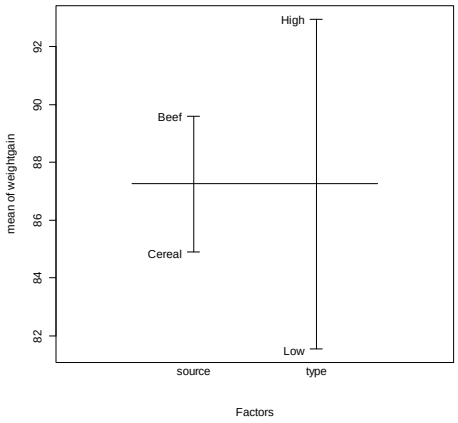
```
  2   3   8  11   3
  3   0   0   1   1
> chisq.test(anomalies)

        Pearson's Chi-squared test

data:  anomalies
X-squared = 145.6554, df = 9, p-value < 2.2e-16

Warning message:
In chisq.test(anomalies) : Chi-squared approximation may be incorrect
```

$(X^TX)^{-1}(X^TY)$:

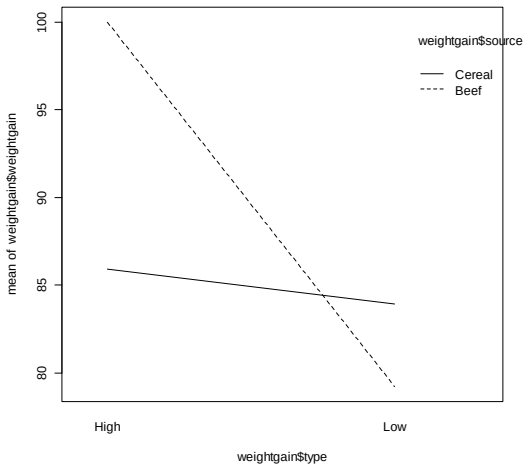
```
> library(faraway)
> data(gala)
> x <- model.matrix(~Area+Elevation+Nearest+Scruz+Adjacent, gala)
> y <- gala$Species
> solve(crossprod(x,x), crossprod(x,y))
```

Two-way data, **summary**, **plot** & **interaction plot**:

```
> library(HSAUR)
> data(weightgain)
> dim(weightgain)
[1] 40  3
> weightgain[1:3,]
  source type weightgain
1   Beef  Low         90
2   Beef  Low         76
3   Beef  Low         90
> tapply(weightgain$weightgain, list(weightgain$source, weightgain$type), mean)
        High  Low
Beef   100.0 79.2
Cereal  85.9 83.9
> wg_aov <- aov(weightgain~source*type, data=weightgain)
> plot.design(weightgain)
```

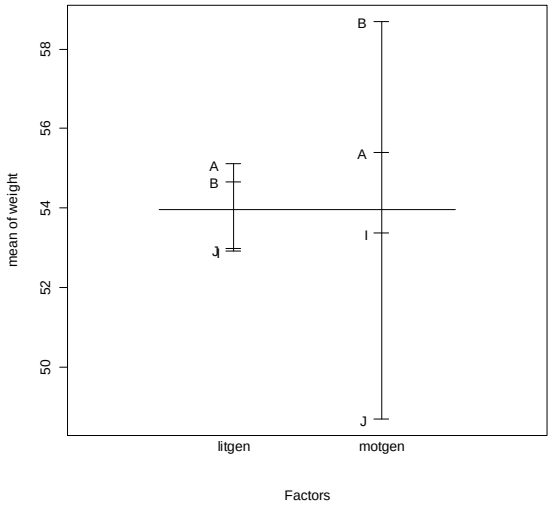



```
> interaction.plot(weightgain$type, weightgain$source, weightgain$weightgain)
```



