- R MAY BE USED AS AN ORDINARY CALCULATOR

```
> 2 + 3 * 5
[1] 17
> log (10)
[1] 2.302585
> 4^2
[1] 16
> 3/2
[1] 1.5
> sqrt (16)
[1] 4
> abs (3-7)
[1] 4
>  pi
[1] 3.141593
>  exp(2)
[1] 7.389056
> 15 %/% 4
[1] 3
>
> # ASSIGNMENT OPERATOR
>
> x<- log(2.843432) *pi
> x
[1] 3.283001
> sqrt(x)
[1] 1.811905
> floor(x)        # largest integer less than or equal to x
[1] 3
> ceiling(x)      # smallest integer greater than or equal to x
```

- VECTORS

```
> x<-c(1,3,2,10,5)    #create a vector x with 5 components
> y<-1:5              #create a vector of consecutive integers
> x
[1]  1  3  2 10  5
> y
[1] 1 2 3 4 5
> y+2                 #scalar addition
[1] 3 4 5 6 7
> 2*y                 #scalar multiplication
[1]  2  4  6  8 10
> y^2                 #raise each component to the second power
[1]  1  4  9 16 25
```

```
> 2^y                 #raise 2 to the first through fifth power
[1]  2  4  8 16 32
> y                   #y itself has not been unchanged
[1] 1 2 3 4 5
> y<-y*2
> y                   #it is now changed
[1]  2  4  6  8 10
> x<-c(1,3,2,10,5); y<-1:5 #two or more statements are separated by semicolons
> y
[1] 1 2 3 4 5
> x+y
[1]  2  5  5 14 10
> x*y
[1]  1  6  6 40 25
> x/y
[1] 1.0000000 1.5000000 0.6666667 2.5000000 1.0000000
> x^y
[1]     1     9     8 10000  3125
> sum(x)              #sum of elements in x
[1] 21
> cumsum(x)           #cumulative sum vector
[1]  1  4  6 16 21
> diff(x)             # first difference
[1]  2 -1  8 -5
> diff(x,2)           #second difference
[1] 1 7 3
> max(x)              #maximum
[1] 10
>  min(x)             #minimum
[1] 1
> x
[1]  1  3  2 10  5
> sort(x)                # increasing order
[1]  1  2  3  5 10
> sort(x, decreasing=T)  # decreasing order
[1] 10  5  3  2  1
>
> # COMPONENT EXTRACTION
>
> x
[1]  1  3  2 10  5
> length(x)           # number of elements in x
[1] 5
> x[3]                # the third element of x
[1] 2
```

```
> x[3:5]              # the third to fifth element of x, inclusive
[1]  2 10  5
> x[-2]               # all except the second element
[1]  1  2 10  5
> x[x>3]              # list of elements in x greater than 3
[1] 10  5
>
> # LOGICAL VECTOR
>
> x>3
[1] FALSE FALSE FALSE  TRUE  TRUE
> as.numeric(x>3)     # as.numeric() function coerces logical components to numeric
[1] 0 0 0 1 1
> sum(x>3)            # number of elements in x greater than 3
[1] 2
> (1:length(x))[x<=2] # indices of x whose components are less than or equal to 2
[1] 1 3
> z<-as.logical(c(1,0,0,1)) # numeric to logical vector conversion
> z
[1]  TRUE FALSE FALSE  TRUE
>
> # CHARACTER VECTOR
>
> colors<-c("green", "blue", "orange", "yellow", "red")
> colors
[1] "green"  "blue"   "orange" "yellow" "red"
>
> # VECTOR COMPONENTS CAN BE NAMED
>
> names(x)            # check if any names are attached to x
NULL
> names(x)<-colors    # assign the names using the character vector colors
> x
 green   blue orange yellow    red
     1      3      2     10      5
> x["green"]          # component reference by its name
green
    1
> names(x)<-NULL      # names can be removed by assigning NULL
> x
[1]  1  3  2 10  5
>
> # CONSTRUCT VECTORS WITH PATTERNS
>
> seq(10)
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
> seq(0,1,length=10)
 [1] 0.0000000 0.1111111 0.2222222 0.3333333 0.4444444 0.5555556 0.6666667
 [8] 0.7777778 0.8888889 1.0000000
> seq(0,20,length=5)
[1]  0  5 10 15 20
> seq(0,1,by=0.1)
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
> rep(1,3)
[1] 1 1 1
> c(rep(1,3),rep(2,2),rep(-1,4))
[1]  1  1  1  2  2 -1 -1 -1 -1
> rep("Small",3)
[1] "Small" "Small" "Small"
> c(rep("Small",3),rep("Medium",4))
[1] "Small"  "Small"  "Small"  "Medium" "Medium" "Medium" "Medium"
> rep(c("Low","High"),3)
[1] "Low"  "High" "Low"  "High" "Low"  "High"
> q()
```

- MATRICES

```
> x<-c(1,3,2,10,5)    #create a vector x with 5 components
> y<-1:5              #create a vector of consecutive integers
> x
[1]  1  3  2 10  5
> y
[1] 1 2 3 4 5
> m1<-cbind(x,y)    #Column bind
> m1
      x y
[1,]  1 1
[2,]  3 2
[3,]  2 3
[4,] 10 4
[5,]  5 5
> t(m1)                # transpose of m1
  [,1] [,2] [,3] [,4] [,5]
x    1    3    2   10    5
y    1    2    3    4    5
> dim(m1)
[1] 5 2
> m1<-rbind(x,y)       # rbind() is for row bind and equivalent to t(cbind()).
