# Getting Started with R and S-plus

*M*ohammad *E*hsanul *K*arim <wildscop@yahoo.com>

Copyright © 2003 Mohammad Ehsanul Karim ™ v1.1 ®
Permission to reproduce individual copies of this manual for personal use is granted. Multiple copies may be created for nonprofit academic purposes. Using this text for commercial instruction is prohibited without permission. The author takes no responsibility for damage that may result from their use.

## Inrtroduction :

If you are a total "newbie" in S-plus (or R; a different non-commercial software originated from same S language; learning any one will be enough to work in another), get at least S-plus 4.0 ® or higher or R 1.3.1 § or higher version and just type the commands after " > " sign given here in "commands window" in S-plus or in "R console" in R. In most cases I presented the respective results as well, just in case you like to match your results with me. Also whenever I used # [….] after any command, write the command with out # [….]. Just try yourself; when time comes, you will know by yourself. If you don't exclude them, don't worry, the command will still execute in its original format! You know what : you don't need to know anything (but a little Statistics and Linear Algebra) to learn R / S-plus, cause they themselves will guide you to be correct since software today are built in such a user-friendly way. Command writing perfection is not the necessary to workout any job here. I checked all commands provided here in S-plus 4.0. Have fun while poking around the following commands! Good luck!!

## Getting help from R or S-plus help :

```
> help()              # [get the entire help content.]
> help(plot)          # [see what happens, also try help(matrix) ! ]
> ?plot               # [same as before, provides specific help.]
```

## Assigning Values to Variables :

```
> x<-3                # [puts x = 3 ]
> x                   # [remember that "X" is not equal to "x" in S-plus / R]
```

```
[1] 3
> 4->x                  # [puts x = 4 , just 2nd way of assigning values to variable ]
> x
[1] 4
> x_5                   # [puts x = 5 , just 3rd way of assigning values to variable]
> x
[1] 5
> 5_x                   # [ This does not input 5 as a value of x, so you will receive an
Error message ]
Error: Left side of assignment can't be of mode numeric
Dumped

> x<-3;y<-4             # [ Puts values of x and y simultaneously]
>
> x
[1] 3
> y
[1] 4
> assign("x",c(1,2,3)) # [puts x = 1,2,3 , just 4th way of assigning values in x]
NULL
> x
[1] 1 2 3
```

## Working with Vectors :

```
> x<- c(1,2,3)
> x
[1] 1 2 3
> x[2]
[1] 2
> y<- c(x,4)
> y
[1] 1 2 3 4
> y[5]
[1] NA
> y[-c(2:3)]            # [ Excludes 2nd and 3rd value of y]
[1] 1 4
```

## Arithmatic Operations and built-in functions in S-plus:

```
> prod(x)
[1] 6
> p<-(12/(2+6)**4)-(((6-1)*5)^3)
> p
[1] -15625
```

```
> print(p)                    # [Same command as “p” only]
[1] -15625
> summary(x)
 Min. 1st Qu. Median Mean 3rd Qu. Max.
    1     1.5      2    2     2.5    3
> summary(y)
 Min. 1st Qu. Median Mean 3rd Qu. Max.
    1    1.75    2.5  2.5    3.25    4
> summary (x[x==y])
 Min. 1st Qu. Median Mean 3rd Qu. Max.
    1     1.5      2    2     2.5    3
Warning messages:
  Length of longer object is not a multiple of the length of the shorter object in: x == y
> length(y)
[1] 4
> sum(y)
[1] 10
> max(y)
[1] 4
> min(y)
[1] 1
> mean(y)
[1] 2.5
> sqrt(y)
[1] 1.000000 1.414214 1.732051 2.000000
> e_sqrt(y)
> sort(e)
[1] 1.000000 1.414214 1.732051 2.000000
> mode(e)
[1] "numeric"
> mode(p)
[1] "numeric"
> rr<-numeric()
> rr
numeric(0)
> mode(rr)
[1] "numeric"
> f<-c(8,-3,6,2,-6,3,-9,2,-6,2)
> abs(f)
 [1] 8 3 6 2 6 3 9 2 6 2
> sort(f)
 [1] -9 -6 -6 -3  2  2  2  3  6  8
> order(f,x)
 [1]  7  5  9  2  4 10  8  6  3  1
> sin(f)
 [1]  0.9893582 -0.1411200 -0.2794155  0.9092974  0.2794155  0.1411200 -0.4121185
```

```
0.9092974  0.2794155
[10]  0.9092974
> floor(sin(f))                    # [ Look ! I'm sick of instructing about all the commands,
just gimme a break!! Why don't you try "help(floor)" to know what its functionality is;
would you ? Same goes for the rest of the Document :p ]
 [1]  0 -1 -1  0  0  0 -1  0  0  0
> trunc(sin(f))
 [1] 0 0 0 0 0 0 0 0 0 0
> ceiling(sin(f))
 [1] 1 0 0 1 1 1 0 1 1 1
> round(sin(f))
 [1] 1 0 0 1 0 0 0 1 0 1
> quantile(x,seq(.1,.9,.1))
 10% 20% 30% 40%  50%  60%  70%  80%  90%
 2.9 4.8 6.7 8.6 10.5 12.4 14.3 16.2 18.1
> quantile(x,.77)
   77%
 15.63
> quantile(x,c(.25,.75))
  25%   75%
 5.75 15.25
```