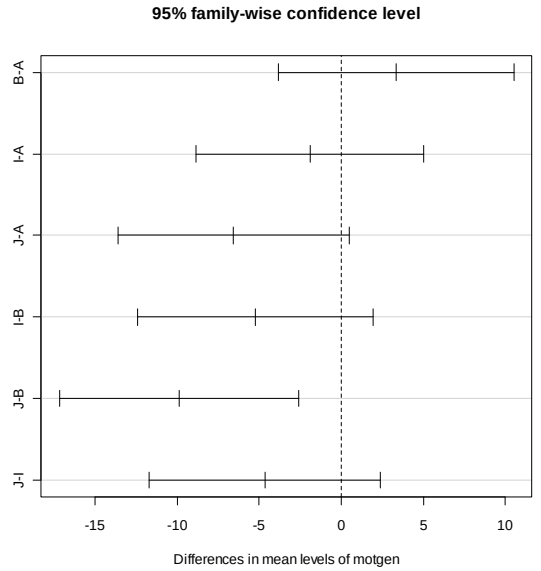
Two-way ANOVA, **Tukey** & **plot**:

```
> library(HSAUR)
> data(foster)
> dim(foster)
[1] 61  3
> foster[1:3,]
  litgen motgen weight
1      A      A   61.5
2      A      A   68.2
3      A      A   64.0
> plot.design(foster)
```




```
> xtabs(~motgen+litgen, data=foster)
      litgen
motgen A B I J
     A 5 4 3 4
     B 3 5 3 3
     I 4 4 5 3
     J 5 2 3 5
> foster_aov <- aov(weight~litgen*motgen, data=foster)
> foster_HSD <- TukeyHSD(foster_aov, "motgen")
> foster_HSD
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = weight ~ litgen * motgen, data = foster)

$motgen
         diff        lwr        upr     p adj
```

```
B-A  3.330369  -3.859729 10.5204672 0.6078581
I-A -1.895574  -8.841869  5.0507207 0.8853702
J-A -6.566168 -13.627285  0.4949498 0.0767540
I-B -5.225943 -12.416041  1.9641552 0.2266493
J-B -9.896537 -17.197624 -2.5954489 0.0040509
J-I -4.670593 -11.731711  2.3905240 0.3035490
> plot(foster_HSD)
```

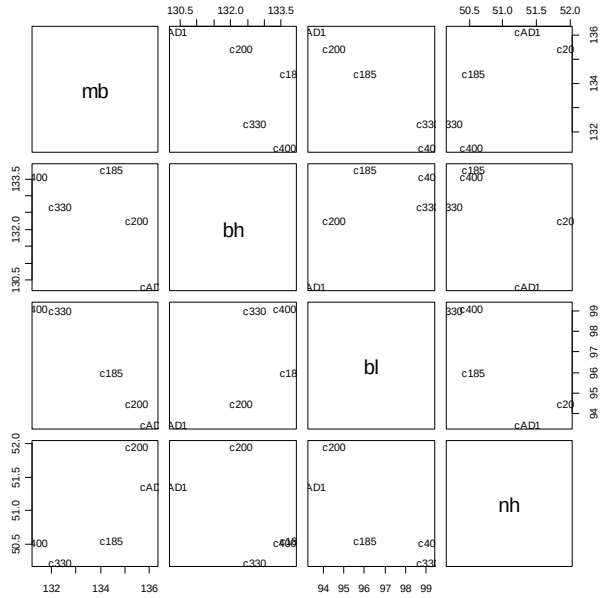


Manova:

```
> library(HSAUR)
> data(water)
> water[1:3,]
  location       town mortality hardness
1    South       Bath      1247      105
2    North Birkenhead      1668       17
3    South Birmingham      1466        5
> summary(manova(cbind(hardness, mortality)~location, data=water), test="Hotelling-Lawley")
          Df Hotelling-Lawley approx F num Df den Df    Pr(>F)
location   1          0.90021   26.106      2     58 8.217e-09 ***
Residuals 59
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Manova data summary:

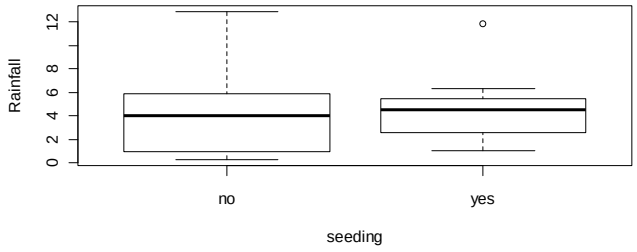
```
> library(HSAUR)
> data(skulls)
> skulls[1:3,]
    epoch  mb  bh bl nh
1 c4000BC 131 138 89 49
2 c4000BC 125 131 92 48
3 c4000BC 131 132 99 50
> means <- aggregate(skulls[,c("mb","bh","bl","nh")],list(epoch=skulls$epoch),mean)
> means
    epoch       mb       bh       bl       nh
1 c4000BC 131.3667 133.6000 99.16667 50.53333
2 c3300BC 132.3667 132.7000 99.06667 50.23333
3 c1850BC 134.4667 133.8000 96.03333 50.56667
4  c200BC 135.5000 132.3000 94.53333 51.96667
5  cAD150 136.1667 130.3333 93.50000 51.36667
> skulls_manova <- manova(cbind(mb,bh,bl,nh)~epoch, data=skulls)
> summary(skulls_manova, test="Pillai")
           Df  Pillai approx F num Df den Df    Pr(>F)
epoch       4 0.35331    3.512     16    580 4.675e-06 ***
Residuals 145
---

