STA 410S    L. # 3    Jan 11/04

§§ 2.1, 2.3, 6.1, ~~6~~.2

§2.1 – every stat. language needs a way to input, output, & store data, lists, ~~numbers~~, and ~~programs~~ (functions)

– and to determine what kind of data, list, function, ... it is

– in S, everything is stored as an 'object'
every 'object' has a class
(more generally has attributes)

– the basic object in S is a vector

```
> a <- 6
> a
[1] 6          # vector of length 1

> a <- 1:6
> a
[1] 1 2 3 4 5 6
```

– vectors can be numeric (most usual), character, logical, complex, integer

or * a list (mixed types)

```
> is.numeric (a)
[1] TRUE
> b <- c("hi", "lo")
> b
[1] "hi" "lo"

> is.character(b)
[1] TRUE
```

- put 2 vectors together with c(v1, v2) (concatenate)

```
> aa <- c(a, a)
> aa
[1] 1 2 3 4 5 6 1 2 3 4 5 6

> ab <- c(a, b)
> ab
[1] "1" "2" ... "6" "hi" "lo"
```

* coerced to character

```
> as.numeric(ab)
[1] 1 2 3 4 5 6 NA NA
```

- > list(a, b)

```
[[1]]                    <- 1st component  } of the list
[1] 1 2 3 4 5 6
[[2]]                    <- 2nd component
[1] "hi" "lo"
```

- statistical data is usually in a matrix

$$
\begin{array}{c|ccc}
 & & j = & \\
 & 1 & \cdots & p \\
\hline
1 & & & \\
i = \;\vdots & & & \\
n & \uparrow & & \uparrow
\end{array}
$$

subjects

cases

variables

- In S a matrix can be assigned by

~~matrix2 ←~~
```
> my.data ← matrix ( data , nrows = , ncol = )
```

! GOTCHA : S fills the matrix by columns
this is not usually what's wanted

(p18) eg
```
> my.data ← matrix ( 1:10, nrow = 2, ncol = 5)
> my.data
      [,1]  ...       [,5]
[1,]    1    3   5   7   9
[2,]    2    4   6   8  10

> matrix (1:10, nrow = 2, ncol = 5, byrow = T)
                nrow = 2, byrow = T
                nr = 2 , by = T
```

- A data matrix usually has row & column names as well.

eg.
```
> library (MASS)
> data (hills)
hills
```

record times of Scottish hill races

| | dist | climb | time |
|---|---|---|---|
| Greenmantle | 2.5 | 650 | 16.083 |
| Carnethy | 6.0 | 2500 | 48.350 |
| Craig Dunain | 6.0 | 900 | 33.650 |
| | ⋮ | ⋮ | ⋮ |

```
> dim (hills)
[1] 35   3
```

- Data can be typed in by hand & converted to a matrix : eg.

```
> x1 <- scan()
1:     type your numbers ... ↲
11:       ...              ↲
21:   ↲
>
> x2 <- scan()                x1  length 20
:                             x2    "    "
:                             x3    "    "
etc.
> mydat <- matrix(c(x1, x2, x3, ..), nr = 20)
```

\# in this case we want column reading
c(x1, x2, x3) length 60 long matrix

- Data can be read from a datafile using

read.table ( ... lots of arguments)

often >read.table (file = "homework.data") will be good enough, but if not, need

> ?read.table or see p. 21

- and finally, most S programmers convert data matrices to

<u>data frames</u> a fancy matrix.

e.g.

```
> is.data.frame (hills)
[1] TRUE
> attributes (hills)
$names
[1] "dist"  "climb"  "time"
$class
[1] "data. frame"
$row.names
[1] "Greenmantle" ...
```

```
> my.frame <- data.frame (mymat)
> names (my. frame)
"X1"  "X2"

> rm (mymat, x1, x2)

> ls ()
[1] "my frame"
```

## §2.3 Data manipulation

```
hills [1, ]      1st row all cols
hills [ , 1]     1st col all rows

hills [1:4, ]    1st 4 rows etc.
```

```
hills $time
hills $dist
hills $climb
```

and VERY HANDY

```
hills [-1, ]     rows 2 through n
hills [, -1]     delete 1st column
```

! GOTCHA

p 29

```
is.matrix ( hills [, -3:-2])      FALSE
hd <- as.matrix (hills)
is.matrix (hd [, -3:-2 ])
, drop = FALSE                    TRUE.
```

- R is very good at operating on vectors, or columns of a matrix, or cols of a data frame
- this is v. useful for programming; see Sort & Data transf p. 32, 33 (skip 34-36 for now)

§ 6.1 : Linear regression

$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$ simple

$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip} + \varepsilon_i$ multiple

$\underset{n\times 1}{\underset{\sim}{y}} = \underset{n\times p}{X} \underset{p\times 1}{\underset{\sim}{\beta}} + \underset{n\times 1}{\underset{\sim}{\varepsilon}}$ $X = \begin{pmatrix} x_{11} & \cdots & x_{1p} \\ & \vdots & \\ x_{n1} & \cdots & x_{np} \end{pmatrix}$

books vary

response – $y$
inputs, covariates, ind't variables, ... $x_1 \dots x_p$
error – $\varepsilon$ additive, independent, mean 0, constant variance
normal
~~indep~~

$\varepsilon_i \sim (0, \sigma^2)$ ind't $\varepsilon_i \sim N(0, \sigma^2)$ ind't

- aspects of analysis: est^n of $\beta$, $\sigma^2$; test which $\beta_j \neq 0$, assess quality of model; decide ~~if~~ all which model gives best fit

- in S the model is fit using the command lm (linear model)

e.g. > lm(y ~ x1 + x2 + x3)

- the result is an object of class lm

> lm( hills$time ~ hills$dist + hills$climb)

> hills.lm ←

$$\begin{pmatrix} x_{11} & \cdots & x_{1p} \\ & \vdots & \\ x_{n1} & \cdots & x_{np} \end{pmatrix}$$

> attributes (hills.lm)
> summary (hills.lm)
> anova (hills.lm)
> plot (hills.lm)

> lm (time ~ dist + climb, $\underbrace{\text{data = hills}}$)

must be a data frame

Call
lm ( formula = time ~ dist + climb, data = hills)

Coefficients

| (Intercept) | dist | climb |
|---|---|---|
| −8.992 | 6.21796 | 0.01105 |
| | (0.601148) | 0.002051 |

HW Question : Compare fits of time ~ dist
time ~ dist + climb

esp. w.r.to influential obs =

- robust regression : not sensitive to failure of normality ass=

- resistant regression : not badly affected by ~~ass=~~ outliers ~~influential~~

§6.3 (more to come on this)

Including categorical / qualitative / discrete covariates
as in §6.1 "analysis of covariance"

- use factor variables (see p. 15)

```
> a <- 1:6
> factor(a)
[1]      1   2   3   4   5   6
levels   1   2   3   4   5   6
```

- model matrix & one-way anova...