> m1
```

```
  [,1] [,2] [,3] [,4] [,5]
x    1    3    2   10    5
y    1    2    3    4    5
>
> # LIST ELEMENTS AND SPECIFY MATRIX DIRECTLY (WITHOUT BINDING)
>
> m2<-matrix(c(1,3,2,5,-1,2,2,3,9),nrow=3)
> m2
     [,1] [,2] [,3]
[1,]    1    5    2
[2,]    3   -1    3
[3,]    2    2    9
>
> m2<-matrix(c(1,3,2,5,-1,2,2,3,9),ncol=3,byrow=T)    # to fill row-wise
> m2
     [,1] [,2] [,3]
[1,]    1    3    2
[2,]    5   -1    2
[3,]    2    3    9
>
> m2[2,3]            #element of m2 at the second row, third column
[1] 2
> m2[2,]             #second row
[1]  5 -1  2
> m2[,3]             #third column
[1] 2 2 9
> m2[-1,]            #submatrix of m2 without the first row
     [,1] [,2] [,3]
[1,]    5   -1    2
[2,]    2    3    9
> m2[,-1]            #without the first column
     [,1] [,2]
[1,]    3    2
[2,]   -1    2
[3,]    3    9
> m2[-1,-1]          #submatrix of m2 with the first row and column removed
     [,1] [,2]
[1,]   -1    2
[2,]    3    9
>
> # COMPONENT-WISE COMPUTATIONS
>
> m1<-matrix(1:4, ncol=2); m2<-matrix(c(10,20,30,40),ncol=2)
> m1
     [,1] [,2]
```

```
[1,]    1    3
[2,]    2    4
> m2
     [,1] [,2]
[1,]   10   30
[2,]   20   40
> 2*m1                # scalar multiplication
     [,1] [,2]
[1,]    2    6
[2,]    4    8
> m1+m2               # matrix addition
     [,1] [,2]
[1,]   11   33
[2,]   22   44
> m1*m2               # component-wise multiplication
     [,1] [,2]
[1,]   10   90
[2,]   40  160
>
> # MATRIX OPERATIONS
>
> m1 %*% m2           # the usual matrix multiplication
     [,1] [,2]
[1,]   70  150
[2,]  100  220
> solve(m1)            #inverse matrix of m1
     [,1] [,2]
[1,]   -2  1.5
[2,]    1 -0.5
> solve(m1)%*%m1       #check
     [,1] [,2]
[1,]    1    0
[2,]    0    1
> diag(3)              #diag() is used to construct a k by k identity matrix
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
> diag(c(2,3,3))       #as well as other diagonal matrices
     [,1] [,2] [,3]
[1,]    2    0    0
[2,]    0    3    0
[3,]    0    0    3
>
> # BE CAREFUL.  R tries to read minds.
```

```
> diag(m2)
[1] 10 40
>
> eigen(m2)           # eigenvalues and eigenvectors of m2
$values
[1] 53.722813 -3.722813

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

> ev <- eigen(m2)     # ev is a variable with two components
> ev
$values
[1] 53.722813 -3.722813

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

> ev$values           # to extract components, use $
[1] 53.722813 -3.722813
> ev$vectors
           [,1]       [,2]
[1,] -0.5657675 -0.9093767
[2,] -0.8245648  0.4159736
> attach(ev)             # to keep from typing ev
> values
[1] 53.722813 -3.722813
> vectors
           [,1]       [,2]
[1,] -0.5657675 -0.9093767
[2,] -0.8245648  0.4159736
```

- DATA INPUT

```
Using read-table to read files:

*****************************************************************
If the data was in a file called freeformat.dat, read this data with the command:
      data <- read.table("freeformat.dat", header=TRUE)
```

```
*********************************************************************
Fixed-format data can also be read with the read.table command. Using the above data,
      data2 <- read.table("fixeddata.dat",
      col.names=c("country", "ud", "sd"),
      sep = c(1,7,11))

*********************************************************************
Here is the command syntax to read in comma-delimited data.