> pairs(means[,-1], panel=function(x,y){text(x,y,abbreviate(levels(skulls$epoch)))})
```

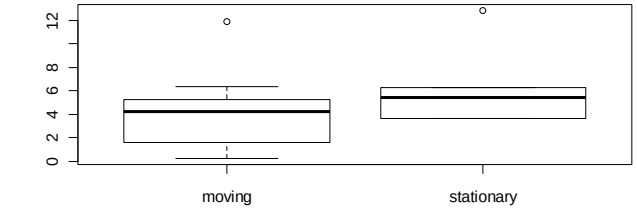



```
> summary(manova(cbind(mb,bh,bl,nh)~epoch, data=skulls, subset=epoch %in% c("c4000BC",
"c3300BC")))
          Df   Pillai approx F num Df den Df Pr(>F)
epoch      1 0.027674  0.39135      4     55  0.814
Residuals 58
```

Boxplot & identifying **outliers**:

```
> library(HSAUR)
> data(clouds)
> clouds[1:3,]
  seeding time  sne cloudcover prewetness echomotion rainfall
1      no    0 1.75       13.4      0.274 stationary    12.85
2     yes    1 2.70       37.9      1.267     moving     5.52
3     yes    3 4.10        3.9      0.198 stationary     6.29
> layout(matrix(1:2, nrow=2))
> bxspeeding <- boxplot(rainfall~seeding, data=clouds, ylab="Rainfall", xlab="seeding")
> bxpecho <- boxplot(rainfall~echomotion, data=clouds)
> rownames(clouds)[clouds$rainfall %in% c(bxspeeding$out, bxpecho$out)]
[1] "1"  "15"
```



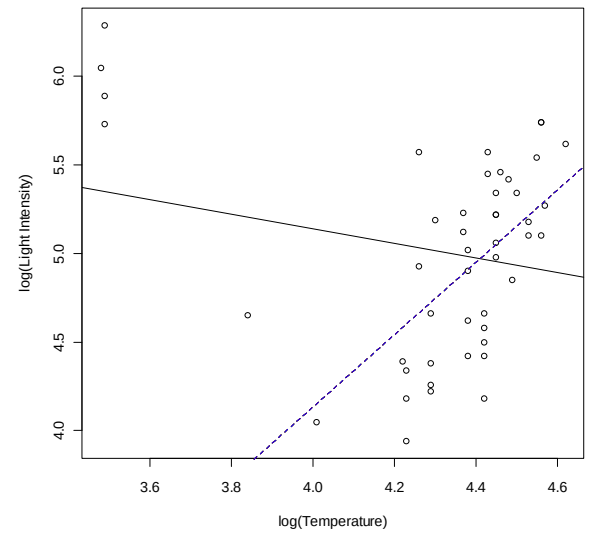


Regression diagnostics:

```
> data(clouds)
> layout(matrix(1:4,nrow=2))
> plot(rainfall~time, data=clouds)
> clouds_formula <- rainfall~seeding*(sne+cloudcover+prewetness+echomotion)+time
> clouds_lm <- lm(clouds_formula, data=clouds)
> clouds_resid <- residuals(clouds_lm)
> clouds_fitted <- fitted(clouds_lm)

> psymb <- as.numeric(clouds$seeding)
> plot(rainfall~sne, data=clouds, pch=psymb)
> abline(lm(rainfall~sne, data=clouds, subset=seeding =="no"))
> abline(lm(rainfall~sne, data=clouds, subset=seeding =="yes"), lty=2)
> legend("topright", legend=c("No seeding", "Seeding"), pch=1:2, lty=1:2, bty="n")

