

# Statlab Workshop on S-PLUS

Instructor: Marios Panayides

January 31, 2003

## 1 General Outline

This basic workshop will mainly focus on the S programming language through the S-PLUS software package. In particular, it will be outlined as follows:

- S-PLUS Software and its Gui capabilities.
- The Introduction to the S-PLUS programming commands.
- Basic Graphical capabilities and Trellis graphics.
- Basic Statistical Implementation of the S-PLUS Language
- The attachment of Libraries and Databases.
- S-PLUS and other software compatibility.
- Workshop Summary.

## 2 Yes, there is the Gui Tool Bar and Menu

S-PLUS has a Graphical User Interphase with scroll down menus and toolbars for data analysis and manipulation. The help Demo provided in the workshop will point out the major directions for learning the click and drop menus. The only gui that will be discussed thoroughly in class is the **Object Explorer**.

## 3 Introduction to S-PLUS Object-Oriented language

The majority of operations in S-PLUS result in the creation of objects. These objects are then the main focus of further analysis – “The nouns where the language refers to” – and S-PLUS programming builds up.

You can access the S-PLUS programming language either using the Script Window or the Command Window. The Script Window is more structural as it allows you to save your script and run incremental commands whereas the Command window is interactive and is used for immediate command execution. Both of them will be shown in this workshop.

The Command window –if it is not already opened by default – can be accessed using the *prompt icon* on the main S-PLUS toolbar. The Script Window is opened like any file through the **File ► New** or **File ► Open** – if you already have saved it as a script file – in the main menu.

### 3.1 Learning Resources

Before starting the intro to the S-PLUS language we should get familiarized with the help files and the help menu for it. This is found *either* under the main menu **Help ► Available help ► Language Reference** *or* by typing “?<the command you want to search help for>” in the command or script windows. I suggest the use of the first option.

In addition, extra help can be found from books and manuals that are available either here in the Statlab or from the current S-PLUS company’s web side at:  
<http://www.insightful.com/support/documentation.asp>.

### 3.2 Objects

All the commands in S-PLUS are either expressions or assignments. Expressions usually call the objects and assignments create them. To assign a value to a name, thus creating an object, use the operator “< –” or “\_” or even “=” (for S-PLUS version 6.1).

CAUTION 1: DO NOT USE THE UNDERSCORE “\_” FOR NAMES.

CAUTION 2: S-PLUS IS CASE SENSITIVE.

CAUTION 3: TRY NOT TO USE THE “=” AS IT MIGHT BE CONFUSING AND CAUSE PROBLEMS. -WE WILL SEE AN EXAMPLE LATER ON-.

The most common objects in the S-PLUS language are: **Vectors, Arrays, Lists, Factors, Data frames, and Functions** We will proceed by listing commands for each object. NOTE: “#” can be used before personal comments is the S language.

### 3.2.1 Vectors

Vectors can either have numeric, character or logical strings. They are assigned using the “concatenation” FUNCTION – or otherwise called “call” FUNCTION – “c”.

```
x <- c(1,2,3,4,5,6) # Assignment of numeric vectors.
y <- c(2,4,8,16,32,64)
3:10 # the colon operator produces a numeric vector of sequential integers
x+y # Vector arithmetic is done element by element
x*y
sqrt(x) # basic functions tend to work on each element of a vector
seq(1:4) * rep(3,5) # CAUTION: S-PLUS DOES NOT RECYCLE THE SHORTEST VECTOR
                    # TO MATCH LENGTHS IN VERSION 6
# A character vector:
n<-c("red", "orange", "yellow", "green", "blue", "purple")
# Examples of logical vectors:
x < 2
x <= 3 & x>1
x ==3
n == "red"
x[1] # Vectors can be addressed by the element number
x[1:2] # A vector of element numbers will return a vector
x[-1] # A negative element number will return all but the
      # elements given
x[x> 4] # A logical vector will return the elements that match up
        # the true elements
y [ (x < 3) | (x>5)]
```

```

names(x)# A vector can have names
names(x) <- n
x["red"]
seq(from=1,to=10,by=.1) # produces regular sequences
rep(3,6) #produces repeated patterns
rep(x,3)
length(x) # returns the number of elements of the vector
sum(y)/length(y)#The mean of the vector x
mean(x) # is also the mean of the vector x
b=seq(4,7) # CAUTION: WHEN USING THE '=' SIGN INSTEAD OF THE '=='
a_seq(1,4) # THE FIRST IS AN ASSIGNING OPERATOR WHEREAS THE '=='
a[b==4]    # IS A LOGICAL OPERATOR.
a[b=4]     # NOTICE THE DIFFERENCE