      data <- read.table("delimit.dat", sep = ",")

*********************************************************************
Read from webpage:
      data <- read.table('http://www.stat.wmich.edu/naranjo/stat4640/data/Grades.dat',
      header=F)

*********************************************************************
It is often helpful to cut-and-paste data onto console:

> sbpdat<-read.table(stdin(), header=F)
                        # You will see the prompt 0:
                        # You can  now paste data onto the console
                        # (I pasted 3 columns of data in the following example)
0: 1  144 39
1: 2  220 47
2: 3  138 45
3: 4  145 47
4: 5  162 65
6: 6  142 46
7: 7  170 67
8: 8  124 42
9: 9  158 67
10: 10 154 56
11: 11 162 64
12: 12 150 56
13: 13 140 59
14: 14 110 34
15: 15 128 42
16: 16 130 48
17: 17 135 45
18: 18 114 17
19: 19 116 20
20: 20 124 19
21: 21 136 36
22: 22 142 50
23: 23 120 39
```

```
24: 24 120 21
25: 25 160 44
26: 26 158 53
27: 27 144 63
28: 28 130 29
29: 29 125 25
30: 30 175 69
31:                       # Press <Enter> to quit input mode

> summary(sbpdat)
       V1              V2              V3
 Min.   : 1.00   Min.   :110.0   Min.   :17.00
 1st Qu.: 8.25   1st Qu.:125.8   1st Qu.:36.75
 Median :15.50   Median :141.0   Median :45.50
 Mean   :15.50   Mean   :142.5   Mean   :45.13
 3rd Qu.:22.75   3rd Qu.:157.0   3rd Qu.:56.00
 Max.   :30.00   Max.   :220.0   Max.   :69.00

> names(sbpdat)<-cbind("Index", "SBP", "Age")

> print(sbpdat)

> mean(sbpdat)
    Index       SBP       Age
 15.50000 142.53333  45.13333

> sd(sbpdat)
    Index       SBP       Age
 8.803408 22.581245 15.294203

> summary(sbpdat)
     Index            SBP             Age
 Min.   : 1.00   Min.   :110.0   Min.   :17.00
 1st Qu.: 8.25   1st Qu.:125.8   1st Qu.:36.75
 Median :15.50   Median :141.0   Median :45.50
 Mean   :15.50   Mean   :142.5   Mean   :45.13
 3rd Qu.:22.75   3rd Qu.:157.0   3rd Qu.:56.00
 Max.   :30.00   Max.   :220.0   Max.   :69.00
```

- SUMMARY STATS

```
> grades<-read.table('http://www.stat.wmich.edu/naranjo/stat4640/data/Grades.dat',
           header=F)
>
```

```
> ls()          # list objects
[1] "colors" "ev"     "grades" "m1"     "m2"     "x"      "y"      "z"
>
> # assign variable names
> names(grades)<-c("ID","Sex", "Class", "Quizzes", "Exam1", "Exam2", "Lab", "Final")
> attributes(grades)
$names
[1] "ID"      "Sex"     "Class"   "Quizzes" "Exam1"   "Exam2"   "Lab"
[8] "Final"

$class
[1] "data.frame"

$row.names
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
[26] 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49

> summary(grades)        # desciptive statistics
       ID     Sex        Class          Quizzes          Exam1
 air    : 1   f:17   Min.   :2.000   Min.   :20.00   Min.   : 33.00
 aln    : 1   m:32   1st Qu.:3.000   1st Qu.:41.00   1st Qu.: 71.00
 bag    : 1          Median :3.000   Median :44.00   Median : 86.00
 bam    : 1          Mean   :3.449   Mean   :42.84   Mean   : 80.53
 bec    : 1          3rd Qu.:4.000   3rd Qu.:47.00   3rd Qu.: 95.00
 bej    : 1          Max.   :4.000   Max.   :50.00   Max.   :100.00
 (Other):43
     Exam2             Lab             Final
 Min.   : 51.00   Min.   : 63.00   Min.   : 74.0
 1st Qu.: 86.00   1st Qu.: 92.00   1st Qu.:124.0
 Median : 93.00   Median : 95.00   Median :147.0
 Mean   : 90.57   Mean   : 93.53   Mean   :139.5
 3rd Qu.: 98.00   3rd Qu.: 98.00   3rd Qu.:161.0
 Max.   :100.00   Max.   :100.00   Max.   :191.0

> summary(grades$Quizzes)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  20.00   41.00   44.00   42.84   47.00   50.00

> summary(grades$Sex)
 f  m
17 32

> mean(grades)
        ID        Sex      Class    Quizzes      Exam1      Exam2        Lab       Final
        NA         NA   3.448980  42.836735  80.530612  90.571429  93.530612  139.510204
```

```
> sd(grades)
        ID        Sex      Class    Quizzes      Exam1      Exam2        Lab       Final
        NA         NA  0.5795553  6.5617164 17.5856263 10.1837289  6.8132898  27.7377018

> attach(grades)

> table(Sex)          # frequency table
Sex
 f  m
17 32

> table(Sex,Class)    # two-way frequency table
   Class
Sex  2  3  4
  f  0  7 10
  m  2 16 14

> grades[1:5,]        # print subset (same as matrix)

   ID Sex Class Quizzes Exam1 Exam2 Lab Final
1 air   f     4      50    93    93  98   162
2 aln   m     4      49    95    98  97   175
3 bam   m     4      39    63    84  95    95
4 bag   f     3      46    92    96  88   150
5 bes   f     4      45   100    98  96   191
```

- BASIC STATISTICAL METHODS

```
> grades<-read.table('http://www.stat.wmich.edu/naranjo/stat4640/data/Grades.dat',
           header=F)
> names(grades)<-c("ID","Sex", "Class", "Quizzes", "Exam1", "Exam2", "Lab", "Final")
> grades[1:10,]
    ID Sex Class Quizzes Exam1 Exam2 Lab Final
1  air   f     4      50    93    93  98   162
2  aln   m     4      49    95    98  97   175
3  bam   m     4      39    63    84  95    95
4  bag   f     3      46    92    96  88   150
5  bes   f     4      45   100    98  96   191
6  bec   f     3      44    98   100  85   175
7  bej   m     3      41    86    86  94   138
8  bis   f     4      50   100   100  99   166
9  blc   m     4      50    95    97  96   162
10 boc   f     4      48    71   100  97   143
```

```
>
> summary(grades)
       ID     Sex        Class          Quizzes          Exam1
 air    : 1   f:17   Min.   :2.000   Min.   :20.00   Min.   : 33.00
 aln    : 1   m:32   1st Qu.:3.000   1st Qu.:41.00   1st Qu.: 71.00
 bag    : 1          Median :3.000   Median :44.00   Median : 86.00
 bam    : 1          Mean   :3.449   Mean   :42.84   Mean   : 80.53
 bec    : 1          3rd Qu.:4.000   3rd Qu.:47.00   3rd Qu.: 95.00
 bej    : 1          Max.   :4.000   Max.   :50.00   Max.   :100.00
 (Other):43
     Exam2             Lab             Final
 Min.   : 51.00   Min.   : 63.00   Min.   : 74.0
 1st Qu.: 86.00   1st Qu.: 92.00   1st Qu.:124.0
 Median : 93.00   Median : 95.00   Median :147.0
 Mean   : 90.57   Mean   : 93.53   Mean   :139.5
 3rd Qu.: 98.00   3rd Qu.: 98.00   3rd Qu.:161.0
 Max.   :100.00   Max.   :100.00   Max.   :191.0

> ###########################################################################
> # ONE-SAMPLE SUMMARY STATISTCS
> attach(grades)
> head(Exam1)
[1]  93  95  63  92 100  98
> mean(Exam1)
[1] 80.53061
> sd(Exam1)
[1] 17.58563
> var(Exam1)
[1] 309.2543
>
> # ONE-SAMPLE t-TEST
> # Exam 1 average over all sections is 80.  Is this class comparable?