> plot(clouds_fitted, clouds_resid, type="n", ylim=max(abs(clouds_resid))*c(-1,1))
> abline(h=0, lty=2)
> text(clouds_fitted, clouds_resid, labels=rownames(clouds))

> Plot(clouds_lm)
```

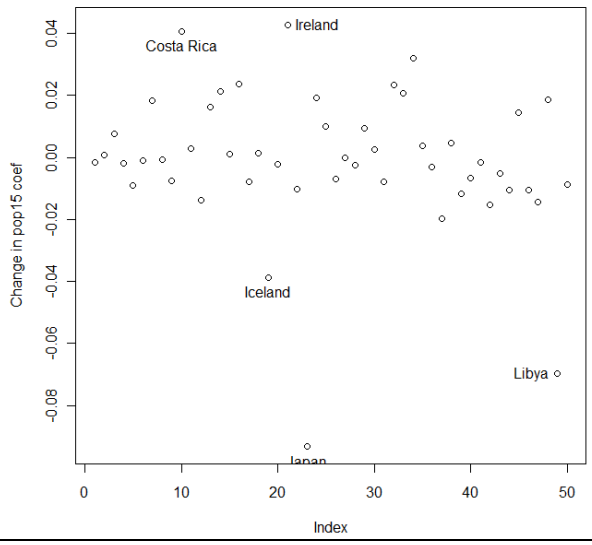
Regression with "subset" function:

```
> library(faraway)
> data(star)
>
plot(star$temp,star$light,xlab="log(Temperature
)",ylab="log(Light Intensity)")
> model1 <- lm(light~temp,star)
> abline(model1)
> model2 <- lm(light~temp,star,
subset=(temp>3.6))
> abline(model2, lty=2, col=4)
```



Identify influential data points:

```
> library(faraway)
> data(savings)
> model1 <- lm(sr ~ pop15+pop75+dpi+ddpi,
savings)
> cook <- cooks.distance(model1)
> model2 <- lm(sr ~ pop15+pop75+dpi+ddpi,
savings,subset=(cook<max(cook)))
> #to remove one observation with the largest
cook's distance
> model1inf <- influence(model1)
> plot(model1inf$coef[,2],ylab="Change in pop15
coef")
> countries <- rownames(savings)
> identify(1:50,model1inf$coef[,2],countries)

> #hit Esc key to terminate
```

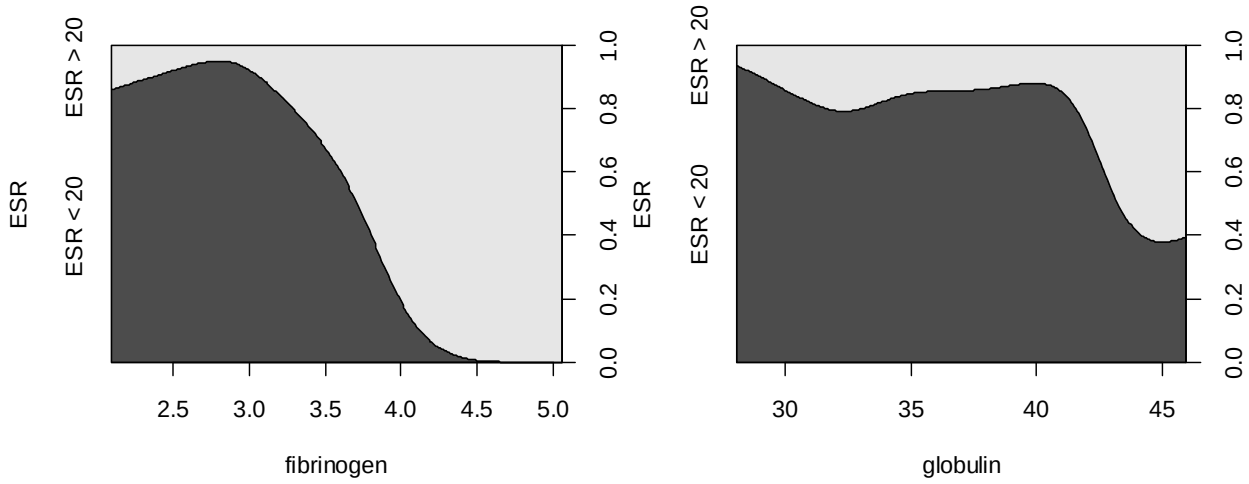


Probability & **quantile** of t-distribution:

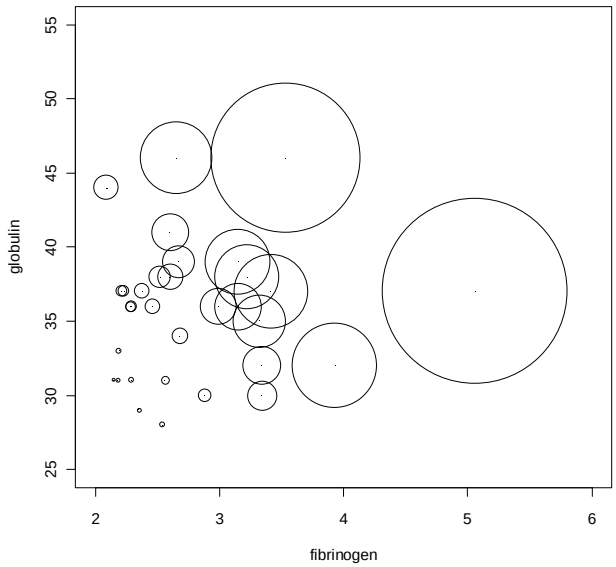
```
> pt(2.05, 16)
[1] 0.9714386
> pt(2.05, 16, lower.tail=F)
[1] 0.02856145
> qt(0.05, 13)
[1] -1.770933
> qf(0.01,2,7)
[1] 0.01006478
> qf(0.01,2,7, lower.tail=F)
[1] 9.546578
```

Logistic regression and **bubble** plot:

