Objectives

- **S-plus**
  - Provide an introduction to the S-Plus environment
    - S language
      » Functions and expressions
      » Data types and structures
      » Operators
      » Basic operations on data
- **DNA microarrays**
  - Discuss general reviews
    - What are they?
    - Types
    - How are they “manufactured”
    - Usage
    - Data and sources of variability
    - Data analysis
S-Plus

- An integrated suite of tools for statistics and data analysis
- An interpreted language for expressing statistical models and tools for using linear and non-linear models
- Graphical facilities for data analysis and display
- An open, object oriented programming language that can be extended by the user community
S-Plus basics

• **Objects**
  – Permanence
  – Storage mode

• **Variables**
  – Scalars*, matrices, arrays

• **Functions vs. commands**
  – Expressions ad assignments
> 2 + 3
[1] 5
>
> sqrt(3/4)/(1/3 - 2/pi^2)
[1] 6.626513
>
> mean(chem)
[1] 4.280417
>
> m <- mean(chem); v <- var(chem)/length(chem)
> m/sqrt(v)
[1] 3.958487
>
> #writing new functions
> std.dev <- function(x) sqrt(var(x))
> t.text.p <- function(x, mu=0)
> + { n <- length(x)
+ + t <- sqrt(n) * (mean(x) - mu) / std.dev(x)
+ + 2 * (1 - pt(abs(t), n - 1))
+ }
>
> #The improved function with a listing of the output
> t.stat <- function(x, mu=0)
+   { n <- length(x)
+     t <- sqrt(n) * (mean(x) - mu) / std.dev(x)
+     list(t = t, p = 2 * (1 - pt(abs(t), n - 1)))
+   }
> z<-rnorm(300, 1, 2)
> t.stat(z)
$\text{t:}$
[1] 9.046642

$p:$
[1] 0

> unlist(t.stat(z, 1))
   t     p
-0.1454954 0.8844178
>
Naming convention

- uc, lc roman, digits (non-initial position) and “.”
- **Reserved identifiers**
  - …, break, for, function, if, in, next, repeat, return, while
- **Replacement functions**
  - End with “<-”
- **Non-standard names**
  - Place in double quotes
- **Case sensitive**
- **Names to avoid**
  - C, q, s, t, C, D, F, I, T, diff, mean, pi, range, rank, tree, var
  - Masking of functions
S objects

- **Language layout**
  - Commands
    - Expressions
      - Evaluated and printed
    - Assignments
      - Stored as object
      - Multiple assignments
        ```
        » a <- b <- 6
        ```
    - Separated by “;” or \n
  - #

  - Command line
    - S prompt - >
    - Line continuation +

  - .Last.value
Vectors and matrices

- Numeric, character strings, logical
- Can’t be mixed

```r
> mydata <- c(2.9, 3.4, 3.4, 3.7, 3.7, 2.8, 2.8, 2.5, 2.4, 2.4)
> colors <- c("red", "green", "blue", "white", "black")
> x1 <- 25:30
> x1
[1] 25 26 27 28 29 30
> mydata[7]
[1] 2.8
> colors[3]
[1] "blue"
>```
S objects

- Logical values

```r
> mydata > 3
[1] F T T T T F F F F F
```

- Naming and accessing data by name

```r
> names(mydata) <- c("a", "b", "c", "d", "e", "f", "g", "h", "i", "j")
> mydata
   a  b  c  d  e  f  g  h  i  j
  2.9 3.4 3.4 3.7 3.7 2.8 2.8 2.5 2.4 2.4
> names(mydata)
[1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"
> mydata["e"]
   e
3.7
```
S objects

- **Selection of vectors with vectors**

```r
> letters[1:5]
[1] "a" "b" "c" "d" "e"
> mydata[letters[1:5]]
   a  b  c  d  e
  2.9 3.4 3.4 3.7 3.7
> mydata[mydata > 3]
   b  c  d  e
  3.4 3.4 3.7 3.7
```

- **Selection by omission**

```r
> mydata[ - c(3:5)]
   a  b  f  g  h  i  j
  2.9 3.4 2.8 2.8 2.5 2.4 2.4
```
S objects

- **Attributes**
  - All S objects have two attributes
    - Mode
    - Length

```r
> mode(mydata)
[1] "numeric"
> mode(letters)
[1] "character"
> mode(sin)
[1] "function"
> length(mydata)
[1] 10
> length(letters)
[1] 26
> length(sin)
[1] 2
```
> names(mydata) <- NULL
> # remove the names
> dim(mydata) <- c(2, 5)
> mydata
> [1,]  2.9  3.4  3.7  2.8  2.4
> [2,]  3.4  3.7  2.8  2.5  2.4
>
> dim(mydata) <- NULL
>
> matrix(mydata, 2, 5)
> [1,]  2.9  3.4  3.4  3.7  3.7
> [2,]  2.8  2.8  2.5  2.4  2.4
>
> matrix(mydata, 2, 5, byrow = T)
> [1,]  2.9  3.4  3.4  3.7  3.7
> [2,]  2.8  2.8  2.5  2.4  2.4

S objects
Lists

> Empl <- list(employee = "Anna", spouse = "Fred", children = 3, +    child.ages = c( 4, 7, 9))
> Empl$employee
[1] "Anna"
> Empl$child.ages[2]
[1] 7
> names(Empl) <- letters[1:4]
> Empl[3:4]
$c:
[1] 3
$d:
[1] 4 7 9

> Empl <- c(Empl, service = 8)
> unlist(Empl)
    a   b   c  d1  d2  d3   service
"Anna" "Fred" "3" "4" "7" "9"     "8"
> unlist(Empl, use.names = F)
[1] "Anna" "Fred" "3" "4" "7" "9" "8"
Lists (cont.)