## Removing an object / variable :

```
> objects()
[1] ".Last.value"  "last.dump"    "last.warning" "p"          "x"          "y"
> rm(y)                    # [ removes "y" from objects]
> objects()
[1] ".Last.value"  "last.dump"    "last.warning" "p"          "x"
```

## Working in Character mode :

```
> z<- c("Help","me","out")
> z
[1] "Help" "me"   "out"
> z[3]
[1] "out"
> sort(z)
[1] "Help" "me"   "out"
> z1<-c("d","a","e","r")
> order(z1)
[1] 2 1 3 4
> sort(z1)
[1] "a"  "d"  "e"  "r"
> factor(z1)
```

```
[1] d a e r
> levels(z1)
NULL
> mode(z) #[modes are: numerical,complex,logical,character.Some consider function as
other mode]
[1] "character"
> rr<-character()
> rr
character(0)
> mode(rr)
[1] "character"
```

## Working in Logical mode :

```
> 3==4                  # [ This means, is 3 = 4 ?]
[1] F                   # [ The result says " FALSE " or F only]
> 3!=4
[1] T
> 3<4
[1] T
> 3>4
[1] F
> 3<=4
[1] T
> 3>=4
[1] F
> x
[1] 1 2 3
> x<3
[1] T T F
> x==2
[1] F T F
> sum(x<3)
[1] 2
> x[x<3]
[1] 1 2
> x[x<3 & x!=1]
[1] 2
> x[x==y]
[1] 1 2 3
Warning messages:
  Length of longer object is not a multiple of the length of the shorter object in: x == y
```

## Working in Complex mode :

```
> sqrt(-17+2i) #[complex number]
[1] 0.2421185+4.130208i
```

## Creating Sequential objects :

```
> seq(1,10,2)            # [ initial value = 1, final value = 10, increment = 2]
[1] 1 3 5 7 9
> seq(100,10,-5)         # [ reversible sequence with –ve increment]
 [1] 100  95  90  85  80  75  70  65  60  55  50  45  40  35  30  25  20  15  10
> b<- 1:30               # [considers all value from 1 to 30]
> b
 [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 27 28 29
30
> c<- 30:1
> c
 [1] 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2
1
> seq(0,2*pi,length=25) # [ specifies length of resulting vector]
 [1] 0.0000000 0.2617994 0.5235988 0.7853982 1.0471976 1.3089969 1.5707963
1.8325957 2.0943951 2.3561945
[11] 2.6179939 2.8797933 3.1415927 3.4033920 3.6651914 3.9269908 4.1887902
4.4505896 4.7123890 4.9741884
[21] 5.2359878 5.4977871 5.7595865 6.0213859 6.2831853
```

## Creating repeatative objects :

```
> rep(3,6)               # [repeats 3 six times]
[1] 3 3 3 3 3 3
> rep(1:3,4:6)
 [1] 1 1 1 1 2 2 2 2 2 3 3 3 3 3 3
> rep(c(1,3,2),length=20)
 [1] 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3
```

