
An introduction to S-PLUS and R

1/20/05

S-PLUS is an object-oriented program language based on S initially developed by Bell Lab. With both the Unix and Windows versions, it is widely used for data analysis and research. S-PLUS offers greater flexibility in data analysis and programming than any other statistical packages, and as a byproduct, demands more investment initially. R is essentially a free-version of SPLUS that can be downloaded at: <http://www.r-project.org/> . S-PLUS commands are of function form with arguments. Therefore, parentheses are always used. This introduction pertains to the Unix environment.

First, create a directory for a particular project with S-PLUS, for example: `mkdir lab1` , and type: `cd lab1`. Then type: `Splus CHAPTER`. This will create a subdirectory: `.Data` . It will store all the data and functions you create with S-PLUS.

To start S-PLUS in your directory `lab1`, type: `Splus` to get the latest version 6.2.1. (respectively type: `R`)

The system will respond by the prompt `>`. To quit S-PLUS, type: `> q()`

You may want to customize your startup routine to automatically attach a help window and a graphic window:

```
> .First <- function(){  
  help.start()  
  motif()  
}
```

1. Help

To see a complete list of S-PLUS functions, first open *Netscape* (or other browser) and type:
`> help.start()`

To see documentation of a particular function, `read.table` for example, type:

```
> help(read.table) or > ?read.table
```

Always take a look at the help file to see what arguments a function may take. One can also look at the source codes of each function by typing the function name without parentheses. This is an effective way to learn S-PLUS programming.

All commands are recorded in a file `.Audit` . Use the command `history()` to retrieve previous commands, edit and re-submit them.

2. Creating Data Objects

Data, functions, etc. are referred to as objects in S-PLUS. Objects reside in special directories such as the *.Data* directory in your current working directory. An object will exist forever until you erase it. To list all the objects accessible in a S-PLUS session, type: `> objects()`

The easiest way to create an object is to assign this object with a value. For example:

```
> y <- 10
> y <- c("a", "b", "d")
```

creates a numerical variable *y* and a character vector respectively. If *y* is an existing object, the previous value is replaced by the most recent value. To view the content of an object, simply type for example: `> y`

3. Data Manipulation

Concatenation:

```
> myvec <- c(3, 9, 7, 0, -8)
> myvec
[1] 3 9 7 0 -8
```

Replication:

```
> myvec <- rep(1:5,3)
> myvec
[1] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
> rep(c("A","B","C"),c(3,2,1))
[1] "A" "A" "A" "B" "B" "C"
> rep(c(1,3,2),length=10)
[1] 1 3 2 1 3 2 1 3 2 1
```

Sequence:

```
> myvec<-5:-4
> myvec
[1] 5 4 3 2 1 0 -1 -2 -3 -4
> seq(0,1,0.1)
[1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

Matrix:

```
> a <- 1:5
> b <- 11:15
> newcol <- cbind(a,b)
> newcol
      a  b
[1,] 1 11
[2,] 2 12
[3,] 3 13
[4,] 4 14
[5,] 5 15
> newrow <- rbind(a,b)
> newrow
  [,1] [,2] [,3] [,4] [,5]
a    1    2    3    4    5
b   11   12   13   14   15
> matrix(1:10, ncol=5)
      [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10
```

4. Data Input and Data Frame

Suppose the data file “mydata” is in your current working directory, containing the following five records on three variables:

```
time status group
1     1     a
1     1     c
2     0     b
3     1     a
4     0     b
```

The following command reads data into an S-PLUS objects called data frame

```
mydata <- read.table("mydata", header=T)
```

where `header=T` indicates the first line of the input file is the header of the variables.

```
> mydata
  time status group
1    1      1     a
2    1      1     c
3    2      0     b
4    3      1     a
5    4      0     b
```

The command

```
> names(mydata)
[1] "time" "status" "group"
```

shows the variable names of `mydata`, and

```
> mydata$time
[1] 1 1 2 3 4
```

extracting the `time` component from the data frame. You can also extract `$time` using

```
> mydata[,1]
[1] 1 1 2 3 4
```

If you know that you are going to work with a particular data frame, you can “attach” it, and thus use the simple variable names, until the data frame is “detached”. For example:

```
> attach(mydata)
> time
 1 2 3 4 5
 1 1 2 3 4
> detach()
```

Note that `mydata` in S-PLUS is a different identity from the file “`mydata`”. Variables in a data frame can be of different types, numerical or character, but must be of same length.

Data Input with `scan()`:

Suppose the file “`mydata`” is in your current working directory, containing three values per line, the first being a name and the second and third being test scores. The file “`mydata`” might look like these:

```
joe    20  49
fred   18  32
harry  19  32
sue    25  51
```

The following command inputs the data into a “list” containing three elements, name, score1, and score2:

```
> grades <- scan("mydata", list(name="",score1=0, score2=0))
> grades
$name:
[1] "joe"   "fred"  "harry" "sue"

$score1:
[1] 20 18 19 25

$score2:
[1] 49 32 32 51
```

In the list, `name=""` indicates the first variable is of character type.

S-PLUS objects will remain in S-PLUS until you remove them. The command `rm(mydata)` removes `mydata` from S-PLUS data directory.

5. Output from S-PLUS

A simple way to output S-PLUS objects to an ASCII file is to use the function `sink()`. For example, the following commands:

