The S Language

The S Language

S is a language and computational environment designed specifically for carrying out "statistical" computations.

- It was designed by statisticians so that they could easily develop and deploy new statistical methodology.
- It brings good ideas on numerical methods, graphics and language design together in a single useful package.
- It represents a philosophy of *turning ideas into programs* quickly and easily.

History

- The S language was developed at AT&T/Lucent Bell Laboratories by *John Chambers*, with the assistance of *Rick Becker*, *Allan Wilks* and *Duncan Temple Lang*.
- The development of the language was aided by collaboration of the AT&T researchers with a much wide group of academics and practitioners.
- The developers of S language recognise a number of distinct *versions* of S which came about during the development of the language.

S Version 1

- Released in 1980.
- The first version of S released outside of Bell Labs.
- Used primarily in North American universities.
- It is described in the book

  S : A Language And Environment For Data Analysis And Graphics

  (also known as the "brown" book).

S Version 2

- Released in 1987.
- Based initial experiences developing S, Chambers developed a new environment called QPE.
- This added the ability for users to define their own extensions to the language. This version of S is described in the book

  The NEW S Language

  (also known as the "blue" book).

S Version 3

- Released in 1990.
- Added some basic object-oriented facilities, which were seen as essential for providing good modelling facilities.
- Included "State of the Art" modelling software.
- These extensions are described in the book

  Models in S

  also known as the "white" book.

S Version 4

- Released in 2000.
- Contains a rather more sophisticated version of object-oriented programming and a number of other developments which support programming activities with S.
- The initial versions of S4 ran very slowly, and this has tended delay its acceptance by the S programming community.
- This version of S is described in the book

  Programming with Data

  (also known as the "green book").

The ACM Software Systems Award

- The importance of John Chambers' work on S was recognised in 1998, when he was awarded the ACM award for software systems.
- The ACM award is the world's most prestigious award for software engineering.
- Other winners of the ACM award include Thompson and Ritchie for the UNIX operating system and Don Knuth for the TEX typesetting system.

John Chambers, Luke Tierney and some other guy.

Using S as a Calculator

A user types *expressions* to the S interpreter.
S responds by computing and printing the answers.

```
> 1 + 2
[1] 3

> 1/2
[1] 0.5

> 17^2
[1] 289
```

S Implementations

S — This was the version of S used internally for research purposes at Bell Labs. A small group of external researchers had access to this version.

S-Plus — In the late 1980s AT&T sold the rights to sell a commercial version of S to Seattle-based *Statistical Sciences*. The company has since been renamed twice and is now known as *Insightful*.

R — In the early '1990s two misguided individuals at the University of Auckland developed an alternative "free" S implementation. It is called R after its original developers: Robert Gentleman and Ross Ihaka.

Grouping and Evaluation

Normal arithmetic rules apply,

```
> 1 + 2 * 3
[1] 7
```

but evaluation can be controlled with parentheses.

```
> (1 + 2) * 3
[1] 9
```

The R developers plotting world domination.

Built-in Functions

Many mathematical and statistical functions are available.

```
> sqrt(2)
[1] 1.414214

> log(10)
[1] 2.302585

> qnorm(.975)
[1] 1.959964
```

Basic S Concepts

- S is a computer language which is processed by a special program called an *interpreter*. This program reads and evaluates S language *expressions*, and prints the values determined for the expressions.
- The interpreter indicates that it is expecting input by printing its *prompt* at the start of a line. By default the S prompt is a *greater than* sign **>**.
- On UNIX or LINUX machines you can start Splus by typing "**Splus**" and R by typing "**R**"

Assignment

Values are stored by *assigning* them a name.
The resulting name-value pair is called a *variable*. The statements:

```
> z = 17

> z <- 17

> 17 -> z
```

all store the value **17** under the name **z**.

Using Variables

Variables can be used in expressions in the same way as numbers. For example,

```
> z = 17

> z = z + 23

> z
[1] 40
```

Vectors

S works naturally with vectors of values. The simple way to combine scalar values into a vector is using the **`c()`** function.

```
> x = c(1, 2, 4, 3, 1)
> x
[1] 1 2 4 3 1
```

Vectors can be manipulated just like scalar values.

```
> 2 * x
[1] 2 4 8 6 2

> x/4
[1] 0.25 0.50 1.00 0.75 0.25
```

Special Values

In addition to ordinary numbers, S has special codes which indicate infinite and undefined numerical values.

```
> 1/0
[1] Inf

> -1/0
[1] -Inf

> 0/0
[1] NaN
```