## Matrix Algebra :

```
> m<-matrix(1:6,nrow=2,byrow=T)  # [ since we gave 6 data only, specifying row is only
enough to create a matrix since S-plus / R can calculate the number of columns by itself]
> m
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
> dim(m)                 # [ Asks the dimension values of “m”]
```

```
[1] 2 3
> dimnames(m)          # [ Asks the dimension names of “m”]
NULL
> col0<- c("c1","c2","c3")
> row0<- c("r1","r2")
> x
[1] 1 2 3
> names(x)<-col0
> x
 c1 c2 c3
  1  2  3
> dimnames(m)<-list(row0,NULL)
> m
   [,1] [,2] [,3]
r1    1    2    3
r2    4    5    6
> dimnames(m)<-list(NULL,col0)
> m
     c1 c2 c3
[1,]  1  2  3
[2,]  4  5  6
> dimnames(m)<-list(row0,col0)
> m
   c1 c2 c3
r1  1  2  3
r2  4  5  6
> dimnames(m)
[[1]]:
[1] "r1" "r2"

[[2]]:
[1] "c1" "c2" "c3"

> m[1,3]                # [ Asks the value of 1st row and 3rd column of “m”]
[1] 3
> m["c2","r1"]
Error in m["c2", "r1"]: Array subscript (3) out of bounds, should be at most 2
Dumped
> m["r1","c2"]          # [ Asks the value of 1st row and 2nd  column of “m” by names]
[1] 2
> m[,"c2"]              # [ Asks the vector values of 2nd  column of “m”]
 r1 r2
  2  5
> m[,1]
 r1 r2
  1  4
```

```
> m[2,]
 c1 c2 c3
  4  5  6
> m[2,2:3]
 c2 c3
  5  6
> m["r1","c2"]<-(-100)          # [inputs new value -100]
> m
   c1   c2 c3
r1  1 -100  3
r2  4    5  6
> m["r1","c2"]<-2
> m
   c1 c2 c3
r1  1  2  3
r2  4  5  6
> m[,2]/m[,1]
 r1   r2
  2 1.25
> m[2,]/m[1,]
 c1  c2 c3
  4 2.5  2
> c4<-c(4,6)
> m1<-cbind(m,c4)               # [combines “c4” as a new column of  “m”]
> m1
   c1 c2 c3 c4
r1  1  2  3  4
r2  4  5  6  6
> r3<-c(4,5,7)
> m2<-rbind(m,r3)
> m2
   c1 c2 c3
r1  1  2  3
r2  4  5  6
r3  4  5  7
> m2-> sqr
> ndim <- array(1:24,c(3,4,2))
> ndim

, , 1
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
```

```
, , 2
     [,1] [,2] [,3] [,4]
[1,]   13   16   19   22
[2,]   14   17   20   23
[3,]   15   18   21   24
```

## Using Frames :

```
> mc<- c("Dhaka","Khulna")
> m.frame <- data.frame(m,mc)
> m.frame                  #  [Frames are used in S language just to merge two type of data
with same patterns(usually in a column basis since S-plus / R works with Columns unless
specified otherwise).]
   c1 c2 c3     mc
r1  1  2  3  Dhaka
r2  4  5  6 Khulna
> apply(m.frame[,1:3],2,max) # [applies “max” function in the given part of frame]
 c1 c2 c3
  4  5  6
> c1
Error: Object "c1" not found
Dumped
> attach(m.frame)                # [ gives columns  of given frame the respect of a vector]
> c1
 r1 r2
  1  4
> r2
Error: Object "r2" not found
Dumped
> detach()                       # [ takes away the respect of columns ]
> c1
Error: Object "c1" not found
Dumped
> m.frame$c1
[1] 1 4
> m.frame$r2
NULL
> m.frame[,"c1"]
[1] 1 4
> m.frame["r1",]
   c1 c2 c3    mc
r1  1  2  3 Dhaka
> edit(data.frame(m.frame))   # [ See what happens]
   c1 c2 c3     mc
r1  1  2  3  Dhaka
r2  4  5  6 Khulna
```

## Using Lists :

```
> md <-
c("a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t","u","v","w",
+ "x",  "y","z")  # [Here you find " + " sign which should come after an incomplete
command line. So whenever you see a " + " just type / copy all the commands from "  > "
presented just before " + "'s line and until the next "> " comes.]
> m.list <- list(m.frame,md)
> m.list  #  [Lists are used in S language just to merge two type of data with different
patterns.]
[[1]]:
   c1 c2 c3     mc
r1  1  2  3  Dhaka
r2  4  5  6 Khulna

[[2]]:
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s" "t" "u" "v"
"w" "x" "y" "z"
> m.list[[1]][1,2]
[1] 2
> m.list[[2]][22]
[1] "v"
> apply(m.list[[1]][,1:3],2,min)
 c1 c2 c3
  1  2  3
> apply(m.list[[1]][,1:3],2,mean)
  c1  c2  c3
 2.5 3.5 4.5
```