```
> library(HSAUR)
> data(plasma)
> plasma[1:3,]
  fibrinogen globulin      ESR
1       2.52       38 ESR < 20
2       2.56       31 ESR < 20
3       2.19       33 ESR < 20
> plasma_glm_1 <- glm(ESR ~ fibrinogen, data=plasma, family=binomial())
> confint(plasma_glm_1, parm="fibrinogen")
Waiting for profiling to be done...
    2.5 %    97.5 %
0.3387619 3.9984921
> exp(coef(plasma_glm_1)["fibrinogen"])
fibrinogen
  6.215715
> exp(confint(plasma_glm_1, parm="fibrinogen"))
Waiting for profiling to be done...
    2.5 %    97.5 %
 1.403209 54.515884

> layout(matrix(1:2, ncol=2))
> cdplot(ESR~fibrinogen, data=plasma)
> cdplot(ESR~globulin, data=plasma)
```




```
> plasma_glm_2 <- glm(ESR ~ fibrinogen+globulin, data=plasma, family=binomial())
> anova(plasma_glm_1, plasma_glm_2, test="Chisq")
Analysis of Deviance Table

Model 1: ESR ~ fibrinogen
Model 2: ESR ~ fibrinogen + globulin
  Resid. Df Resid. Dev Df Deviance P(>|Chi|)
1        30     24.840
2        29     22.971  1   1.8692    0.1716

> prob <- predict(plasma_glm_2,type="response")
> plot(globulin~fibrinogen, data=plasma, xlim=c(2,6), ylim=c(25,55), pch=".")
> symbols(plasma$fibrinogen, plasma$globulin, circles=prob, add=TRUE)
> plot(globulin~fibrinogen, data=plasma, xlim=c(2,6), ylim=c(25,55), pch=".")
> symbols(plasma$fibrinogen, plasma$globulin, circles=prob, add=TRUE)
```



anova table based on Type III SS (R's default is by Type I SS):

```
> library(faraway)
> data(sexab)
> dim(sexab)
[1] 76  3
> sexab[1:3,]
       cpa     ptsd    csa
1  2.04786  9.71365 Abused
2  0.83895  6.16933 Abused
3 -0.24139 15.15926 Abused
> table(sexab$csa)

   Abused NotAbused
       45        31
> model1 <- lm(ptsd ~ csa*cpa,sexab)
> model1 <- lm(ptsd ~ cpa*csa,sexab)
> anova(model1)                       #uses Type I SS
Analysis of Variance Table

