

S-Plus Short Course

— Part 1 —

These notes are mostly based on

- The S-Plus manuals — available online at Mathsoft's webpage, www.mathsoft.com, for downloads (as PDF-files). Also available on the Stat Dept Computer System (see section on *UMIX*).
- Venable's and Ripley's book on S-Plus: *Modern Applied Statistics with S-Plus*, Springer-Verlag. See www.stats.ox.ac.uk/pub/MASS3.

The Schedule

This second part of this S-Plus Short Course will go through:

EDA Exploratory Data Analysis. This focus on transforming variables, selecting (subset of) variables, and plotting. Work with the toxicity data already created and a short look at two new datasets; `ethanol` and `fuel.frame`.

Classical Univariate Tests t-test, χ^2 -test and other classical univariate tests.

Statistical Models Use the toxicity data to fit models already discussed in class.



The computer that S-Plus is installed on in the Stat Dept is a **UNIX** machine; a Sun SPARC running the SunOS 5.5 UNIX operating system. The following explains how to access this UNIX workstation from computer labs around campus (and using *telnet* at home) and how to start S-Plus.

Enable your OSU Access and Login to Stat

How to login to the Stat Dept computer, *stat.mps.ohio-state.edu*, from computer labs on campus (the following refers to *Macs* on campus):

- Activate your **OSU** internet access. Select **Enable/Disable Your Access** under the apple-menu on the Macs (type **1** to activate your network access and type in your OSU *login-name* and *password* and answer *yes* to the last question). The program which is open now is *telnet* (or *bettertelnet* in some cases).
- From the *telnet* menu select **File** and then **Open Connection...** and in the field, **Host Name**;

type `stat.mps.ohio-state.edu` and click on **Connect**.

- At the resulting *xterm* display, type your *login-name* (for the Stat Dept Account) following the **login:** displayed, and press `return`. Then, following the **password:** displayed, type your *password*, and press `return`.
- Now you are at the UNIX *shell* prompt, `stat>`.

The IP Address, the DISPLAY and x-windows

- Select **Net** in the *telnet* menu and then select **Sent IP Address**, which then will be displayed at the UNIX prompt. Then press `return` (ignore all error messages). This is the **IP Address** of the Mac you are working on.
- Set your working display (monitor) as the Mac you are working on. To do so, type at the UNIX prompt:

```
setenv DISPLAY <your Macs IP number>:0.0
```

where, instead of `<your Macs IP number>` you type in the IP number of your Mac (from above). Then press `return`.

- Open the **X-windows** program. To do so, click on the harddisk icon, then on the **Communication** folder, then on the **MacX** folder, and finally on the program icon, named **MacX**.

Open an *xterm* window in X-windows

- Go back to the **telnet** program already open and type at the UNIX prompt:

```
dterm -ls &
```

 this will open another UNIX *shell*, but within the X-windows program (**MacX**) instead of **telnet**.
- Hide the **telnet** program and go back to the X-windows program, **MacX**.

Basic UNIX Commands

The purpose here is not to teach UNIX, just list few basic commands available.

ls To list files and folders in the current folder. (`ls -l` gives a detailed output, and e.g. `ls *.s` lists all files ending with `.s`.)

cd To change folder. (`cd ..` goes up one level, e.g., `cd Notes` goes to the folder **Notes** in current folder.)

rm To remove files and folders. (`rm talk.tex cmd.s` removes the two files `talk.tex` and `cmd.s`, and the command `rm -r Notes` removes the directory **Notes** — note the `-r` option needed.)

mv To move (rename) files and folders. (`mv talk.tex talk-v2.tex` moves `talk.tex` to `talk-v2.tex` in the same folder — after executing the command, the file `talk.tex` doesn't exist any more.)

cp To copy files and folders. (`cp talk.tex talk-v2.tex`)

emacs An editor. (`emacs cmd.s &` opens the file `cmd.s`. In *emacs*, a *file* is also called a *buffer*.)

acroread Adobe's Acrobat reader. (`acroread`

`/opt/local/splusr5.1/doc/unixug.pdf` & opens S-Plus language manual, other S-Plus manuals available (in the same folder) are `statman1.pdf` and `statman2.pdf`.)

To get more information (help) on UNIX commands, e.g., do `man cp` on the UNIX prompt to get help on the copy command, `cp`.

Now we are ready to use S-Plus.

A Gentle Introduction to S-Plus

Before I Start

Before starting S-Plus for the *first time*, type at the *UNIX* prompt (`stat>`) in an `xterminal` open in the X-windows program, the following:

S CHAPTER

and press `return`. (This creates a folder, `.Data`, in your current folder (where you executed the command `S CHAPTER`. In `.Data`, S-Plus stores the *objects* created in a S-Plus session.)

Starting S-Plus

To Start S-Plus, type:

S -e

What you see now is the S-Plus prompt, `>`, a S-Plus *expression* is typed at the prompt and *evaluated* by pressing `return`. More than one expression can be typed in a single line by using `;` as separator. Also, the same expression can be typed in more than one line by pressing `return` when the expression is not completed. At that point, the prompt

changes to `+` to show that the expression is unfinished. The symbol `#` tells S-Plus to ignore whatever is typed after the symbol on the command line (used to put comments).