```

### 3.2.2 Matrices and Arrays

Matrices and Arrays have similar characteristics to vectors

```

m <- matrix(y,2,3) # Assignment of a numeric matrix
t(m) # matrix transpose
mm <- matrix(y,3,2,byrow=T) # New assignment
dim(mm)
dim(mm) <- c(1,6)
mm
dim(mm)<-c(3,2)
#An array is an extension of a matrix of length(dim)>=3
a <- array(c(1:8, 11:18, 111:118), dim = c(2,4,3)) # Array
arr <- 1:8
dim(arr) <- c(2,2,2)
m[1,2] #accessing matrices is similar to vectors
m[1,] #you can specify all the row or columns by leaving it blank
#rbind and cbind add rows and columns to matrices

```

```

rbind(1:2,mm)
cbind(1:3,mm)
cbind(matrix(4:8,2,2),matrix(rep(6,4),2)) # Note: One dimension
rbind(matrix(4:8,2,2),matrix(rep(6,4),2)) # See help on matrix
# Some Arithmetics
m + x # adding a vector to a matrix does element by element addition
      # the vector is matched to the matrix going down the columns first
m + m # matrices with the same dimensions can be added
m + m == 2*m # logical matrices can be generated
m + mm # CAUTION: ONLY MATRICES OF THE SAME DIMENSIONS CAN BE ADDED
m*m # * also works element by element (not as you may expect)
m %*% mm # %*% is the matrix multiplication operator
matrix(c(1,3,3,1),2,2) %*% c(2,4) # Inner product ALSO
c(3,1) %*% c(2,2) # Vectors are made into rows or columns as necessary
m <- matrix(c(1,7,2,3),2,2)
solve(m)# "solve" FUNCTION inverts Square matrices
m%*%solve(m)#Verification

```

### 3.2.3 Lists

A list is a collection of items of different types and very useful in S-PLUS.

```

ll<-list(vec1 = x, vec2 = c(3, 49, 54),l.matrx=matrix(c(T, T, F, T),2,2),
state = c("Maine", "New Hampshire", "Massachusetts","Vermont"))
ll[1:3]#The first three elements of ll
ll[1]#The first element, CAUTION: IT IS A LIST
ll[[1]]#The first element, CAUTION: A VECTOR
ll$vec1 # using the $ sign,you recall the element of the list which has
        # the name as it appears after the $ sign
ll$vec2[3] # vec2 is a vector
ll$state[ll$l.mat]# the name of an element can be abbreviated
unlist(ll)#The FUNCTION "unlist" converts a list into a vector

```

```
unlist(l1, use.names=F) # converts into a vector without the names
```

### 3.2.4 Factors

A factor is a special kind of a vector that is really useful when a categorical variable is considered.

```
sex <- rep(c("Male","Female"),c(6,4)) #Produces a character vector
sex.fac <- factor(sex)# changes it to a factor object
sex.fac# notice that the quotes are not printed
print.default(sex.fac) #The underlining vector
levels(sex.fac)# the levels of the vector
sum(sex.fac == "Male")# a numeric vector under condition
sex.fac1_factor(sex,levels=c("Male", "Female")) #change the levels
levels(sex.fac)#The first level gets the value 1
```