## Working With Missing Values :

```
> q <- matrix(c(4,3,2,3,4,2,3,4,1,2,4,1,2,4,1,2,3,4,2,4),ncol=4,byrow=T)
> q
     [,1] [,2] [,3] [,4]
[1,]    4    3    2    3
[2,]    4    2    3    4
[3,]    1    2    4    1
[4,]    2    4    1    2
[5,]    3    4    2    4
> attr(q,"dim")
[1] 5 4
> w <- rbind (q, matrix(c(1,4,6,8,3,4,5,7),2,4,byrow=T))
> w
     [,1] [,2] [,3] [,4]
[1,]    4    3    2    3
```

```
[2,]    4    2    3    4
[3,]    1    2    4    1
[4,]    2    4    1    2
[5,]    3    4    2    4
[6,]    1    4    6    8
[7,]    3    4    5    7
> u <- cbind (w, matrix(c(1,2,3,4,5,6,7,8,9,0,1,2,3,4),7,2,byrow=F))
> u
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    4    3    2    3    1    8
[2,]    4    2    3    4    2    9
[3,]    1    2    4    1    3    0
[4,]    2    4    1    2    4    1
[5,]    3    4    2    4    5    2
[6,]    1    4    6    8    6    3
[7,]    3    4    5    7    7    4
> u[2,2]<- Inf
> u[4,2]<- Inf
> u[5,2]<- -Inf
> u[5,3]<- NA
> u[3,3]<- NA
> u
     [,1]  [,2]  [,3]  [,4]  [,5]  [,6]
[1,]    4     3     2     3     1     8
[2,]    4   Inf     3     4     2     9
[3,]    1     2    NA     1     3     0
[4,]    2   Inf     1     2     4     1
[5,]    3  -Inf    NA     4     5     2
[6,]    1     4     6     8     6     3
[7,]    3     4     5     7     7     4
> nrow(u)
[1] 7
> ncol(u)
[1] 6
> u[,2]
[1]    3  Inf    2  Inf -Inf    4    4
> v1 <- is.inf(u[,2])
> v1
[1] F T F T T F F
> max(u[,2][!v1])                 # [Excludes infinite values]
[1] 4
> u[,3]
[1]  2  3 NA  1 NA  6  5
> v2 <- is.na(u[,3])
> v2
[1] F F T F T F F
```

```
> max(u[,3][!v2])                 # [Excludes Not Available values]
[1] 6
> mean(u[,3][!v2])
[1] 3.4
> dimnames(u)<-
list(c("row1","row2","row3","row4","row5","row6","row7"),c("col1","col2",
+ "col3","col4","col5","col6"))
> u
      col1  col2  col3  col4  col5  col6
row1     4     3     2     3     1 8
row2     4   Inf     3     4     2     9
row3     1     2    NA     1     3     0
row4     2   Inf     1     2     4     1
row5     3  -Inf    NA     4     5     2
row6     1     4     6     8     6     3
row7     3     4     5     7     7     4
> w1<- u[,"col2"]
> w1
 row1 row2 row3 row4 row5 row6 row7
    3  Inf    2  Inf -Inf    4    4
> is.inf(w1)
 row1  row2  row3  row4  row5  row6  row7
  F     T     F     T     T     F     F
> w2<- u[,"col3"]
> w2
 row1 row2 row3 row4 row5 row6 row7
    2    3   NA    1   NA    6    5
> is.na(w2)
 row1  row2  row3  row4  row5  row6  row7
   F     F     T     F     T     F     F
```

