36-315: Statistical Graphics and Visualization

Lab 1
Date: January 15, 2002 **Due: start of class January 21, 2002**

---

1 Introduction

Welcome to the first lab of Statistical Graphics and Visualization! In this lab, you will learn the basics of S-PLUS, some simple S-PLUS commands, and load some of the data for your state into S-plus. There are 3 questions. For each one, submit your S-PLUS code and a response from S-PLUS demonstrating that it works.

2 Starting S-PLUS

Create a folder in which to save your work. To start S-PLUS, use the Start menu:

`Start -> Programs -> Math & Stats -> S-PLUS 2000`

It may ask you for a folder name. Browse up to the folder you just created and press Ok. After a short wait, you will get a blank S-PLUS window with a menu bar. If you still need to tell S-PLUS about your work folder, select

`File -> Workspace -> New...`

and give the name of the folder you created. S-PLUS will create a directory within the work folder called `_Data` to store any functions or variables you define. This automatically saves your work from session to session. S-PLUS will confirm this with you when you quit.
To enter commands, select

`Window -> Commands Window`

The `>` is the prompt for entering commands.

3 Entering Commands

To have S-PLUS execute a command, enter the command at the prompt and press enter. For instance, type `2+3` and then press enter.

```
> 2 + 3
[1] 5
```

Arithmetic and order of operations work in the usual way. Multiplication is `*`, division is `/` and exponentiation is `^`.
The format for calling a function is `functionname`(*arg1*, *arg2*, ...), or `functionname()` for a function which takes zero arguments. Typing the function name by itself (with no parentheses) will return the definition of the function. S-PLUS is case-sensitive, so be sure to use the correct capitalization: `Functionname()` and `functionname()` are different things.
The square root and absolute value functions need just one argument each:

```
> sqrt(9)
[1] 3

> abs(-5)
[1] 5
```

The "[1]" which keeps showing up means that the first item on that line is the first item in the vector of output. This indexing is useful when you want to find a particular observation in a long list of output. In the output below, 1000 is the 36th observation.

```
 [1]    0.73    1.46    2.19    2.92    3.65    4.38    5.11    5.84    6.57
[10]    7.30    8.03    8.76    9.49   10.22   10.95   11.68   12.41   13.14
[19]   13.87   14.60   15.33   16.06   16.79   17.52   18.25   18.98   19.71
[28]   20.44   21.17   21.90   22.63   23.36   24.09   24.82   25.55 1000.00
[37]    1.04    1.30    1.56    1.82    2.08    2.34    2.60    2.86    3.12
[46]    3.38    3.64    3.90    4.16    4.42    4.68    4.94
```

If you enter an incomplete command (for instance, by forgetting a close-parenthesis), S-PLUS will give you another prompt, this time a "+".

```
> sqrt(16
+
```

At this point, type the close-parenthesis and the command will execute normally.

```
> sqrt(16
+ )
[1] 4
```

> **Question 1:** If a cube has volume 9 cu.in., what is the length of a side? (Solve it in S-PLUS.)

4 Variables and Assignments

The command for assigning a value to a variable is `<-`. You can basically use any name for a variable, although it will cause problems if you give a variable the same name as a function, so be careful not to use `c` or `length` as a variable name. Although innocuous, these are both functions in S-PLUS. If you do accidentally give a variable the same name as a function, you can remove the variable (see below). Variable names can be as long as you need, but should not contain blank spaces. If you have two words in a variable name, such as "Water Depth", you could write it as `WaterDepth`, `waterdepth`, or `water.depth`. You may not separate words with an underscore (that is a special character in S-PLUS) or a dash (that would be interpreted as a minus sign).
To assign the value `9` to the variable `x`, type:

```
> x <- 9
```

(Read this as "x gets 9".) Note that S-PLUS does not return any output after an assignment. If you type the variable name, S-PLUS will return its value (if you are ever unsure, this is a good way to confirm that an assignment was successful).
Now that `x` has a value, it can be manipulated like any other number.

```
> sqrt(x)
[1] 3

> y <- (5 * (x + 2)) - 3
```

