

# Splus Tutorial

## Tao Jiang

### January 11, 2002

#### 1. Invoking Splus

If you are using a PC to connect to a unix machine, you must have an X-server running. (All the PCs in room 211 of Sequoia Hall has it running by default) You need to connect to a leland machine such as tree1 or saga via samson or ssh. Then at unix prompt type

```
setenv DISPLAY seqpc-xx:0
```

where xx is the number on the upper right corner of the monitor.

Splus can be invoked from the unix prompt by typing Splus at the command line. A better way to do it is to use Splus from within emacs. Type emacs & to launch emacs. In your emacs window, type ESC x S and hit the return key. Emacs will ask you a “starting data directory”, just accept the default one. Once you see the Splus prompt “>”, you are ready to use Splus.

#### 2. Survival Kit

- (a) To exit Splus, type q().
- (b) To start a help window, type help.start().
- (c) To get text only help on a command, type help(< *command* >)
- (d) To start a graphics window, type motif().
- (e) To close a graphics window, type dev.off().

#### 3. Vectors

The simplest data type of Splus is vector. A scalar is just a vector with length 1. The following examples show how to create vectors.

```
> c(1, 3, 5, 9)
[1] 1 3 5 9
> 1:10
[1] 1 2 3 4 5 6 7 8 9 10
> seq(1, 2, 0.1)
[1] 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0
> rep(2, 4)
[1] 2 2 2 2
> rep(2:3, 2)
[1] 2 3 2 3
> length(3:9)
[1] 7
```

#### 4. Expressions

You can use Splus to calculate any expression you typed. Splus will evaluate it, the result will be printed and then discarded. For arithmetical operations, if the two operands are not of the same length, the shorter vector is recycled as often as need to match the length of the longer vector. For example,

```
> 1:4 + 2
[1] 3 4 5 6
> 1:4 / 2
[1] 0.5 1.0 1.5 2.0
> (2:3) ^ (2:3) ## note exponentiation has higher precedence
[1] 4 27
> log(c(2, 3))
[1] 0.6931472 1.0986123
> sqrt(4)
[1] 2
```

#### 5. Assignments

An assignment evaluates an expression and passes the value to a variable but the result is not printed. Assignments are indicated by the assignment operator “< -” or “\_”. The Splus versions on Ieland machines allow using “=” as assignment operator. For example,

```
> x <- c(1,3,5,7,8,9)
> x
[1] 1 3 5 7 8 9
```

#### 6. Vector and Matrix indexing

In Splus the subscripts of vectors and matrices start from 1. A negative index means all the elements in the vector except it. If the index is out of bound, the result is an NA. NA is the value Splus uses for a missing or undefined value.

```
> x <- c(1,3,5,7,8,9)
> x[3]
[1] 5
> x[1:3]
[1] 1 3 5
> x[-2]
[1] 1 5 7 8 9
> x[10]
[1] NA
```

You can create a matrix with `cbind()`, `rbind()` or `matrix()` function. `cbind()` binds vectors together in columns, `rbind()` binds vectors together in rows and `matrix()` fill a matrix with the elements of a vector.

```

> X <- cbind(x, y=1:6)
> X
      x y
[1,] 1 1
[2,] 3 2
[3,] 5 3
[4,] 7 4
[5,] 8 5
[6,] 9 6
> Y <- matrix(0,2,3)
> Y
      [,1] [,2] [,3]
[1,]    0    0    0
[2,]    0    0    0
> Y <- matrix(x,2,3)
> Y
      [,1] [,2] [,3]
[1,]    1    5    8
[2,]    3    7    9
> Y[1,2] ## extract element on the first row and the second column
[1] 5
> Y[1,]  ## extract the first row
[1] 1 5 8
> Y[,1]  ## extract the first column
[1] 1 3
> Y[2,c(1,3)] ## extract elements (1,3) of the second row
[1] 3 9

```

To multiply two matrices together use the operator `%*%`, `t()` is the matrix transpose operator and function `solve()` can be used to inverse matrices or solve linear systems.

```

> X %*% Y
> t(X)
> solve(Y[,1:2])

```

## 7. Listing/Removing objects from persistent storage

All top level assignments are permanent. The “left value” of an assignment, i.e. an object on the left side of an assignment operator has a persistent storage, which is typically in the folder `MySworck/.Data` in your home directory for the Splus versions on leland machines. For older version Splus it is in the folder `.Data` in your home directory. You can do something in Splus, then close the Splus session and go for lunch. When you are back and relaunch an Splus session the objects you defined are still there. The only exception is that the assignments done inside a function are not stored permanently. All stored objects, including vectors, matrices, lists and functions are global variables and can be accessed inside any function if the function does not have a local variable defined with the same name.