> ?
>
> t.test(Exam1, mu=80)

        One Sample t-test

data:  Exam1
t = 0.2112, df = 48, p-value = 0.8336
alternative hypothesis: true mean is not equal to 80
95 percent confidence interval:
 75.47943 85.58179
sample estimates:
mean of x
```

```
 80.53061

> ###############################################################################
> # Two-sample test
>
> # Are Exam1 scores for class 3 and class 4 different?
>
> sub3<-Exam1[Class==3]                # Extracts scores for Class=3
> sub3
 [1] 92 98 86 64 44 99 68 97 78 90 76 77 89 79 76 33 95 83 89 94 41 63 95
> sub4<-Exam1[Class==4]
> sub4                                 # Extracts scores for Class=4
 [1]  93  95  63 100 100  95  71  64  92  50  80  86  83  84  87  65  93  96  97
[20]  71  34  96  88  99

> t.test(sub3,sub4,var.equal = T)

        Two Sample t-test

data:  sub3 and sub4
t = -0.7718, df = 45, p-value = 0.4443
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -14.661193   6.538004
sample estimates:
mean of x mean of y
 78.52174  82.58333

> ###################################################################
> Or better, do extraction and analysis in one step

> t.test(Exam1[Class == 3], Exam1[Class == 4])

        Welch Two Sample t-test

data:  Exam1[Class == 3] and Exam1[Class == 4]
t = -0.7703, df = 44.267, p-value = 0.4452
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -14.685746   6.562558
sample estimates:
mean of x mean of y
 78.52174  82.58333

> boxplot(Exam1[Class == 3], Exam1[Class == 4])
```

```
> ############## PAIRED t-TEST ###################
> #### Q: Is Exam2 higher than Exam1?
>
> t.test(Exam1, Exam2, paired=T, alt = "less")

        Paired t-test

data:  Exam1 and Exam2
t = -4.6736, df = 48, p-value = 1.213e-05
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
      -Inf -6.437472
sample estimates:
mean of the differences
              -10.04082

> boxplot(Exam1,Exam2)

>
> ############ REGRESSION AND CORRELATION ################

> cor(Exam1,Exam2)
[1] 0.5215269
> y<-Exam2
> x<-Exam1
> reg.out<-lm(y~x)
> reg.out

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
     66.250        0.302

> summary(reg.out)

Call:
lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q     Max
-35.787  -3.579   1.965   4.649  13.481
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 66.25014    5.93805   11.16 8.32e-15 ***
x            0.30201    0.07207    4.19 0.000122 ***
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 8.781 on 47 degrees of freedom
Multiple R-squared:  0.272,     Adjusted R-squared:  0.2565
F-statistic: 17.56 on 1 and 47 DF,  p-value: 0.0001217

> head(cbind(y,x,reg.out$fitted.values,reg.out$residuals))
    y   x
1  93  93 94.33735 -1.337346
2  98  95 94.94137  3.058628
3  84  63 85.27696 -1.276955
4  96  92 94.03533  1.964667
5  98 100 96.45144  1.548563
6 100  98 95.84741  4.152589
>
> ############ ANOVA (Comparison of means) ################
>
> levels(Class)    # Values found in the Class variable
[1] "2" "3" "4"
>
> table(Class)     # Frequency table (to know sample sizes)
Class
 2  3  4
 2 23 24

> aov(Exam1 ~ Class)
Call:
   aov(formula = Exam1 ~ Class)

Terms:
                   Class Residuals
Sum of Squares    198.632 14645.572
Deg. of Freedom         2        46

Residual standard error: 17.84326
Estimated effects may be unbalanced
>
> # calculate the sample means of the subgroups
>
```

```
> tapply(Exam1, Class, mean)
       2        3        4
79.00000 78.52174 82.58333
>
> ?tapply
>
> tapply(Exam1, Class, var)
       2        3        4
  8.0000 353.7154 298.0797

> tapply(Exam1, Class, length)
 2  3  4
 2 23 24

> table(Class)     # same result
Class
 2  3  4
 2 23 24

> boxplot(Exam1 ~ Class)
> boxplot(Exam1 ~ Class*Sex)

> ######################################################
> # Saving the ANOVA and using summary() is better
>
> anova.fit <- aov(Exam1~Class)
> anova.fit
Call:
   aov(formula = Exam1 ~ Class)

Terms:
                    Class Residuals
Sum of Squares    198.632 14645.572
Deg. of Freedom         2        46

Residual standard error: 17.84326
Estimated effects may be unbalanced

> summary(anova.fit)
            Df  Sum Sq Mean Sq F value Pr(>F)
Class        2   198.6    99.3  0.3119 0.7336
Residuals   46 14645.6   318.4
```

- REGRESSION

```
> #### Simple Linear Regression
>
> fit<-lm(Final~Exam1)
> attributes(fit)
$names
 [1] "coefficients"  "residuals"     "effects"       "rank"