The `-e` option used when starting S-Plus, in this case, tells S-Plus to use *emacs* type of command line editor. The following table is useful when editing your expressions at the S-Plus command line before evaluating:

Action	keystroke
backward character	Ctrl-b
forward character	Ctrl-f
previous line	Ctrl-p
next line	Ctrl-n
beginning of line	Ctrl-a
end of line	Ctrl-e
kill line	Ctrl-k

My First S-Plus Session

You have already started your S-Plus session. Let's try out few basic evaluations (the S-Plus prompt, `>`, is shown, but

should not be typed, and the output from the expression is also showned; following the command line).

```
> 4 + 5           ## addition
[1] 9
> (9 * 5) - 7     ## use parentheses
[1] 38
> (8/2)^2         ## power
[1] 16
>
```

Output from an *expression* can be *assigned* to an *object* for later use (instead of re-typing the same expression).

Assignments are done with the operator `<-`, or just `_` (easier to read `<-`). Example:

```
> x <- 45         ## assigning 45 to x
> x              ## the object x
[1] 45
> mu <- 40; s <- 10 ## mean and SD
> z <- (x - mu)/s  ## Z-value
> z
[1] 0.5
> lik <- 1/sqrt(2*s^2) * ## the normal density
+ exp(-0.5 * z^2)
> lik
[1] 0.06240195
> log.lik <- log(lik)    ## the log-likelihood
> log.lik
[1] -2.774159
```

```
>
```

Here we both assigned the value of expressions to objects and used some of the *functions* available in S-Plus (`sqrt`, `exp` and `log`).

We can see (*list*) the objects that we have created hitherto:

```
> ls()
[1] ".Last.value" "lik"      "log.lik"
[4] "mu"           "s"         "x"
[7] "z"
>
```

and we can also *remove* objects that we have created, e.g.

```
> rm(log.lik)
>
```

To quit S-Plus

```
> q()
```

Note that the objects created in this S-Plus session are all available to S-Plus when started again.

S, the Programming Language

Objects in S-Plus are basically of three types; *vectors*, *lists* and *functions*.

Vectors

The elements of vectors are either numeric, logical or character strings.

```
> data <- c(25,31,27,29,26)    ## concatenating
> data                        ## print data
[1] 25 31 27 29 26
> length(data)                ## the length
[1] 5
> data[3]                     ## element 3
[1] 27
> ind <- 2:4                  ## sequence (indexing)
> ind
[1] 2 3 4
> data[ind]                   ## subset
[1] 31 27 29
> log(data)                   ## logarithm
[1] 3.218876 3.433987 3.295837 3.367296 3.258097
> sum(data)                   ## sum
[1] 138
```

```
> mean(data)                 ## mean
[1] 27.6
> sqrt(var(data))           ## SD
[1] 2.408319
> stdev(data)                ## SD also
[1] 2.408319
> lev <- c("high", "low")   ## characters
> lev[1]
[1] "high"
>
```

Lists

A *list* can store different types of data — think about it as a list of other S-Plus objects:

```
> my.l <- list(x=1:5, y=data)
> my.l                       ## print out the list
$x:
[1] 1 2 3 4 5
$y:
[1] 25 31 27 29 26
> names(my.l)                ## the names (attributes)
[1] "x" "y"
> my.l[1]                    ## the x item of the list
[1] 1 2 3 4 5
```

```
> my.1$y[2:4]      ## elements 2-4 of y
[1] 31 27 29
>
```

Matrices and Indexing (Logicals)

Data are very often stored in tables (matrices):

```
> mat <- cbind(x=1:5, y=rnorm(5,mean=10,sd=5))
> mat
      x      y
[1,] 1 21.978406
[2,] 2 10.412500
[3,] 3  9.875592
[4,] 4 13.762809
[5,] 5  4.460788
>
```

(Note, your own `y` data, the random numbers from the normal random numbers, will not be the same as above.)

```
> dim(mat)          ## the dimension
[1] 5 2
> dimnames(mat)    ## row and column names, if any
[[1]]:
character(0)
[[2]]:
[1] "x" "y"
```

```
> mat[1:2,]        ## first two rows
  x      y
1 21.97841
2 10.41250
> mat[,1]          ## the first column, by number
 1 2 3 4 5
> mat[, 'y']       ## the second column, by name
21.97841 10.4125 9.875592 13.76281 4.460788
> mat.2 <- mat[-1,] ## remove the first line
> mat.2
      x      y
2 10.412500
3  9.875592
4 13.762809
5  4.460788
>
```

Instead of picking specific lines from the `mat` object created above, we can pick (remove!) lines, conditional on other data.

```
> ind <- mat[, 'y'] > 10 ## which 'y' are > 10 ?
> ind ## vector of logicals, TRUE/FALSE
 T T F T F
> mat.3 <- mat[ind,] ## store in a separate matrix
> mat.3
```

```

>
> x      Y
> 1 21.97841
> 2 10.41250
> 4 13.76281

```

Other logical operators for direct comparison are `<` (less), `==` (equal), `<=` (less or equal), `>=` (greater or equal), `!=` (not equal). And also; `&` (AND), `|` (OR) and `!` (NOT). What does this gives:

```

> mat[!(mat[, 'y'] > 8) & (mat[, 'y'] < 12)], ]
>
> x      Y
> 1 21.978406
> 4 13.762809
> 5 4.460788
>

```

Functions

There are many *functions* built into S-Plus. Here are few:

Arithmetical `abs`, `log`, `sqrt`, `exp`, `sin`, `asin`.