Response: ptsd
          Df Sum Sq Mean Sq F value    Pr(>F)
cpa        1 449.80  449.80  41.827 1.044e-08 ***
csa        1 624.03  624.03  58.028 7.917e-11 ***
cpa:csa    1   7.81    7.81   0.726     0.397
Residuals 72 774.28   10.75
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> library(car)
> Anova(model1, type=c("III"))        #uses Type III SS
Anova Table (Type III tests)

Response: ptsd
             Sum Sq Df  F value    Pr(>F)
(Intercept) 1843.79  1 171.4545 < 2.2e-16 ***
cpa           50.11  1   4.6595   0.03422 *
csa          438.28  1  40.7559 1.479e-08 ***
cpa:csa        7.81  1   0.7260   0.39702
Residuals    774.28 72
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

OR

```
> model1 <- lm(ptsd ~ csa*cpa,sexab)
> drop1(model1,~.,test="F")           #another way
Single term deletions

Model:
```

```
ptsd ~ csa * cpa
        Df Sum of Sq     RSS    AIC F value     Pr(F)
<none>                774.28 184.41
csa      1    438.28 1212.56 216.50 40.7559 1.479e-08 ***
cpa      1     50.11  824.38 187.18  4.6595   0.03422 *
csa:cpa  1      7.81  782.08 183.17  0.7260   0.39702
---
```

Linear Model Formula:

$y \sim x1 + x2$

Formula	Meaning
x1+x2	x1 + x2
.~. – x2	without x2 from previous model (used with "update" function)
x1 : x2	x1×x2
x1 * x2	x1 + x2 + x1×x2
A / B	B is nested within A
A | B	A for given B
(A + B + C)^2	all the main effect terms and all 2-way interactions
Error(A/B)	specifies proper error term
I(x^2)	include such a term
-1	without y-intercept term

```
> data1 <- read.csv("D:\\Stat333\\hw11-1.csv")
> data1[1:3,]
      X     XS     Y
1 0.196 -0.004 31.54
2 0.924  0.724 31.98
3 0.249  0.049 30.78
> model1 <- lm(Y ~ X + I(X^2) + I(X^3), data=data1)
> summary(model1)
> anova(model1)
> model2 <- update(model1, ~.-1, data=data1)
> summary(model2)
```

Repeated Measures anova (I):

```
> groceries <- read.table("http://ww2.coastal.edu/kingw/statistics/R-
tutorials/text/groceries.txt", header=T)
> dim(groceries)
[1] 10  5
> groceries
             subject storeA storeB storeC storeD
1            lettuce   1.17   1.78   1.29   1.29
2           potatoes   1.77   1.98   1.99   1.99
3               milk   1.49   1.69   1.79   1.59
4               eggs   0.65   0.99   0.69   1.09
5              bread   1.58   1.70   1.89   1.89
6             cereal   3.13   3.15   2.99   3.09
7        ground.beef   2.09   1.88   2.09   2.49
8        tomato.soup   0.62   0.65   0.65   0.69
9  laundry.detergent   5.89   5.99   5.99   6.99
10           aspirin   4.46   4.84   4.99   5.15
> groceries2 <- stack(groceries)
> subject <- rep(groceries$subject,4)
> groceries2[3] <- subject
> rm(subject)
> colnames(groceries2) <- c("price", "store", "subject")
> groceries2[1:3,]
  price  store  subject
1  1.17 storeA  lettuce
2  1.77 storeA potatoes
3  1.49 storeA     milk
> model1 <- aov(price ~ store + Error(subject/store), data=groceries2)
> summary(model1)

Error: subject
          Df Sum Sq Mean Sq F value Pr(>F)
Residuals  9 115.19  12.799

Error: subject:store
          Df  Sum Sq  Mean Sq F value  Pr(>F)
```

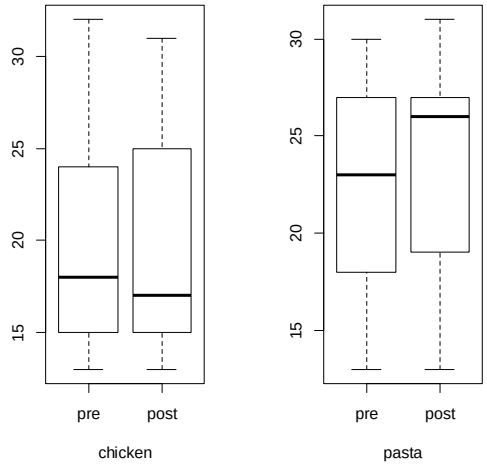
```
store      3 0.58586 0.195287  4.3442 0.01273 *
Residuals 27 1.21374 0.044953
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> with(groceries2,pairwise.t.test(price,store,p.adjust.method="holm",paired=T))

        Pairwise comparisons using paired t tests

data:  price and store

       storeA storeB storeC
storeB 0.17   -      -
storeC 0.17   0.69   -
storeD 0.07   0.49   0.33

P value adjustment method: holm
```