### 3.2.5 Data frames

Data are stored in S-PLUS usually as Data frames.

```
dat.frm<-data.frame(age=c(12,32,43,55,43,55),sex=c(rep("M",3),rep("F",3)))
dat.frm[1,] # data frames can be addressed as either a matrix
dat.frm$sex # or a list
#Using rbind and cbind for adding observations and columns
rbind(dat.frm,data.frame(age=43,sex="M"))
cbind(dat.frm,data.frame(Novalue=rep("NA",6)))
dat.frm$old <- dat.frm$age > 40#OR new columns can be assigned
```

**CAUTION: UNLESS OTHERWISE SPECIFIED THE “data.frame” COMMAND CONVERTS CHARACTER AND LOGICAL COLUMNS TO FACTORS.**

```
#with the I() command vectors retain their original mode (i.e. character)
dat.frm1_data.frame(age=c(12,32,43,55,43,55),sex=I(c(rep("M",3),rep("F",3)))
dat.frm1$sex
dat.frm$sex #notice the difference
```

### 3.2.6 Functions

We have used functions before in assigning objects. More generally, functions have input arguments that can either be entered in order in which they occur or be specified by the argument's name. For example:

```
seq(1,10,2)#No names, the arguments must be in the right order
seq(by=2, from=1, to=10)#with names, the order doesn't matter
seq(1,10,length=5)#mixing order and names
seq(from=1,by=3,length=12)#arguments not given are set to default values
# A number of other useful basic functions are :table(x)for tabulations
# apply(x,y,..) is used to apply a function to sections of an array
# tapply(x,y,..), lapply(x,...) and sapply(x,..) apply a function to a list
```

Out of personal experience S-PLUS has a lot of built in functions that can not be learned in one workshop. Actually, they can not be learned in a lot of workshops. Everyday use of the package is needed for you to get to know the different functions and the help language menu is your best guidance.

It is extremely convenient to create your own functions especially for operations repeated frequently. This is done using the command:

```
fname<-function(<AnyArguments>){<the function's body>}
```

Example 1:

```
Range<-function(vec)
{ # the body of the function is enclosed in curly brackets
minvec<-min(vec)
maxvec<-max(vec)
c(minvec,maxvec)#note that the last (unassigned) object
                #listed is the output of the function
}
Range(1:20)#We call the function with its name and
           #the necessary input arguments
```

Example 2:

```
quadr<-function(a,b,cc) {#See that we do not use the call function "c"  
  discrim <- b^2 - 4*a*cc  
  print(discrim)  
  if(discrim < 0) {  
    rt.discrim <- sqrt(-discrim)  
    sol1 <- (-b+rt.discrim*(0+1i))/(2*a)  
    sol2 <- (-b-rt.discrim*(0+1i))/(2*a)  
  } else {  
    sol1 <- (-b + sqrt(discrim))/(2*a)  
    if(discrim==0){return(c(sol1,sol1))}#"return" terminates quadr  
    sol2 <- (-b - sqrt(discrim))/(2*a)  
  }  
  c(sol1,sol2)  
}  
quadr(2,-1,1)  
quadr(1,2,1)
```

### 3.2.7 How to recognize the different objects

Use the commands “attributes(< *objectname* >)” and “mode(< *objectname* >)”.

Examples:

```
attributes(x)  
mode(x)  
attributes(m)  
mode(m)  
attributes(l1)  
mode(l1)  
attributes(dat.frm)  
mode(dat.frm)
```

```
attributes(bear.fit)
mode(bear.fit)
```

Also the object explorer in the Gui can be used as it mentions exclusively the kind of object in the database.