Statistics `sum`, `mean`, `var`, `median`, `min`, `max`, `quantile`, `summary`.

Integers (or digits) `round`, `trunc`, `floor`, `ceiling`.

Distributions For the normal distribution: `dnorm`, `pnorm`, `qnorm` and `rnorm` for the density, `cdf`,

`quantile` and random-numbers, respectively. Similarly for other distributions, e.g., for the densities; `dt` (t), `df` (F), `dchisq` (χ^2), `dbinom` (binomial), `dgamma` (Gamma), `dexp` (exponential), etc.

To get *help* on functions, do e.g. `?rnorm` at the S-Plus prompt. To just see what *arguments* a function takes, do e.g. `args(rnorm)` at the prompt.

Working with Data in S-Plus

In most cases, the data you are working with has already been entered into a *file*; this could just be a plain table in a text-file created by some editor.

In the file `/home/gardar/pub/toxicity.dat` is the toxicity data introduced in class:

con	no	killed
0.10	47	8
0.15	53	14
0.20	55	24
0.30	52	32
0.50	46	38
0.70	54	50
0.95	52	50

In this case, the columns are separated with `tab`. (If you don't have access to this file, it is easy to create and all references to the file in below S-Plus expressions should be changes accordingly, i.e., name and location.)

Import – Using data.Frame objects

An object of *class* `data.Frame` is a list object which

looks and feels like a table. Let's import the toxicity data into S-Plus:

```
> tox <- read.table("/home/gardar/pub/toxicity.dat",
+                   header=T)
> tox
  con no killed
1 0.10 47     8
2 0.15 53    14
3 0.20 55    24
4 0.30 52    32
5 0.50 46    38
6 0.70 54    50
7 0.95 52    50
> class(tox)
[1] "data.frame"
> names(tox)
[1] "con"      "no"       "killed"
> tox$con
[1] 0.10 0.15 0.20 0.30 0.50 0.70 0.95
> names(tox) <- c('d','n','r')
> tox[1:2,]
   d n r
1 0.10 47 8
2 0.15 53 14
>
```

If the data-file was without *headers* (like the file `/home/gardar/pub/toxicity_v2.dat`), then the

header argument to read. table would be given the value FALSE (F). (If you have time, try to read in the file without header and assign c('d', 'n', 'r') as names to the resulting data.frame.)

Explore

```
> summary(tox)      ## summary info for every col
      d
      n
      r
Min.:0.1000      Min.:46.00
1st Qu.:0.1750   1st Qu.:49.50
Median:0.3000   Median:52.00
Mean:0.4143     Mean:51.29
3rd Qu.:0.6000   3rd Qu.:53.50
Max.:0.9500     Max.:55.00

      r
Min.: 8.00
1st Qu.:19.00
Median:32.00
Mean:30.86
3rd Qu.:44.00
Max.:50.00

> sapply(tox,sum)      ## apply sum to every col
      d      n      r
2.9 359 216
>
```

```
> res <- tox$n - mean(tox$n)      ## deviation from mean
> summary(res)
      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-5.2860 -1.7860  0.7143  0.0000  2.2140  3.7140
>
```

Manage

```
> tox$p <- tox$r/tox$n * 100      ## percentage killed
> tox
      d      n      r      p
1 0.10 47      8 17.02128
2 0.15 53 14 26.41509
3 0.20 55 24 43.63636
4 0.30 52 32 61.53846
5 0.50 46 38 82.60870
6 0.70 54 50 92.59259
7 0.95 52 50 96.15385
> tox[tox$p > 50,]
      d      n      r      p
4 0.30 52 32 61.53846
5 0.50 46 38 82.60870
6 0.70 54 50 92.59259
7 0.95 52 50 96.15385
> tox$p/tox$d      ## % killed per toxicity
[1] 170.2128 176.1006 218.1818 205.1282 165.2174
[6] 132.2751 101.2146
>
```

Homework 2

Work with the `tox` data. `frame` already created.

- Compute summary statistics (using the function `summary`) for the `p` column (percentage killed) *only*. Compute the standard deviation of percentage killed.
- Print out the first line in the `tox` data where percentage killed is above 50% (i.e., line 4) *without* specifying the line by number (do not do `tox[4, 1]`), but using the percentage column instead (the `p` column) and logical operators. (That is, even though the toxicity data would look different, your commands would still print out the first line where the percentage killed exceeded 50% for the first time.)

Hint: it is good to write your **S-Plus** commands into a file (any Mac editor installed is fine) and then copy and paste the commands to the **S-Plus** prompt and evaluate and copy and paste the result into a file.

Turn in the **S-Plus** output (showing the commands used also).