Repeated Measures anova (II):

```
> data1
   subject    diet test SSS
1        A chicken  pre  18
2        B chicken  pre  13
3        C chicken  pre  18
4        D chicken  pre  15
5        E chicken  pre  22
6        F chicken  pre  32
7        G chicken  pre  31
8        H chicken  pre  24
9        I chicken  pre  15
10       A chicken post  15
11       B chicken post  13
12       C chicken post  17
13       D chicken post  15
14       E chicken post  24
15       F chicken post  31
16       G chicken post  31
17       H chicken post  25
18       I chicken post  17
19       J   pasta  pre  17
20       K   pasta  pre  30
21       L   pasta  pre  18
22       M   pasta  pre  13
23       N   pasta  pre  23
24       O   pasta  pre  27
25       P   pasta  pre  27
26       Q   pasta  pre  24
27       R   pasta  pre  23
28       J   pasta post  19
29       K   pasta post  31
30       L   pasta post  18
31       M   pasta post  13
32       N   pasta post  24
33       O   pasta post  27
34       P   pasta post  26
35       Q   pasta post  28
36       R   pasta post  26
```

```
attach(data1)
model1 <- aov(SSS~diet*test+Error(subject/test))
summary(model1)

Error: subject
          Df Sum Sq Mean Sq F value Pr(>F)
diet       1   40.1   40.11   0.509  0.486
Residuals 16 1259.8   78.74

Error: subject:test
          Df Sum Sq Mean Sq F value Pr(>F)
test       1  2.778   2.778   2.174   0.16
diet:test  1  2.778   2.778   2.174   0.16
Residuals 16 20.444   1.278
library(nlme)
model2 <- lme(SSS~test+diet,random=~1|subject)
summary(model2)
Linear mixed-effects model fit by REML
 Data: NULL
       AIC      BIC    logLik
  186.7878 194.2703 -88.39389

Random effects:
 Formula: ~1 | subject
        (Intercept) Residual
StdDev:    6.219726 1.168766

Fixed effects: SSS ~ test + diet
                Value Std.Error DF   t-value p-value
(Intercept) 21.166667 2.1005158 17 10.076890  0.0000
testpre     -0.555556 0.3895888 17 -1.426005  0.1720
dietpasta    2.111111 2.9577768 16  0.713749  0.4857
       #0.713749²=0.5094376, same as F value in AOV
 Correlation:
          (Intr) testpr
testpre   -0.093
dietpasta -0.704  0.000

Number of Observations: 36
Number of Groups: 18
library(lme4)
model3 <- lmer(SSS~test+diet+(1|subject))
summary(model3)
Linear mixed model fit by REML ['lmerMod']
Formula: SSS ~ test + diet + (1 | subject)

REML criterion at convergence: 176.7878

Random effects:
 Groups   Name        Variance Std.Dev.
 subject  (Intercept) 38.685   6.220
 Residual              1.366   1.169
Number of obs: 36, groups: subject, 18

Fixed effects:
            Estimate Std. Error t value
(Intercept)  21.1667     2.1005  10.077
```

```
testpre      -0.5556     0.3896  -1.426
dietpasta     2.1111     2.9578   0.714

Correlation of Fixed Effects:
          (Intr) testpr
testpre   -0.093
dietpasta -0.704  0.000

# To show there is no significant difference
> test=factor(test,levels=c("pre","post"))
> table(test)
test
 pre post
  18   18
> tapply(SSS,list(diet,test),mean)
            pre     post
chicken 20.88889 20.88889
pasta   22.44444 23.55556
> par(mfrow=c(1,2))
>
boxplot(SSS[diet=="chicken"]~test[diet=="chicken"],xlab="chicken")
> boxplot(SSS[diet=="pasta"]~test[diet=="pasta"],xlab="pasta")
```




```
>
```