 [5] "fitted.values" "assign"        "qr"            "df.residual"
 [9] "xlevels"       "call"          "terms"         "model"

$class
[1] "lm"

> fit

Call:
lm(formula = Final ~ Exam1)

Coefficients:
(Intercept)        Exam1
    60.9026       0.9761

> summary(fit)

Call:
lm(formula = Final ~ Exam1)

Residuals:
    Min      1Q  Median      3Q     Max
-80.586 -13.634   1.223  15.076  32.485

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  60.9026    14.8897   4.090 0.000168 ***
Exam1         0.9761     0.1807   5.401 2.14e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 22.02 on 47 degrees of freedom
Multiple R-squared: 0.383,      Adjusted R-squared: 0.3699
F-statistic: 29.17 on 1 and 47 DF,  p-value: 2.141e-06

> anova(fit)
Analysis of Variance Table

Response: Final
```

```
          Df  Sum Sq Mean Sq F value    Pr(>F)
Exam1      1 14143.7 14143.7  29.173 2.141e-06 ***
Residuals 47 22786.5   484.8
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> confint(fit)                    # Confidence Intervals
                2.5 %    97.5 %
(Intercept) 30.9483634 90.856852
Exam1        0.6125545  1.339687

> ?confint
> confint(fit, level=.90)
                  5 %      95 %
(Intercept) 35.918756 85.886459
Exam1        0.672882  1.279359

> ###############################################################
> plot(Exam1,Final)
> abline(fit)        # Or abline(60.9026, .9761)
>
> # Residual plot
> plot(Exam1, fit$residuals)
>
> # Add a reference line
> abline(0,0)

> # Confidence intervals for predicted values
> predict.lm(fit,interval="confidence")
        fit       lwr      upr
1  151.68183 143.89753 159.4661
2  153.63407 145.40505 161.8631
3  122.39821 113.41683 131.3796
4  150.70571 143.12738 158.2840
5  158.51468 149.02010 168.0093
6  156.56243 147.59684 165.5280
7  144.84899 138.21595 151.4820
8  158.51468 149.02010 168.0093
9  153.63407 145.40505 161.8631
10 130.20718 122.99265 137.4217
11 123.37433 114.64720 132.1015
12 150.70571 143.12738 158.2840
13 109.70864  96.93167 122.4856
14 123.37433 114.64720 132.1015
15 103.85192  89.14016 118.5637
```

```
16 157.53856 148.31184 166.7653
17 127.27881 119.48153 135.0761
18 155.58631 146.87449 164.2981
19 138.99226 132.66136 145.3232
20 144.84899 138.21595 151.4820
21 141.92062 135.52929 148.3120
22 137.04002 130.64552 143.4345
23 142.89675 136.44429 149.3492
24 145.82511 139.07416 152.5761
25 124.35045 115.86959 132.8313
26 148.75347 141.54961 155.9573
27 135.08778 128.54895 141.6266
28 136.06390 129.60706 142.5207
29 147.77735 140.73998 154.8147
30 136.06390 129.60706 142.5207
31 138.01614 131.66376 144.3685
32 151.68183 143.89753 159.4661
33 154.61019 146.14413 163.0763
34 155.58631 146.87449 164.2981
35 135.08778 128.54895 141.6266
36  93.11459  74.71188 111.5173
37 153.63407 145.40505 161.8631
38 141.92062 135.52929 148.3120
39 147.77735 140.73998 154.8147
40 130.20718 122.99265 137.4217
41  94.09071  76.02897 112.1525
42 152.65795 144.65646 160.6594
43 154.61019 146.14413 163.0763
44 139.96838 133.63812 146.2986
45 100.92356  85.22014 116.6270
46 146.80123 139.91518 153.6873
47 122.39821 113.41683 131.3796
48 157.53856 148.31184 166.7653
49 153.63407 145.40505 161.8631

> To estimate expected values of new observations
> predict.lm(fit,newdata=data.frame(Exam1=c(55, 95)),interval="confidence")
       fit      lwr      upr
1 114.5892 103.3554 125.8231
2 153.6341 145.4050 161.8631

> ##############################################################################
> # Multiple Linear Regression
>
> fit<-lm(Final~Exam1+Exam2)
```

```
> fit

Call:
lm(formula = Final ~ Exam1 + Exam2)

Coefficients:
(Intercept)        Exam1        Exam2
    17.8765       0.7800       0.6494

> summary(fit)

Call:
lm(formula = Final ~ Exam1 + Exam2)

Residuals:
   Min     1Q Median     3Q    Max
-83.48 -10.57   4.80  16.05  31.48

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  17.8765    27.7671   0.644 0.522902
Exam1         0.7800     0.2068   3.772 0.000462 ***
Exam2         0.6494     0.3571   1.819 0.075472 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 21.5 on 46 degrees of freedom
Multiple R-squared: 0.4244,     Adjusted R-squared: 0.3993
F-statistic: 16.96 on 2 and 46 DF,  p-value: 3.042e-06

> anova(fit)
Analysis of Variance Table

Response: Final
          Df  Sum Sq Mean Sq F value    Pr(>F)
Exam1      1 14143.7 14143.7 30.6056 1.446e-06 ***
Exam2      1  1528.6  1528.6  3.3076   0.07547 .