Properties of Vectors

The functions **`length`** and **`mode`** return information about the values stored in vectors.

```
> x = c(1, 2, 4, 3, 1)

> length(x)
[1] 5

> mode(x)
[1] "numeric"
```

Special Values

Infinities and NaNs have expected properties.

```
> 100 + Inf
[1] Inf

> 100 + NaN
[1] NaN

> 100/Inf
[1] 0

> Inf/Inf
[1] NaN
```

Simple Subsetting

Individual vector elements can be accessed with a subsetting mechanism.

```
> x = c(1, 2, 4, 3, 1)

> x[5]
[1] 1

> x[3]
[1] 4
```

Other Types

In addition to real numbers, S also has complex numbers, logical values and character strings.

```
> sqrt(-1+0i)
[1] 0+1i

> 10 > 20
[1] FALSE

> s = "hello"
> s
[1] "hello"
```

Vector Subsets

Extracting vector subsets.

```
> x = c(1, 2, 4, 3, 1)

> x[c(1, 2, 3)]
[1] 1 2 4

> x[-5]
[1] 1 2 4 3
```

(When subscripts are all negative, all elements except those are extracted).

Logical Subsetting

Values can be extracted using logical conditions. For example, the command **x[x>2]** extracts all elements from **x** which are greater than 2.

```
> x = c(1, 2, 4, 3, 1)

> x[x > 2]
[1] 4 3
```

Logical subsetting can be very expressive.

```
> mean(income[gender == "male"])
```

Patterned Sequences Using seq

More general sequences can be created by using the "**seq**" function. The expression **seq(0,5,by=0.2)** generates the sequence of values from 0 to 5 in steps of 0.2.

```
> seq(0, 5, by = 0.2)
 [1] 0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6
[10] 1.8 2.0 2.2 2.4 2.6 2.8 3.0 3.2 3.4
[19] 3.6 3.8 4.0 4.2 4.4 4.6 4.8 5.0
```

It is also possible to create sequences of a specified length.

```
> seq(0, 4, length = 9)
[1] 0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0
```

Modifying Subsets

It is possible to change the values in a subset of a vector by using subsetting in conjunction with assignment. For example, the expression

```
> x[1:2] = 10
```

will change the first two elements of the vector **x** to **10**.

Patterned Sequences Using rep

Another function which is useful for creating patterned sequences is the function "**rep**", which repeats its first argument according to the value of its second. The second argument can either be a single count, giving the number of times to repeat the first,

```
> rep(1:3, 5)
 [1] 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3
```

or it can be a vector of counts, indicating how many times to repeat each element of the first argument.

```
> rep(1:3, c(3, 4, 3))
 [1] 1 1 1 2 2 2 2 3 3 3
```

Patterned Sequences

S has a number of ways of generating vectors containing special sequences of values. The one that is most commonly used is the sequence operator "**:**". The expression n_1**:**n_2 returns the sequence of integer values from n_1 to n_2.

```
> 1:10
 [1]  1  2  3  4  5  6  7  8  9 10

> 10:1
 [1] 10  9  8  7  6  5  4  3  2  1
```

Arithmetic on Vectors

S has very general capabilities for vector arithmetic. Arithmetic operations are carried out on vectors according to an *element recycling rule*. Under this rule, when vectors of different lengths are combined in an arithmetic operation, the shorter vector is first enlarged to match the length of the longer vector by recycling its elements. Then the vectors are combined element by element.

Patterned Sequences Using :

The vectors created with **:** can be quite large, and when printed they may span several lines.

```
> 1:50
 [1]  1  2  3  4  5  6  7  8  9 10 11 12
[13] 13 14 15 16 17 18 19 20 21 22 23 24
[25] 25 26 27 28 29 30 31 32 33 34 35 36
[37] 37 38 39 40 41 42 43 44 45 46 47 48
[49] 49 50
```

The value in brackets at the start of each line gives the index of the first value on the line. This can make it easier to locate particular values.

Example

Consider adding the vectors **1:3** and **1:6**.

$$\begin{bmatrix}1\\2\\3\end{bmatrix} + \begin{bmatrix}1\\2\\3\\4\\5\\6\end{bmatrix} \xrightarrow{recycle} \begin{bmatrix}1\\2\\3\\1\\2\\3\end{bmatrix} + \begin{bmatrix}1\\2\\3\\4\\5\\6\end{bmatrix} \xrightarrow{add} \begin{bmatrix}2\\4\\6\\5\\7\\9\end{bmatrix}$$

This is how the result appears when computed with S.

```
> 1:3 + 1:6
[1] 2 4 6 5 7 9
```