## Matrix Operations :

```
> sng<- matrix(1:4,2,2,byrow=F)
> sng
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> t(sng)
     [,1] [,2]
[1,]    1    2
[2,]    3    4
> solve(sng)
     [,1] [,2]
[1,]   -2  1.5
[2,]    1 -0.5
```

```
> solve(t(sng))
     [,1] [,2]
[1,] -2.0  1.0
[2,]  1.5 -0.5
> A<-sng
> A%*%A   #A(prime).A
     [,1] [,2]
[1,]    7   15
[2,]   10   22
> A%o%A   #A.A(prime)

, , 1, 1
     [,1] [,2]
[1,]    1    3
[2,]    2    4

, , 2, 1
     [,1] [,2]
[1,]    2    6
[2,]    4    8

, , 1, 2
     [,1] [,2]
[1,]    3    9
[2,]    6   12

, , 2, 2
     [,1] [,2]
[1,]    4   12
[2,]    8   16
```

## Using Graphs :

```
> x<- 1:20; y<- x^3
>
> graphsheet()                    # [ Opens graphing device]
> plot(x~y)
> plot(x,y)
> hist(x)
> stem(y)

N = 20   Median = 1165.5
Quartiles = 170.5, 3735.5

Decimal point is 3 places to the right of the colon
```

```
  0 : 000112357
  1 : 037
  2 : 27
  3 : 4
  4 : 19
  5 : 8
  6 : 9
  7 :
  8 : 0
> graphics.off()                # [ Closes graphing device]
> dev.list()
NULL
> graphsheet()
> par(mfrow=c(4,2))
> plot(x,y,type="p")
> plot(x,y,type="l")
> plot(x,y,type="b")
> plot(x,y,type="h")
> plot(x,y,type="s")            #[also try "S"]
> plot(x,y,type="n")
> plot(x,y,type="o")
> plot(x,y,pch="*")
> par(mfrow=c(1,1))
> plot(x~y,pch=116)
> plot(x~y,pch=11)
>plot(x,y, pch="*", main="Graph of \nX and y", sub="Done By Ehsan",
+ xlab="Here we plotted X", ylab="Herewe plotted Y",
+ xlim=c(-5,25), ylim=c(0,7000), type="o", lty=2, axes =T, lwd=2, col=8, box= T)
>plot(x,y,pch="*",main="Graph of \nX and y",sub="Done By Ehsan",xlab="Here we
plotted X",
+ ylab="Here we plotted Y",xlim=c(-5,25),ylim=c(0,7000),type="o",lty=1,axes =F,
lwd=3, col=8, box= F)
> arrows(5,1000,15,2000,col=3)# [What happens in GSD# ?]
> arrows(5,1500,15,2500,col=4)
> box()
> axis(2)
> axes()
> points(5,4000, pch ="#",col=3)
> text(5,3000, "i am here : can you see me ?",col= 7)
> abline(v=c(6,1,9))
> abline(h=c(1000,2000,3000))
>abline(4,2000)
> plot(10:1,c(1:4,NA,6:10),type="b",col=3)
> text(5,5,"Value Not Available here \nso this place remains empty",col=7)
```

## Making Use of External Resources :

```
> scan(file="c:\\s\\data2.dat",what=numeric(),36)  # [Uses External file data2.dat located in c:\\s\ having numerical values only]
 [1] 4 4 1 2 3 1 3 3 3 4 8 4 4 2 3 2 1 4 5 3 4 1 2 4 8 7 1 3 4 5 6 7 8 9 0 1
> scan(file="c:\\s\\data2.dat",what=numeric(),20)
 [1] 4 4 1 2 3 1 3 3 3 4 8 4 4 2 3 2 1 4 5 3
> get1 <- scan(file="c:\\s\\data1.dat",what=numeric(),10)
> get1
 [1] 4 4 1 2 3 3 2 2 4 4
> edit(get1)
 [1] 4 4 1 2 3 3 2 2 4 4
>  get2 <- matrix(scan(file="c:\\s\\data2.dat",what=numeric(),36),ncol=9,byrow=T)
```

## Programming with S-plus :

```
> AppStat <- c("Stat","Math","Econ","Bio")
> for(subj in AppStat) {print (subj)}
[1] "Stat"
[1] "Math"
[1] "Econ"
[1] "Bio"
> # Calculating the sum of positive integers untill it gets larger than 1000
># a) using while loop :
> n <- 0
> m <- 0
> while(m <= 1000)
+ {
+ n <- n+1
+ m <- m+n
+ }
># b) using repeat loop :
> n <- 0
> m <- 0
> repeat
+ {
+ n <- n+1
+ m <- m+n
+ if (m > 1000) break
+ }
```

*- Mohammad Ehsanul Karim* <wildscop@yahoo.com> -