Residuals 46 21258.0   462.1
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
>   confint(fit)       # the default is a 95% confidence level
                   2.5 %    97.5 %
(Intercept) -38.01587470 73.768866
Exam1         0.36372682  1.196230
```

```
Exam2        -0.06934886  1.368248
```

- CONTINGENCY TABLES

```
> coursegrade
 [1] 4 4 1 3 4 4 3 4 4 3 3 3 2 1 0 4 1 2 3 4 3 2 3 2 3 4 2 3 3 3 4 3 4 3 3 0 4 3
[39] 3 3 1 4 3 3 1 4 2 4 3

> table(coursegrade)
coursegrade
 0  1  2  3  4
 2  5  6 21 15

> table(coursegrade,Class)
           Class
coursegrade  2  3  4
          0  0  2  0
          1  0  3  2
          2  0  4  2
          3  2  8 11
          4  0  6  9
>
> # Read summary counts into a matrix for chi-square analysis
>
> classdata<-matrix(c(0,2,0,0,3,2,0,4,2,2,8,11,0,6,9),nrow=5,byrow=T)
> classdata
     [,1] [,2] [,3]
[1,]    0    2    0
[2,]    0    3    2
[3,]    0    4    2
[4,]    2    8   11
[5,]    0    6    9
> dimnames(classdata)<-list(c("F","D","C","B","A"),c("Sophomore","Junior","Senior"))
> classdata
  Sophomore Junior Senior
F         0      2      0
D         0      3      2
C         0      4      2
B         2      8     11
A         0      6      9

*****************************************************************************
> # Chi-square test for independence
```

```
>
> chisq.test(classdata, correct=F)

        Pearson's Chi-squared test

data:  classdata
X-squared = 6.8326, df = 8, p-value = 0.5548

Warning message:
In chisq.test(classdata, correct = F) :
  Chi-squared approximation may be incorrect

# Better to store output
>
> out<-chisq.test(classdata, correct=F)
Warning message:
In chisq.test(classdata, correct = F) :
  Chi-squared approximation may be incorrect

> attributes(out)
$names
[1] "statistic" "parameter" "p.value"   "method"    "data.name" "observed"
[7] "expected"  "residuals"

$class
[1] "htest"

> out$expected
   Sophomore    Junior     Senior
F 0.08163265 0.9387755  0.9795918
D 0.20408163 2.3469388  2.4489796
C 0.24489796 2.8163265  2.9387755
B 0.85714286 9.8571429 10.2857143
A 0.61224490 7.0408163  7.3469388
```

- SOME PLOTS

```
> # Printing plots
> ?device
> pdf()
> plot(Exam1,Exam2, main="Scatterplot", type="l",xlab="1st Exam", ylab="2nd Exam")
> dev.off()
```

```
X11cairo
       2

> postscript()
> plot(Exam1,Exam2, type="p")

> jpeg()
> plot(Exam1,Exam2, type="p")
> dev.off()

> system("ls")

> dev.list()                 # open devices
X11cairo     jpeg
       2        3

> dev.cur()                  # current active device
> dev.set(3)                 # set new active
jpeg
   3
> dev.cur()
jpeg
   3
> dev.off()
X11cairo
       2
> dev.cur()
X11cairo
       2

>  # More Plots

> dotchart(grades)
> dotchart(grades[,4:8])
> grades2<-cbind(Exam1,Exam2,Final)
> dotchart(grades2)
> hist(Exam1)
> plot(Exam1, Exam2)
> plot(Exam1, Exam2, pch=22)
> plot(Exam1, Exam2, pch=22, col="red")
> plot(Exam1, Exam2, pch=22, col="red", bg="yellow")
> plot(Exam1, Exam2, pch=22, col="red", bg="yellow",xlim=c(50,100),ylim=c(50,100))

> # Change graphing parameters (beyond arguments in plot command)
```

```
> opar<-par()    # save default parameters
> par(bg="lightyellow", col.axis="blue")
> plot(Exam1, Exam2, pch=22, col="red", bg="yellow",xlim=c(50,100),ylim=c(50,100))
> par(opar)      # Back to default

> # Low level plotting commands  (adds to current graph)
> plot(Exam1,Exam2)
> abline(0,1)
> plot(Exam1,Exam2)
> segments(40,60,100,100)
> abline(h=80)
> abline(v=80)
> legend(70,52,"Outlier")