## 4 Graphical output

The main function for diagnostic plots is the “plot(args)” function. In order to find out all the arguments that are involved with this function look the help menu. A list of a number of other useful functions for simple plotting are the following:

```
plot(x,y,...) # creates a plot
lines(x,y,...) # Adds lines to an existing plot
points(x,y,...) # Adds points to a current plot
abline(x,y,...) # Draws a line to a current plot
title(...)# Adds a title
axes(...)# Adds axes
legend(...)# Adds a legend to a current plot

barplot(x,...) # Bar graphs
qqplot(...)# Compares the distribution of two samples
qqnorm(...)# compares the distribution of a sample to a Normal
pairs(x..)# All pairwise scatter plots between multiple variables
hist(x,..)# Histogram -- frequency plot -- of a variable x
par(mfrow=c(i,j)) # To put multiple figures in one plot in an i*j array
                    # starting to fill rows first
par(mfcol=c(i,j))# starting to fill columns first
```

Note: S-PLUS can provide more detailed graphic facilities using the Trellis Graphics. These are mostly used for conditioning plots. Some functions of Trellis plots are:

```
xyplot(formula,...) # a scatter plot
```

```

bwplot(formula,...) # a boxplot
histogram(formula,...) # a trellis histogram
print.trellis(x,split=c(x,y,nx,ny))# multiple figures in one plot

```

NOTE: The formula argument in Trellis commands gives the conditioning variables and it is of the form  $y \sim x \mid g1 * g2 * \dots$

## 5 Statistical S-PLUS language Application

We are going to apply some of the functions mentioned in the previous sections with a data set that is called Bears and it is a text file. Note here that the easiest way to import a data set into S-PLUS is through the gui version under **File ► Import Data ► From File**. However, as we focus on S-PLUS commands we will proceed with the basic commands for importing data. In order to use these commands we need to have the data set in a particular “clean” format preferably as a tab or space delimited.

```

#"read table" creates a Data frame of the data
Bears<-read.table("L:\\S-PLUS\\S+ Workshop\\Bears.txt",header=T,sep="\t",
row.names=NULL)

```

```

#"scan" gives a list where you can specify each element
#as numeric or character.
Bears_scan("L:\\S-PLUS\\S+ Workshop\\Bears.txt",what=list(
ID=0,Age=0,Month=0,Sex=" ",Head.L=0,Head.W=0,Neck.G=0,Length=0,Chest.G=0,
Weight=0,Obs.No=0,Name=""),sep="\t", skip=1)
Bears<-data.frame(Bears) # We need to have a Data frame anyway.

```

Once we have the data in S-PLUS we proceed with a basic statistical analysis:

```

#A general view of the data through scatter plots
pairs(Bears)
#linear analysis is of the form  $y \sim x1+x2+\dots$ 

```

```

bear.fit <- lm(Weight ~ Chest.G,data=Bears)#bear.fit is a "lm" object
names(bear.fit)#as an "lm" object it has names
bear.fit$coef# you can call the individual names
coef(bear.fit)#A different function of getting the coefficients
summary(bear.fit)# a linear model summary
plot(predict(bear.fit),resid(bear.fit))#Notice function predict and resid
hist(resid(bear.fit),nclass=20)# a basic plot
histogram(resid(bear.fit),nint=20)# a trellis plot
plot(bear.fit)#The default plots in a linear models fit
plot(Bears$Chest,Bears$Weight)
abline(bear.fit)#The regression fit is plotted on the existing plot

#What about taking logs
Bears$lweight <- log(Bears$Weight) #creates new columns
Bears$lchg <- log(Bears$Chest.G)
bear.fit1 <- lm(lweight~ lchg,data=Bears)# an new lm object
par(mfrow=c(2,1))# to compare the two graphs
plot(Bears$lc,Bears$lw)
abline(bear.fit1)
plot(Bears$Chest,Bears$Weight)
abline(bear.fit)
summary(bear.fit)
summary(bear.fit1)

par(mfrow=c(1,1))# return to the default one graph per figure
plot(predict(bear.fit1),resid(bear.fit1))
identify(predict(bear.fit1),resid(bear.fit1))
fit <- lm(lweight ~ lchg, data = Bears[-122,])#or
fit <- lm(lweight ~ lchg, data = Bears, subset=c(1:121,123:143))
summary(fit)#Our fit gets better and better!!!!