```
> sink("outfile")
> mydata
> grades
> sink()
```

forward output that usually appear on screen to the file “outfile”. The last `sink()` stop the output to file “outfile”, and resumes the output to screen. The command `> !lp -dname outfile` sends “outfile” to a laser printer with name given by “name”. Another useful command to output S-PLUS objects is `write.table`. See the help for its use.

6. Execution of Program

S-PLUS is often run interactively in that each command/function is interpreted and executed by computer immediately once issued. However, you can also execute a sequence of functions as a batch by putting them in a file, say “myprogram”, and `> source("myprogram")` If the program is large, one may wish to put the execution in a batch mode from outside S-PLUS. That is, at Unix prompt, do *Splus BATCH myprogram outfile* where “outfile” is the name of the file where the output of the program goes.

7. S-PLUS Graphics

The first step to start graphics is to activating a graphic device. Available devices include

- (1) `motif()`: High resolution graphics on window
- (2) `postscript("file")`: generating postscript file
- (3) `printer()`: low resolution graphics on screen (use it only when high resolution graphics not available)

To illustrate, we plot the two scores in data frame “grades”. Type: `> motif(ask=T)` or `> motif()` and a graphic window will pop up. With the option of `ask=T`, there is a prompt between graphs if they are generated in a batch.

```
> plot(grades$score1, grades$score2, xlab="1st score", ylab="2nd score",
      type="o")
```

By default, label will be taken as the variable name, i.e. `grades$score1`. The option `type=` specifies the type of graph, `type=o` stipulates that data points overlay line. For more details, use `help(par)` in S-PLUS. You can also print a hard copy of the graph directly from the window. In doing so, you also create a postscript file for the graph with file name “ps.out.*.ps” where * is a wild card representing a sequence number such as 0002.

You may wish to use the device `postscript()` especially when the system does not provide access to `motif()` and also when you need a hard copy of the graphs.

```
> postscript("graph_file")
> plot(grades$score1, grades$score2, xlab="1st score", ylab="2nd score",
      type="l")
> title("scores", cex=2, side=1, line=3)
```

The function `title` adds a title to the graph. See `help(par)` for more details.

When your terminal does not have a graphic emulator (e.g. dialing from home), you may use the device `printer()`. For example,

```
> printer(height=30, width=60)
> plot(grades$score1, grades$score2, xlab="1st score", ylab="2nd score",
      type="l")
> title("scores", cex=2, side=1, line=3)
> show()
```

The last command `show()` displays the graph.

8. Escaping From S-PLUS

To interact with Unix, one could escape from S-PLUS temporarily using `!` followed by any Unix commands. Example: `> !ls` to list directory content in the working directory, `> !more grades` to view the data file, `> !mail` to see your mail.

9. S-PLUS drill

Let's type the following to see how S-PLUS works:

- `> a<-c(1,1,1,2,2,3)` creates the variable `a` and assigns the vector `(1,1,1,2,2,3)` to it.
- `> m<-mean(a)` will put the mean of `a` in the variable `m`.
- `> median(a)` finds the median of the above vector.
- `> var(a)` finds the variance of `a`.
- `> s<-sqrt(var(a))` will put the standard deviation of `a` in the variable `s`.
- `> hist(a)` creates a histogram of the values in `a`.
- `> b<-c(4,5,6,7,8,9)` creates another vector of length 6.
- `> cor(a,b)` finds the correlation between `a` and `b`.
- `> boxplot(a,b)` creates a boxplot for the two variables `a` and `b`.
- `> d<-1:50` creates the vector `d` containing the numbers 1 through 50. This will come in handy in later problems!
- `> length(a)` returns the length of the vector `a`.
- `> exp(a)` Note the vector result!
- `> 2*a-1` Note the vector result!
- `> a+b` computes the sum of the two above vectors.
- `> a[-c(2,6)]` removes the components 2 and 6 from the vector `a`.
- `> a[a>=2]` extracts the values greater or equal to 2 from `a`.
- `> b[a==2]` extracts the values of `b` corresponding to values of `a` equal to 2.
- `> m<-cbind(a,b)` creates a matrix with the two above vectors.
- `> t(m)%*%m` computes the transposition of `m` and the matrix multiplication.
- `> solve(t(m)%*%m)` computes the inverse of a matrix.
- `> r<-round(20*runif(10))` returns 10 random numbers between 0 and 20.
- `> sort(r)` sorts the vector `r`.
- `> order(r)` returns the permutation that will sort the vector `r`.

10. References

- (1) Becker, R., Chambers, J., Wilks, A. (1988), "The New S Language", Wadsworth & Brooks: California.
- (2) Chambers, J. M., Hastie, T. J. (1993), "Statistical Models in S", Wadsworth & Brooks: California.
- (3) Krause, A., Olson, M. (1997), "The Basics of S and S-plus", Springer.
- (4) Spector, P. (1994), "An Introduction to S and S-plus", Duxbury Press: California.
- (5) Venables, W. N., Ripley, B.D. (1999), "Modern Applied Statistics with S-PLUS", Third Edition, Springer.