> title("Test Plot")

> #### T-DENSITY WITH SHADED TAILS
> x<-seq(-3.5,3.5,by=.1)
> y<-dt(x,100)
> plot(x,y,type="l",xlab="t",ylab="f(t)")
> a<-c(seq(1.7,3.5,by=.1),seq(3.5,1.7,by=-.1))
> a
 [1] 1.7 1.8 1.9 2.0 2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9 3.0 3.1 3.2 3.3 3.4 3.5 3.5 3.4
[22] 3.3 3.2 3.1 3.0 2.9 2.8 2.7 2.6 2.5 2.4 2.3 2.2 2.1 2.0 1.9 1.8 1.7
> b<-dt(a,100)
> b[20:38]=0
> b
 [1] 0.094401610 0.079524428 0.066380394 0.054908643 0.045014094 0.036577181 0.029462833
 [8] 0.023528353 0.018630004 0.014628231 0.011391541 0.008799144 0.006742523 0.005126090
[15] 0.003867144 0.002895306 0.002151587 0.001587248 0.001162555 0.000000000 0.000000000
[22] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
[29] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
[36] 0.000000000 0.000000000 0.000000000
> #### POLYGON SHADES REGION INSIDE GIVEN COORDINATES
> polygon(a, b,  col = "gray")
> a<-c(seq(-3.5,-1.7,by=.1),seq(-1.7,-3.5,by=-.1))
> b<-dt(a,100)
> b[20:38]=0
> polygon(a, b,  col = "gray")
> #### MTEXT WRITES TEXT ON MARGINS
> mtext("|t_obs|",side=1,at=c(1.7),cex=.7)
```

- BASIC SIMULATION

```
> #### Simulate 20 rolls of a die using 'sample' function
>
> sim1.out<-sample(c(1,2,3,4,5,6), 20, replace=T)
> sim1.out
 [1] 1 1 6 5 2 5 2 2 5 3 5 2 5 6 2 1 2 2 4 1
>
> # Specify probabilities
> sim2.out<-sample(c(1,2,3,4,5,6), 20, replace=T, prob=c(1/9, 1/9, 1/9, 1/9, 2/9, 3/9))
> sim2.out
 [1] 2 5 6 6 3 6 6 4 6 5 4 1 5 2 6 3 6 2 6 6
>
> # Calculate the ratio Y/X where Y is bigger and X is smaller of the two rolls
> roll1 <- sample(c(1,2,3,4,5,6), 1)
> roll2 <- sample(c(1,2,3,4,5,6), 1)
> Y<-max(roll1,roll2)
> X<-min(roll1,roll2)
> Z<-Y/X
> cbind(roll1, roll2, Y, X, Z)
     roll1 roll2 Y X Z
[1,]      4      1 4 1 4
>
> ######## SIMULATIONS: 'REPLICATE' set of commands n=50 times, store output
>
> store1<-replicate(50, {
+ roll1 <- sample(c(1,2,3,4,5,6), 1)
+ roll2 <- sample(c(1,2,3,4,5,6), 1)
+ Y<-max(roll1,roll2)
+ X<-min(roll1,roll2)
+ Z<-Y/X
+ })
> store1
 [1] 1.500000 1.666667 1.500000 1.200000 4.000000 1.250000 1.500000 1.250000 2.000000
[10] 1.666667 3.000000 2.000000 1.666667 1.500000 2.000000 4.000000 1.000000 2.500000
[19] 2.000000 2.000000 2.500000 6.000000 2.000000 2.500000 6.000000 1.000000 5.000000
[28] 5.000000 1.000000 3.000000 2.000000 1.000000 1.200000 2.000000 1.666667 2.000000
[37] 2.000000 2.000000 1.666667 1.333333 1.200000 6.000000 1.333333 1.500000 2.000000
[46] 3.000000 1.500000 2.000000 2.000000 5.000000

> ######## SIMULATIONS USING 'FOR' LOOP
>
> store2 <-rep(NA, 50)   # Initialize storage
> for (i in 1:50) {
+ roll1 <- sample(c(1,2,3,4,5,6), 1)
+ roll2 <- sample(c(1,2,3,4,5,6), 1)
+ store2[i]<-max(roll1,roll2)/min(roll1,roll2)
```

```
+ }
> store2
 [1] 3.000000 1.333333 2.000000 1.500000 2.500000 2.000000 4.000000 3.000000 1.000000
[10] 4.000000 2.500000 3.000000 1.200000 1.666667 3.000000 2.500000 1.000000 3.000000
[19] 1.666667 1.333333 1.500000 1.000000 1.250000 5.000000 1.000000 1.333333 1.500000
[28] 1.000000 1.500000 1.200000 1.500000 1.333333 2.000000 6.000000 1.333333 1.333333
[37] 1.333333 3.000000 1.666667 2.000000 5.000000 1.500000 1.250000 2.000000 1.000000
[46] 1.666667 1.333333 2.500000 1.500000 1.000000

> ######## SIMULATIONS USING VECTORIZATION
>
> vec1<-sample(c(1,2,3,4,5,6), 50, replace=T)
> vec2<-sample(c(1,2,3,4,5,6), 50, replace=T)
> vec1
 [1] 6 5 5 6 4 2 1 1 1 2 6 4 6 6 4 3 2 4 6 6 3 1 2 6 1 2 3 1 3 2 6 5 2 3 1 2 5 1 5 2
[41] 3 6 1 2 6 4 3 3 4 3
> vec2
