1 Acquiring R

If you would like to use your home computer and don’t feel like paying for Splus, then you can download "R" from the web. This is a free statistical software package that has almost the identical syntax to Splus.
You can download this package from
http://cran.r-project.org/
where there are “precompiled binary distributions” for almost any platform you want eg. Windows, Mac, Linux.

More general information about R is available at
http://www.r-project.org/

2 Loading Splus and R

Splus and R can be loaded:

* on Windows: by finding and clicking on the Splus and R icons.
* on UNIX or Linux: by typing Splus and R at the unix prompt.
* inside Emacs: by typing \texttt{esc-x} then S and R. [This requires the package ESS (Emacs Speaks Statistics) pre-installed in your system, you may ignore this if you don’t know what I am talking about.]

The command prompt for both Splus and R looks like “>”.

3 Quitting Splus and R

You can quit both Splus and R:

* on Windows: by closing the window and answering some simple questions about “saving your work” while doing so.
* on UNIX or Linux: by typing \texttt{q()} and answering some simple questions about “saving your work”.
* inside Emacs: by typing \texttt{C-c C-q} and answering some simple questions about “saving your work”.
4 Getting Help

If you are sure about a particular command called doit then by typing help(doit) or doit brings up the
detailed usage of the command doit.
On the other hand if you know a key word and do not exactly remember the name of the command you want
you should type help.start( ) at the prompt. This brings up a nice search engine and usage details viewer
utility both is Splus and R. Now you can use this to find out the exact command and its usage.

5 Using Splus and R

5.1 Entering Data at the Prompt

The commands here are the same in Splus and R.

> subject <- "Statistics"   # character data
> subject
[1] "Statistics"
>
> number.of.students <- 35  # integer data
> number.of.students
[1] 35
>
> mean.gpa <- 3.99          # float data
> mean.gpa
[1] 3.99
>
> x <- 0.5; y <- 2          # some common mathematical operations
> c(x + y, x * y, x - y, x / y, x^y)
[1] 0.5 2.0 -0.5 -0.5 2
>
> c(exp(x), log(x), sin(x), cos(x), tan(x), asin(x), acos(x), atan(x))
[1] 1.6487213 -0.6931472 0.4794255 0.8775826 0.5463026 0.5235988 1.0471976
[6] 0.4636476
>
5.2 Reading Data from File

The basic commands here are the same in Splus and R but the options vary a lot.

> test.data <- read.table(file = "test.txt", header = FALSE)
> test.data
          V1    V2    V3
1 1  0.1070608 0.15807889
2 2 -0.4709370 -0.45412249
3 3 -0.8566450 -0.96357742
4 4  1.2997959  1.31766887
5 5  1.1973334  1.06245754
6 6  0.0798444  0.08445113
7 7 -0.3512141 -0.44916497
8 8  0.3184387  0.40434262
9 9 -0.4372785 -0.44059976
10 10 -0.9859180 -0.94598495
5.3 Vectors

test.data <- scan(file = "test.txt")
Read 30 items

test.data
[1] 1.00000000 0.10706083 0.15807889 2.00000000 -0.47093701 -0.45412249
[7] 3.00000000 -0.85664496 -0.96357742 4.00000000 1.29979586 1.31766887
[13] 5.00000000 1.19733337 1.05245754 6.00000000 0.07984441 0.08445113
[19] 7.00000000 -0.35121408 -0.44916497 8.00000000 0.31843866 0.40434262
[25] 9.00000000 -0.43727846 -0.44059976 10.00000000 -0.98591799 -0.94598495

> students <- c("Tom", "Dick", "Harry")
> students
[1] "Tom"  "Dick"  "Harry"
> gpa <- c(3.9, 4.0, 3.8)
> gpa
[1] 3.9 4.0 3.8
> students[1]
[1] "Tom"
> gpa[3]
[1] 3.8
> names(gpa) <- students
> gpa
Tom Dick Harry
3.9 4.0 3.8
> gpa["Dick"]
Dick
4
> iq <- c(140, 135, 145)
iq
[1] 140 135 145
> names(iq) <- students
> iq * gpa
Tom  Dick  Harry
546 540 551
> iq / gpa
Tom  Dick  Harry
35.89744 33.75000 38.15789
5.4 Matrices

> data1 <- data2 <- matrix(nrow = 4, ncol = 3)  # declaring matrices
> data1[1,1] <- 1  # inputting the entries
> data1[1,2] <- 2; data1[1,3] <- 3
> data1  # displaying the matrix

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

> data1 <- matrix(1:12, nrow = 4, ncol = 3)  # another way to create a matrix
> data1

