#### R Notes

* Objects
  + 5 basic types of objects: character, numeric, integer, complex, logical
  + Vectors can only contain objects of the same class
  + Lists can contain objects of different classses, but are represented as vectors
  + Empty vectors can be created with the vector() function
* Numbers
  + Numbers in R are generally treated as numeric objects
  + Special number inf which represents infinity
  + The value NaN represents an undefined value or a missing value
* Attributes
  + R objects can have names, dimnames, class, dimensions, length, etc
  + Attributes can be accessed using the attributes() function
* Vectors
  + The c() and vector() function can be used to create vectors of objects
  + When different objects are mixed in a vector, coercion occus so that every element in the vector is of the same class
  + Explicit coercion can occur using the as.\*() functions, ie as.numeric()
  + Nonsensical coercion results in NAs
  + Lists are s special type of vector that can contain elements of different classes

a <- c(0.5, 0.6) ##numeric  
b <- c(TRUE, FALSE) ##logical  
c <- c(T, F) ##logical  
d <- c("a", "b", "c") ##character  
e <- 1:35 ##integer  
f <- c(1+0i, 2+4i) ##complex  
g <- vector ("numeric", length = 10)  
h <- c(1.7, "a") ##character  
i <- c(TRUE, 2) ##numeric  
j <- c("a", TRUE) ##character  
a

## [1] 0.5 0.6

b

## [1] TRUE FALSE

c

## [1] TRUE FALSE

d

## [1] "a" "b" "c"

e

## [1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23  
## [24] 24 25 26 27 28 29 30 31 32 33 34 35

f

## [1] 1+0i 2+4i

g

## [1] 0 0 0 0 0 0 0 0 0 0

h

## [1] "1.7" "a"

i

## [1] 1 2

j

## [1] "a" "TRUE"

k <- 2:8  
class(k)

## [1] "integer"

as.numeric(k)

## [1] 2 3 4 5 6 7 8

as.logical(k)

## [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE

as.character(k)

## [1] "2" "3" "4" "5" "6" "7" "8"

l <- c("a", "b", "c")  
as.numeric(l)

## Warning: NAs introduced by coercion

## [1] NA NA NA

as.logical(l)

## [1] NA NA NA

as.complex(l)

## Warning: NAs introduced by coercion

## [1] NA NA NA

m <- list(1, "a", TRUE, 1+4i)  
m

## [[1]]  
## [1] 1  
##   
## [[2]]  
## [1] "a"  
##   
## [[3]]  
## [1] TRUE  
##   
## [[4]]  
## [1] 1+4i

* Matrices
  + Vectors with a dimension attribute [an integer vector of length 2 (nrow, ncol)]
  + Matrices are constructed column-wise starting with the upper left corner and running down the columns
  + Matrices can also be created from vectors by adding a dimension attribute or by column-binding or row-binding with cbind(), rbind()

a <- matrix(nrow = 2, ncol = 3)  
a

## [,1] [,2] [,3]  
## [1,] NA NA NA  
## [2,] NA NA NA

dim(a)

## [1] 2 3

attributes(a)

## $dim  
## [1] 2 3

b <- matrix(1:6, nrow = 2, ncol = 3)  
b

## [,1] [,2] [,3]  
## [1,] 1 3 5  
## [2,] 2 4 6

c <- 1:10  
c

## [1] 1 2 3 4 5 6 7 8 9 10

dim(c) <- c(2,5)  
c

## [,1] [,2] [,3] [,4] [,5]  
## [1,] 1 3 5 7 9  
## [2,] 2 4 6 8 10

d <- 1:3  
e <- 10:12  
cbind(d, e)

## d e  
## [1,] 1 10  
## [2,] 2 11  
## [3,] 3 12

rbind(d, e)

## [,1] [,2] [,3]  
## d 1 2 3  
## e 10 11 12

* Factors
  + These are used to represent categorical data and can be ordered or unordered
  + Using factors with labels are better than using integers because factors are self-describing i.e. Male/Female rather than 1/2
  + Factors are treated specially by modelling functions like lm() and glm()
  + The order to the levels can be set using the levels argument, this is important since the first level is used as the baseline level

a <- factor(c("yes", "yes", "no", "yes", "no"))  
a

## [1] yes yes no yes no   
## Levels: no yes

table(a)

## a  
## no yes   
## 2 3

unclass(a)

## [1] 2 2 1 2 1  
## attr(,"levels")  
## [1] "no" "yes"

a <- factor(c("yes", "yes", "no", "yes", "no"), level = c("yes", "no"))  
a

## [1] yes yes no yes no   
## Levels: yes no

* Missing Values
  + These are denoted by NA or NaN and are tested by is.na() or is.nan()
  + NA values have the normal classes (integer, character, etc)
  + is.na or is.nan are used to test if objects are NA or NaN
  + A NaN value is also NA, but the converse is also true

a <- c(1, 2, NA, 10, 3)  
is.na(a)

## [1] FALSE FALSE TRUE FALSE FALSE

is.nan(a)

## [1] FALSE FALSE FALSE FALSE FALSE

b <- c(1, 2, NaN, NA, 4)  
is.na(b)

## [1] FALSE FALSE TRUE TRUE FALSE

is.nan(b)

## [1] FALSE FALSE TRUE FALSE FALSE

* Data Frames
  + Used to store tabular data and are represented as a special of list where every element has to have the same length
  + Each element of the list can be thought of as a column and the length of the list the number of rows
  + Unlike matrices, data frames can store different classes of objects in each column
  + Have special attribute row.names
  + Created by calling read.table() or read.csv()
  + Can be converted to a matrix by calling data.matrix()

a <- data.frame(foo = 1:4, bar = c(T, T, F, F))  
a

## foo bar  
## 1 1 TRUE  
## 2 2 TRUE  
## 3 3 FALSE  
## 4 4 FALSE

nrow(a)

## [1] 4

ncol(a)

## [1] 2

* Names
  + Are useful for writing readable code and self-describing objects
  + Lists and matrices can also have names

a <- 1:3  
names(a)

## NULL

names(a) <- c("foo", "bar", "norf")  
a

## foo bar norf   
## 1 2 3

names(a)

## [1] "foo" "bar" "norf"

b <- list(a = 1, b = 2, c = 3)  
b

## $a  
## [1] 1  
##   
## $b  
## [1] 2  
##   
## $c  
## [1] 3

c <- matrix(1:4, nrow = 2, ncol = 2)  
dimnames(c) <- list(c("a", "b"), c("c", "d"))  
c

## c d  
## a 1 3  
## b 2 4

* Reading Tabular Data
  + read.table(), read.csv() are for reading text files (inverse of write.table())
  + readLines() is for reading lines of a text file (inverse of writeLine())
  + source() and dget() are for reading in R code files (inverse of dump() and dput())
  + load() is for reading in saved workspaces (inverse of save())
  + unserialize() is of reading single R objects in binary form (inverse of serialize())
    - read.table