 [1] 3 4 5 3 1 5 2 1 1 1 6 1 6 1 3 1 3 2 5 3 3 6 5 5 1 6 4 4 2 1 2 6 1 1 6 3 2 3 4 2
[41] 5 5 1 1 5 4 2 4 4 3
> store3<-apply(cbind(vec1,vec2),1,max)/apply(cbind(vec1,vec2),1,min)
> store3
 [1] 2.000000 1.250000 1.000000 2.000000 4.000000 2.500000 2.000000 1.000000 1.000000
[10] 2.000000 1.000000 4.000000 1.000000 6.000000 1.333333 3.000000 1.500000 2.000000
[19] 1.200000 2.000000 1.000000 6.000000 2.500000 1.200000 1.000000 3.000000 1.333333
[28] 4.000000 1.500000 2.000000 3.000000 1.200000 2.000000 3.000000 6.000000 1.500000
[37] 2.500000 3.000000 1.250000 1.000000 1.666667 1.200000 1.000000 2.000000 1.200000
[46] 1.000000 1.500000 1.333333 1.000000 1.000000
```

- LOGICAL OPERATORS

```
> ######### Generate n=30 observations from Normal with mean=10 and SD=9
> vec1<-rnorm(30,10 ,9)
> > vec1
 [1]  2.22291562 11.50199530 16.92193190  9.44570397 21.93644176 13.63682677
 [7] 22.02488952 14.55572916  8.60853717  4.91032621 -5.53497885 20.60865627
[13] 20.49993573 -0.05165645  0.56842967 -0.55399406  8.69629780  1.02993680
[19] 2.59011827 18.71027607  4.74164980  5.42508909 16.55819195 17.95701268
[25]  2.36378439 13.64130019 25.87822665  3.22077925  0.84679924  0.50391216
>
> ######### Initialize vec2 for Likert storage
> vec2<-rep(NA,30)
> vec2
 [1] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
```

```
[26] NA NA NA NA NA
>
> ######### Create categorical Likert using logical operators
> vec2[vec1<=qnorm(.20,10,9)]="Low"
> vec2[vec1>qnorm(.20,10,9) & vec1<=qnorm(.40,10,9)]="Low-Mid"
> vec2[vec1>qnorm(.40,10,9) & vec1<=qnorm(.60,10,9)]="Middle"
> vec2[vec1>qnorm(.60,10,9) & vec1<=qnorm(.80,10,9)]="Mid-High"
> vec2[vec1>qnorm(.80,10,9)]="High"

> ######### Initialize vec3 for Likert storage
> vec3<-rep(NA,30)
> vec3
 [1] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
[26] NA NA NA NA NA
>
> ######### Create numerical Likert
> vec3<-rep(NA,30)
> vec3[vec1<=qnorm(.20,10,9)]=1
> vec3[vec1>qnorm(.20,10,9) & vec1<=qnorm(.40,10,9)]=2
> vec3[vec1>qnorm(.40,10,9) & vec1<=qnorm(.60,10,9)]=3
> vec3[vec1>qnorm(.60,10,9) & vec1<=qnorm(.80,10,9)]=4
> vec3[vec1>qnorm(.80,10,9)]=5
>
> cbind(vec1,vec2,vec3)
      vec1                   vec2       vec3
 [1,] "2.22291561713552"     "Low"      "1"
 [2,] "11.5019952957777"     "Middle"   "3"
 [3,] "16.921931896917"      "Mid-High" "4"
 [4,] "9.44570396785374"     "Middle"   "3"
 [5,] "21.936441760283"      "High"     "5"
 [6,] "13.6368267701268"     "Mid-High" "4"
 [7,] "22.0248895201799"     "High"     "5"
 [8,] "14.5557291577566"     "Mid-High" "4"
 [9,] "8.60853717350829"     "Middle"   "3"
[10,] "4.91032621009245"     "Low-Mid"  "2"
[11,] "-5.53497884706063"    "Low"      "1"
[12,] "20.6086562744806"     "High"     "5"
[13,] "20.4999357338484"     "High"     "5"
[14,] "-0.0516564484029889" "Low"      "1"
[15,] "0.568429671773474"    "Low"      "1"
[16,] "-0.553994055885202"   "Low"      "1"
[17,] "8.69629779889798"     "Middle"   "3"
[18,] "1.02993679879524"     "Low"      "1"
[19,] "2.59011826964375"     "Low-Mid"  "2"
[20,] "18.7102760669909"     "High"     "5"
```

```
[21,] "4.74164980290352"    "Low-Mid"  "2"
[22,] "5.4250890905658"     "Low-Mid"  "2"
[23,] "16.5581919493113"    "Mid-High" "4"
[24,] "17.9570126828948"    "High"     "5"
[25,] "2.36378439038029"    "Low"      "1"
[26,] "13.641300190615"     "Mid-High" "4"
[27,] "25.8782266462673"    "High"     "5"
[28,] "3.22077924913933"    "Low-Mid"  "2"
[29,] "0.84679923631457"    "Low"      "1"
[30,] "0.503912163783978"   "Low"      "1"
```