```
> c(list(x = 1:3, a = 3:6), list(y = 8:23, b = c(3, 8, 39)))
$x:
[1] 1 2 3

$a:
[1] 3 4 5 6

$y:
[1]  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23

$b:
[1]  3  8 39
```
Factors

• Categorical variables

```r
> citizen <- factor(c("uk", "us", "no", "au", "uk", "us", "us"))
> citizen
[1] uk us no au uk us us

print.default(citizen)
[1] 3 4 2 1 3 4 4
attr(, "levels"):
[1] "au" "no" "uk" "us"
attr(, "class"):
[1] "factor"
unclass(citizen)
[1] 3 4 2 1 3 4 4
```
Explicit ordering of factors

```
citizen <- factor(c("uk", "us", "no", "au", "uk", "us", "us"),
                   levels = c("us", "fr", "no", "au", "uk"))
> citizen
[1] uk us no au uk us us

> citizen
[1] uk us no au uk us us
Levels:
[1] "us" "fr" "no" "au" "uk"
> table(citizen)
us fr no au uk
3 0 1 1 2
```
Data frames

- **A data matrix**
  - List of variables, all of equal length
  - May be mixed data types

- **Restrictions**
  - Character and logical variables
  - Length and name
    - `cbind()` and `rbind()`

```r
> painters
     Composition Drawing Colour Expression School
 Da Udine       10       8     16          3      A
 Da Vinci       15      16      4         14      A
Del Piombo       8      13     16          7      A
 Del Sarto      12      16      9          8      A
   Fr. Penni     0      15      8          0      A
Guilio Romano    15      16      4         14      A
Michelangelo     8      17      4          8      A
```
Indexing

• Method of extracting subsets from vectors, matrices, arrays and data frames

```r
> painters[1:5, c(2, 4)]
          Drawing Expression
Da Udine     8          3
Da Vinci     16         14
Del Piombo   13          7
Del Sarto    16          8
Fr. Penni    15          0
```
Character vectors

- Character strings, not characters

```r
> paste(c("X", "Y"), 1:4)
[1] "X 1" "Y 2" "X 3" "Y 4"

> paste(c("X", "Y"), 1:4, sep = "")
[1] "X1" "Y2" "X3" "Y4"

> paste(c("X", "Y"), 1:4, sep = "", collapse = " + ")
[1] "X1 + Y2 + X3 + Y4"

> substring(state.name[44:50], 1, 4)
[1] "Utah" "Verm" "Virg" "Wash" "West" "Wisc" "Wyom"
```

Recycling of shorter arguments
Separator character
Collapsing the string
Index types

Vectors

– Logical vectors
  • y <- x[!is.na(x)]
  • z <- (x+y)[!is.na(x) & x > 0]

– Positive integers
  • letters[1:3]
  • letters[1:3][c(1:3, 3:1)]

– Negative integers
  • y <- x[-c(1:5)]

– Character strings

> longitude <- state.center[["x"]]
> names(longitude) <- state.name
> longitude[c("Hawaii", "Alaska")]
  Hawaii  Alaska
  -126.25  -127.25
Arrays

- A multiply indexed collection of data entries
  - Matrix
    - Special case
- Dimension vector
  - Positive integers
  - Length > 0
  - Start at 1
- Additional types
  - Empty array index positions
    - a[1,,] or m[,1]
  - Indexing by a matrix
    - a [1:2, 1:5, ]
    - An example
Function calls

- **Arguments**
  - Specified or unspecified
    - Checking the syntax
  - Lazy evaluation
    - By order
    - Specified
      - name=value
  - Naming conventions may be mixed
  - Default values

```r
> args(hist)
function(x, nclass = "Sturges", breaks, plot = TRUE, probability = FALSE,
    include.lowest = T, ..., xlab = deparse(substitute(x)))
NULL
```
Other important operators and values

- **Arithmetical operators**
  - Vectors
    - All operations are performed element by element
    - +, -, *, /, ^
  - Recycling
    - Shorter vectors are repeated to match length of longer vector
    - Precautions

- **Logical operators**
  - Usually generated by conditions
  - <, <=, =>, >, ==, !=, &, |
  - May be used in ordinary arithmetic
    - F=0, T=1

- **Missing values**
  - NA
    - A special value
    - is.na( ) and na.omit( )
Other important operations

- Generating regular sequences
  - Index
    - ",", rep( ) and seq( )
  - Sampling
  - Complex series

```r
> x <- 1:4
> x
[1] 1 2 3 4
> i <- rep(2, 4)
> i
[1] 2 2 2 2
> y <- rep(x, 2)
> y
[1] 1 2 3 4 1 2 3 4
> z <- rep(x, i)
> z
[1] 1 1 2 2 3 3 4 4
> w <- rep(x, x)
> w
[1] 1 2 2 3 3 4 4 4 4
```

# puts c(1,2,3,4) into x
# puts c(2,2,2,2) into i
# puts c(1,2,3,4,1,2,3,4) into y
# puts c(1,1,2,2,3,3,4,4) into z
# puts c(1,2,2,3,3,3,4,4,4,4) into w