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

> data1 <- matrix(1:12, nrow = 4, ncol = 3, byrow = TRUE)  # yet another way
> data1

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

> data1[1,]  # 1st row
[1] 1 2 3

> data1[,1]  # 1st column
[1] 1 4 7 10

> data2 <- matrix(-12:-1, nrow = 4, ncol = 3, byrow = TRUE)  # another matrix
> data2

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

> data1 + data2  # element by element addition

[,1] [,2] [,3]
[1,] -11  -9  -7
[2,] -5  -3  -1
[3,]  1  3  5
[4,]  7  9 11

> data1 * data2  # element by element multiplication
[,1] [,2] [,3]  
[1,] -12 -22 -30  
[2,] -36 -40 -42  
[3,] -42 -40 -36  
[4,] -30 -22 -12  
>  
> t(data2)                 # transpose of a matrix  
[1,] -12 -9 -6 -3  
[2,] -11 -8 -5 -2  
[3,] -10 -7 -4 -1  
>  
> data1 %*% t(data2)       # matrix multiplication  
[1,] -64 -46 -28 -10  
[2,] -163 -118 -73 -28  
[3,] -262 -190 -118 -46  
[4,] -361 -262 -163 -64  
>  
> t(data1) %*% data2       
[,1] [,2] [,3]  
[1,] -120 -98 -76  
[2,] -150 -124 -98  
[3,] -180 -150 -120  
>  

5.5 Functions

> do.nothing1 <- function(foo) {          # a simple function  
+     print(foo + 1)  
+ }  
>  
> do.nothing1(3.5)  
[1] 4.5  
>  
> do.nothing2 <- function(foo, bar = 2) {    # here we assign default value  
+     temp = foo * bar  
+     cat("foo = ", foo, " bar = ", bar, " temp = ", temp, "\n")  
+     return(temp)  
+ }  
>  
> val <- do.nothing2(3.5)  
foo = 3.5 bar = 2 temp = 7  
> val  
[1] 7  
>  
> val <- do.nothing2(3.5, 3)  
foo = 3.5 bar = 3 temp = 10.5  
> val  
[1] 10.5  
>
5.6 Plots

> planetorder <- c(1,2,3,4,5,6,7,8,9,10)
> planetdist <- c(4,8,10,16,30,50,90,200,300,400)

Scatterplots

> plot(planetorder, planetdist)

Stem and Leaf Diagrams

> data <- c(2,6,10,11,12,15,20,22,23,34,35,36)
> stem(data)

N = 12  Median = 17.5  Quartiles = 10.5, 28.5

Decimal point is 1 place to the right of the colon

  0 : 26
  1 : 0125
  2 : 023
  3 : 456

Boxplots

> data <- c(2,6,10,11,12,15,20,22,23,34,35,36)
> data2 <- c(4,5,11,12,14,16,21,34,45,50,55)
> boxplot(data, data2)

Histograms

> diff <- c(-2,-16,-8,-4,-6,2,6,-12,-8,0,0,8,12,-6,-2,6,4,8,16,2)
> hist(diff)
> abline(v=-2)
5.7 Random Number Generation

> population <- seq(1:10)       # population to draw sample from
> population
[1] 1 2 3 4 5 6 7 8 9 10
> samp2 <- sample(population, 5)
> samp2                          # drawing the sample with replacement
[1] 6 4 5 10 9
> samp2 <- sample(population, 5, replace = FALSE)
> samp2                          # drawing the sample without replacement
[1] 3 8 9 5 4
>
> mu <- 5.5; sigma <- 2
> samp1 <- rnorm(10, mu, sigma)  # generates sample of size 100 from a normal
> # distribution with mean mu and s.d. sigma
> samp1
>
There are functions of the form rfoo( ), dfoo( ), pfoo( ), qfoo( ) which draws random numbers from the distribution foo, calculates the density, the cdf, quantiles of the distribution foo respectively.

5.8 Control Flow

We have constructs like if(cond) expr else expr, for (var in seq), while(cond) expr constructs available in Splus and R.

> count <- 0
> if (count <= 0) cat("We are done!\n") else cat("Not yet done!\n")
We are done!
>
> for (i in 1:10) {
+   temp <- c(i, i^2)
+   print(temp)
+ }
[1] 1 1
[1] 2 4
```r
> count <- 5
> while (count > 0) {
+   temp <- c(count, count^2)
+   print(temp)
+   count <- count - 1
+ }
```

```r
[1] 5 25
[1] 4 16
[1] 3 9
[1] 2 4
[1] 1 1
```