Last time: we introduced the vector

- **the basic data structure in R**
- **a series of values**
- **arithmetic works on vectors**
- **recycling rule**
- **indexed by positions starting with 1**
- **[ ] is indexing operator**
  - **any number of numerical indices in any order - an 'index vector'**
  - **a vector of True and False - a 'logical vector' recycled if necessary**

exercise:

- **there's a built in vector of the alphabet: `letters`**
- **familiarise yourself with manipulating vectors by**
  - **displaying letters a through m**
  - **the even numbered letters**
  - **the odd numbered letters**
  - **the vowels**
  - **the consonants**

example:

```
beans <-
c(179,160,136,227,217,168,108,124,143,140)

casein <-
c(368,390,379,260,404,318,352,359,216,222,283,332)

boxplot(beans,casein)

t.test(beans,casein)

t.test(beans,casein,var.equal=T)
```

dynamic typing

last time we saw vectors of 'integer' and 'logical' values

there are several types of values in R

- logical
- numeric
- character
- factor
- (integer, complex, raw)

a given vector can only contain one type

you don't normally have to declare the type

automatic type conversion where it makes sense

OK now you're fed up with vectors.

real data tends to be in tables

typically each row is an individual

each column is an observation of some type

Tables/matrices/arrays are also vectors

they have a dimension attribute

```
a <- 1:100
dim(a) # NULL
dim(a)<-c(10,10)
a[10,10] #100
a[100] #100
```

You can name the rows and columns

```
rn<-c('one','two','three','four','five','six','seven','eight','nine','ten')
```

```
rownames(a)<- rn
```

```
a['three',]
```

this structure allows very elegant and efficient computation

but all data must be same type

Last slide on data structures

A list is like a vector, but its contents can be a mixture

of any kind of objects

```
alist<-list(a,'cowboys and indians',42)
```

this allows data structures of arbitrary complexity

better to keep it simple for casual use

a data frame is a list of vectors of the same length

the columns can be of different types

chickwts example: high level graphics

- extensive example data comes with R and S+

```
data(chickwts) # get chickwts data
chickwts
plot(chickwts[,1])
hist(chickwts[,1])
hist(chickwts$weight)
hist(chickwts$w)
plot(chickwts)
plot(weight ~ feed, data=chickwts)
```

Properties of high-level graphics

- functions like plot can deal with almost any data
  - do something sensible based on what they're given
  - method dispatch based on type of first argument
- the resulting plots are usually aesthetically reasonable
  - carefully chosen defaults
- you can control the details by passing low-level parameters
  - these can also be set as global defaults
- graphs can be saved in a variety of formats
  - metafile, postscript, pdf # vector-graphic
  - png, bmp, jpg # bitmap

chickwts example: anova

```
aov(weight ~ feed, data=chickwts)

chickova <- aov(weight ~ feed, data=chickwts)

summary(chickova)

plot(chickova)

plot(TukeyHSD(chickova))
```

```
Last time: we introduced the vector
        the basic data structure in R
        a series of values
        arithmetic works on vectors
        recycling rule
        indexed by positions starting with 1
        [ ] is indexing operator
                any number of numerical indices in any order - an 'index vector'
                a vector of True and False - a 'logical vector' recycled if necessary

exercise:
        there's a built in vector of the alphabet: letters
        familiarise yourself with manipulating vectors by
                displaying letters a through m
                the even numbered letters
                the odd numbered letters
                the vowels
                the consonants

example:
        beans  <- c(179,160,136,227,217,168,108,124,143,140)
        casein <- c(368,390,379,260,404,318,352,359,216,222,283,332)
        boxplot(beans,casein)
        t.test(beans,casein)
        t.test(beans,casein,var.equal=T)

dynamic typing
        last time we saw vectors of 'integer' and 'logical' values
        there are several types of values in R
                logical
                numeric
                character
                factor
                (integer, complex, raw)
        a given vector can only contain one type
        you don't normally have to declare the type
        automatic type conversion where it makes sense

OK now you're fed up with vectors.
        real data tends to be in tables
        typically each row is an individual
        each column is an observation of some type

Tables/matrices/arrays are also vectors
        they have a dimension attribute
        a <- 1:100
        dim(a)              # NULL
        dim(a)<-c(10,10)
        a[10,10]            #100
        a[100]              #100

You can name the rows and columns
        rn<-c('one','two','three','four','five','six','seven','eight','nine','ten')
        rownames(a)<- rn
        a['three',]
        this structure allows very elegant and efficient computation
                but all data must be same type

Last slide on data structures
        A list is like a vector, but its contents can be a mixture
```