To list all objects in the `.Data` directory use `ls()`. To remove an object from the `.Data` directory use `rm()`.

```

> ls()
[1] "a"    "b"    "x"    "y"
> rm(a)
> ls()
[1] "b"    "x"    "y"

```

## 8. Input and output functions

The function `read.table()` is useful for reading data from an external ascii file into Splus and storing it in a data frame. A data frame is both a matrix and a list of column vectors. `write.table(X, "filepath")` writes `X` into a file. `scan("filepath")` reads a file into a vector. See the help files on these functions for more details. Example:

```

> write.table(X, "foo")
> read.table("foo", sep=",") ## separator is comma
> scan("foo", sep=",", skip=1) ## skip the first line

```

## 9. Sort, rank, order and permutaion (crucial for nonparametric analysis)

```

> a <- c(4, 7, 1, 9, 5)
> sort(a)
[1] 1 4 5 7 9
> rank(a)
[1] 2 4 1 5 3
> order(a)
[1] 3 1 5 2 4

```

Function `sample(x, size)` produces a vector of random partial permutation of the vector `x`, where `size` is the length of the output vector. If the argument `size` is omitted, a random full permutation of vector `x` is returned. For example,

```

> sample(1:10)
[1] 4 3 10 2 8 5 6 1 9 7

```

## 10. Graphics output

Splus privdes comprehensive graphics facilities. Most frequently used tools probably will be scatter plots and histograms. By default, `plot()` function produces scatter plots. You can change the graph style to line plot by providing argument `type = "l"`. You can have both points and lines by providing argument `type = "b"`. There are a whole bunch of options you can specify in the `plot()` function, please refer to Chapter 3 of Venable and Ripley's book for more details.

```

> a <- rnorm(20)
> b <- rnorm(20)
> plot(a)
> plot(a, type="l")

```

```

> plot(a, type="b")
> plot(a, b, main="Line Plot", xlab="X", ylab="Y")
> hist(a)
> printgraph(file="histogram.ps")

```

The last command generates a postscript file for the graphics window. It is a good idea to write the file extension as .ps to avoid confusion. This routine is very useful, especially when you don't have access to a physical printer. It is also possible to print the graphics window to a printer directly. Make sure you know which printer it goes.

Another useful function is par, which enable you to set or ask about graphics parameters. Calling par(mfrow=c(2,2)) will divide the graphics window into 4 cells (2 rows and 2 columns), it is handy if you want to put more plots on one page. To restore to 1 cell setting use par(mfrow=c(1,1)).

## 11. Writing your own functions

Technically Splus is a function language. As you have seen it has a lot of built-in functions, but you will soon come upon situation where you want to write one of your own. Here is a simple example.

```

std.dev <- function(x)
{
# Input: a vector x
# Output: the standard deviation of x
#
  return(sqrt(var(x)))
}

```

The function takes one argument, a vector x, and returns a scalar. The lines beginning with # are comments. To invoke the function on a vector x, you can type std.dev(x). To see the commands which make up the function, just type std.dev (without any brackets).

You can write a function with multiple outputs as shown below.

```

mean.stdev <- function(x)
{
  m <- mean(x)
  stdev <- sqrt(var(x))
  return(list(mean=m, stdev=stdev))
}

> mean.stdev(1:5)
$mean:
[1] 3

$stdev:
[1] 1.581139

```

## 12. Logical objects

So far all the examples we've shown use numeric objects. There are some other modes in Splus as well, namely logical objects, factor objects and character objects. Logical objects are more often used so we discuss them here.

A logical vector is a vector with each element either "T" or "F". Operators like `<`, `<=`, `>`, `>=`, `==` and `!=` take two numeric argument and return a logical vector. Operators like `|`, `&`, `!` are logical or, and, negation. The above operators are vector operators too. Logical objects are often used in indexing. For example,

```
> x
[1] 1 3 5 7 8 9
> x[x>4]
[1] 5 7 8 9
> x[x>4 & x<=7]
[1] 5 7
```

Logical vectors may be used in ordinary arithmetic. They are coerced into numeric vectors. F becoming 0 and T becoming 1. It is useful in calculating the number of elements in a vector that satisfies a certain condition.

```
> a <- c(-1, -4, 3, 5, 7)
> sum(a > 0)
[1] 3
```

## 13. Control structures

### (a) Conditional execution

The format is  
`if (condition) statement else statement`

```
if (is.numeric(x) && min(x) > 0) {
  sx <- sqrt(x)
}
else {
  stop("x must be all positive");
}
```

### (b) Loop

The formats are  
`for (variable in sequence) statement`  
`while (condition) statement`

```
sum <- 0
for (i in 1:100) sum <- sum + i
```

Note that Splus loops are very slow. Use `apply()` for big loops. Try `help(apply)` to get more information on how to use this function.

#### 14. Working from emacs

Emacs is a extremely powerful, flexible and configurable, although it takes some time to learn. I usually invoke Splus from within emacs because it is very easy to modify and rerun your previous commands by moving the cursor to those commands, modifying them and typing the return key to run.

Emacs uses the control key and the escape key to initiate commands. For example, to open a file use `CTRL-x CTRL-f`, which means press and hold the control key and press x, then press and hold the control key and press f. Recall that we used `ESC x S` to invoke Splus in emacs. Note there is no hyphen between `ESC` and `x`, which means you don't have to hold the `ESC` key when you press `x`, instead, just press each key one by one.

With emacs one often splits the window into two parts. One for editing you Splus programs and one for Splus itself. You can copy lines from your program window and paste to the Splus window. Emacs command `CTRL-x 2` splits the window and `CTRL-x 1` restores to one window. Some useful commands are listed below.

<code>CTRL-x CTRL-f</code>	open a file.
<code>CTRL-x CTRL-s</code>	save the current file.
<code>CTRL-x CTRL-c</code>	exit emacs
<code>CTRL-x 1</code>	split window
<code>CTRL-x 0</code>	one window
<code>CTRL-x 5 2</code>	make new frame
<code>CTRL-x 5 0</code>	remove current frame

#### 15. Other references

The text "Modern Applied Statistics with S-PLUS" by Venables and Ripley is a comprehensive survey on how to use Splus to solve statistical problems. There is also a nice online tutorial from Dalhousie University at <http://www.mscs.dal.ca/Splus/contents.html>.