These manipulations do not affect the value of `x`, it is still `9`. To change it, you must assign a new value to `x`.

```
> sqrt(x)
[1] 3
> x
[1] 9
> x <- sqrt(x)
> x
[1] 3
```

Text values can also be assigned to a variable:

```
> y <- "hello"
> y
[1] "hello"
```

The variables `x` and `y` are now stored in your .Data directory. This means you will be able to use them for the remainder of this session and any time you run S-PLUS in the future. For a list of objects in your directory, use the `objects()` command. Unneeded objects may be removed with the `remove` or `rm` commands. The two work the same, only `remove` requires quotation marks around the argument, for instance `remove("x")` is the same as `rm(x)`.

> **Question 2:** How could you exchange the values of two variables x and y using S-PLUS commands?

5 Vectors

All of the preceding examples were concerned with single numbers, but most statistical analyses involve groups of numbers. In S-PLUS, groups of numbers can be manipulated most easily as vectors. Suppose ten replications of an experiment had the following results:

`2, 4.6, 1, 3.7, 5.9, 4.0, 6.7, 2.8, 1.4, 3.1`

You can enter this data into S-PLUS as a vector, and store that vector as a variable (for instance, a variable called "observations").

```
> observations <- c(2, 4.6, 1, 3.7, 5.9, 4.0, 6.7, 2.8, 1.4, 3.1)
```

The function `c()` takes a list and returns it a vector (think of `c` as "concatenate"). Check the value of the variable `observations` by entering it at the prompt.

```
> observations
[1] 2.0 4.6 1.0 3.7 5.9 4.0 6.7 2.8 1.4 3.1
```

Manipulations on vectors work the same way as manipulations on single numbers. Suppose these observations were in inches but need to be converted to centimeters:

```
> 2.54*observations
[1]  5.080 11.684  2.540  9.398 14.986 10.160 17.018  7.112  3.556  7.874
```

Note that the vector `observations` still contains the original data. An arithmetic operation between two vectors will apply element-by-element, such as:

```
> observations*observations
[1]  4.00 21.16  1.00 13.69 34.81 16.00 44.89  7.84  1.96  9.61
```

6 Selecting from Vectors

Subsets of a vector can be selected by using square brackets.

```
> observations[3]
[1] 1
> observations[5:7]
[1] 5.9 4.0 6.7
> observations[c(1,2,7,1)]
[1] 2.0 4.6 6.7 2.0
```

The above functions return the third observation, the fifth through seventh observations, and the first, second, seventh and first (again) observations, respectively.

7 Data frames

Now you will load some of the census data for your state. Use a web browser to go to `ftp://ftp.ciesin.org/pub/census/usa/stf/`, and click on your state. There will be several files. Pick any file that starts with `tr` (these are the census tract files) and ends with `a.zip`. Download it to your work directory, and unzip it. You will get a Comma-Separated Value `csv` file. These can be loaded directly into S-PLUS via the command `read.table`. Substituting the filename you picked, type

```
frame <- read.table("trxxxxx.csv",header=T,sep=",")
```

This gives you a `data frame` where the rows are census tracts and the columns are different variables like `POPPSQMI` and `PCTFEMAL`. You can get the size of the table via

```
dim(frame)
```

Be careful: if you type `frame`, you may get a massive printout. To get a vector of the column names, type

```
dimnames(frame)[[2]]
```

In the file, these names were surrounded by quotes, and S-PLUS converted them into dots. To remove the dots, type this:

```
trim <- function(s) substring(s,2,nchar(s)-1)
dimnames(frame)[[2]] <- sapply(dimnames(frame)[[2]], trim)
```

For a description of the variables, go to the class home page.
A data frame is simply a collection of vectors, which are the columns of the frame. Columns can be selected just like elements of a vector, and the result is a vector.

```
> frame[3]
...
> frame[5:7]
...
> frame[c(1,2,7,1)]
...
```

You can also select columns by name, using a dollar sign:

```
frame$POPPSQMI
```

> **Question 3:** Check that PCTFEMAL is the same as FEMALE divided by TOTPOP times 100 for all census tracts. For example, you can subtract one from the other. (Give S-PLUS code.)