```

```

anova(fit,test="F")#Analysis of variance to see the Sum of Squares
plot(predict(fit),resid(fit))#No outliers
#What about adding a new variable
fit <- lm(lweight ~ lchg + log(Neck.G), data=Bears[-122,])
anova(fit)
fit <- lm(lweight ~ log(Neck.G)+lchg, data=Bears[-122,])
anova(fit)#Notice how to compare the Sum of Squares

#What about adding a categorical variable. Let us look at Sex
xyplot(lweight ~ lchg |Sex, data=Bears[-122,],col=8) # Trellis plot
fit <- lm(lweight ~ lchg*Sex, data=Bears[-122,] )
summary(fit)
fit$contrasts # This is how the factor was evaluated in the regression
f.Sex<-factor(Bears$Sex[-122],levels=c("Male","Female"))
fit1 <-lm(lweight ~ lchg*f.Sex, data=Bears[-122,])
summary(fit1)#Notice the difference
fit1$contrasts # This is why.
#If you want to find more about contrasts, type "?contrasts"
#You can change the contrast to the one you find most appropriate:
contrasts(f.Sex)_contr.treatment(2)#The usual Zero/One
fit2 <-lm(lweight ~ lchg*f.Sex, data=Bears[-122,])
summary(fit2)#Similar T-values BUT different coefficients
fit2$contrasts
summary(fit1)#Notice the difference again!!
fit1$contrasts

```

## 6 Databases and Libraries

During today's session we have created a number of objects. They can be viewed either with "ls()" or "objects()" or with the **Object Explorer**. Objects can be removed permanently with the

command “rm(name of the objects)”. The S-PLUS objects, are saved in the current **database**. This was specified when we started S-PLUS and can be found as the first directory when we type the command “search()”. This is our current workspace and where objects are saved and will be there after we quit the program (with “q()”). This database is loaded by default however it is not unique. Thus, we can have a number of databases for different research projects. The easiest way to do so is by creating a subdirectory `_data` under our research directory and attach the new database with the command

```
attach("~/\\_data", pos=1)# CAUTION:WHEN SPECIFYING A PATH IN
                        # S-PLUS USE \\ INSTEAD OF \
```

Then you can create objects in the new database instead of the default one. You can detach a database with the command “detach(what=1)” for the first database in the “search()” list.

A **library** is a directory containing mainly functions that deal with a specific subject. For example the library `Matrix` has a number of functions that focus on matrix operations such as the “Identity(x)”, “Diagonal(x)”, “lower.tri(x)”. Another useful library is library `Chron` which deals with dates and times, mostly useful for time series data. The libraries –and their functions– are loaded in S-PLUS using the command “library(libraryname)” and are added to the “search()” directories. For example:

```
library(Matrix);library(chron);search()
```

## 7 Interconnectivity

S-PLUS use and output can be incorporated directly in a lot of software programs like Excel, SPSS, PowerPoint, Mathcad and Arcview. In particular the latter is highly developed and is a great help for people that deal with the GIS science, i.e. highly detailed graphs in combinations with Spatial and Time Series Data.

## 8 Summary-Things to Remember

- S-PLUS is an object oriented language thus, you need to know the particular nature of the object you have created and dealing with. There are particular functions for different objects.
- There are a lot of functions you have to remember and the Help Menu is the best solutions to look something up, especially because the function can take a number of arguments.
- Creating your own function is part of why S-PLUS is such a powerful language. Start with small functions and gradually you can built up your own library.
- The Graphical capabilities of S-PLUS are great, especially the use of Trellis graphs
- The pull-down menus and icons do exists in S-PLUS but limit the capabilities of the software. The **File ► Import Data ► From File** is a useful menu command to know.
- Databases can be created and your work –objects– can be saved there and called whenever necessary.