Ashish Mahabal
California Institute of Technology

R - II

Sponsored by the Jet Propulsion Laboratory and by the Keck Institute for Space Studies, California Institute of Technology

Assignments

```
z = 3.14          # using = possible, but …
z <- 3.14         # preferred. _ in emacs, Alt-_ in Rstudio

a <- rnorm(100,mean=5,sd=1) # keywords use =
y <<- 7           # assignment to enclosing scope

z == 3            # Don't confuse = with ==
```

Global variables

```
#bar
foo <- function() {
    bar <<- 1
}
foo()
bar
```

Assignments

```
x <- c(10.4, 5.6, 3.1, 6.4, 21.7)
assign("x", c(10.4, 5.6, 3.1, 6.4, 21.7))
c(10.4, 5.6, 3.1, 6.4, 21.7) -> x                # !!
```

```
x <- c(10.4, 5.6, 3.1, 6.4, 21.7)

1/x                 # 0.09,0.17,0.32,0.15,0.04
y <- c(x,0,x)       # 10.4,.. ,21.7,0,10.4,..,21.7
v <- rep(x) + y + 1 # x repeated 2.2 times, 1 11
var <- sum((x-mean(x))^2)/(length(x)-1)
```

help(sum)

Working with tables/objects (R frame)

```
X = read.table('foo',header=TRUE)
objects()
objects(X)
names(X)        # namespaces!
X
Name1
X$Name1
attach(X)
Name1
```

Num	Name1	Value2
1	1.1	three
2	4.4	four

Redirections, objects etc.

source(“myfile.R”) # source is like <

sink(“outfile”)   # sink is like >

Equivalent of Unix tee

sink("filename",split=TRUE)

Capture output in a variable

output <- capture.output(example(by))

.Rhistory, .Rdata

ls()/objects(), rm() deal with objects

ls() # has lots of options

rm(list = ls()) **# Please don't in a script!**

Everything is a function (returns something)

Spaces do not matter

CapiTaliZation matters

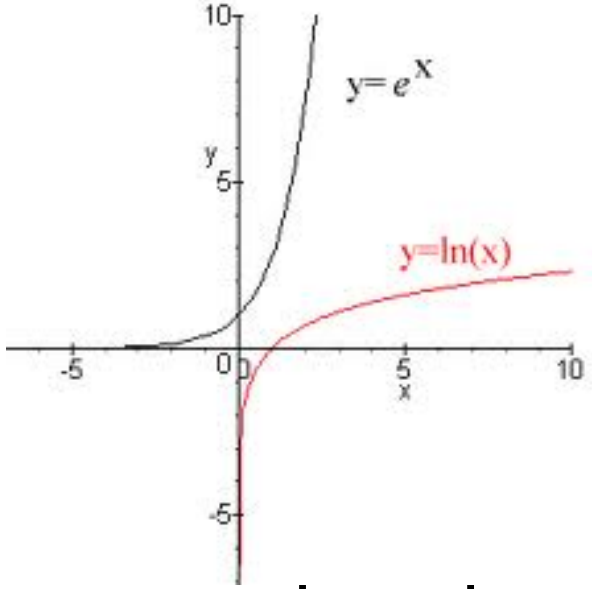
Parameters can be named (odd names!)

```
X = read.table('foo',header=TRUE,sep=',')
```

NA for missing data : is.na() is the corresponding test

```
X = read.table('foo',col.names=c('x','y','z'))
names(X) = c('z','w','t')
```

Simple/standard functions



- + - * / ^
- log, exp, sin, cos, tan, sqrt
- range, min, max, length, sum, var, prod, sort
- sort.list, order
- pmax, pmin # return vectors
- sqrt(-17+0i) # overloaded opeartors

Sequences

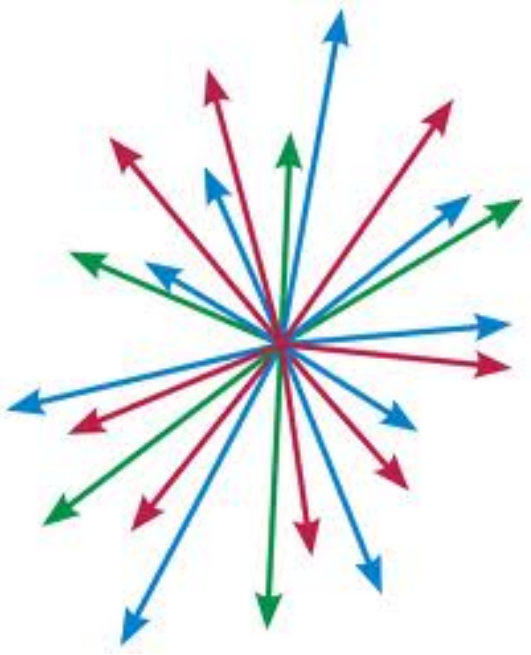
```
1:30                    # 1,2,3,

# ':' is a function and has precedence
n <- 10
seq1 <- 1:n-1
seq2 <-  1:(n-1)

2*1:15      #2,4,..,30
```

Sequences

- seq() with named params: from, to, by, length
  - seq4 <- seq(length=51, from=-5, by=.2)
  - Along can be used only by itself to create same sized vector as another 1:length(vector)
- seq5 <- rep(x, times=5) # x1 x2 .. Xn x1 x2 …
- seq6 <- rep(x, each=5) # x1 x1 x1 x1 x1 x2 x2 ..

Logical vectors

- n <- x >13 # conditional
  - TRUE, FALSE, NA
  - length(n) == length(x)
  - c1 & c2 # intersection
  - c1 | c2 # union
  - !c1 # negation
  - FALSE = 0 and TRUE = 1 when coerced
  - Missing values, NA, NaNs is.nan(x)



Indexing

```
x[1:10]   # get first 10
 x[-(1:5)]     # leave out first 5
x[is.na(x)] <- 0 # replace missing values by 0
y[y < 0] <- -y[y < 0]  OR     y <- abs(y)

z <- c(6,7,8)
z[3] <- 5                  # ]<- is a function here
```

Arrays and matrices

```
dim(z) <- c(3,5,100)          # 3-d array with size
c(a[2,1,1], a[2,2,1], a[2,3,1], a[2,4,1],
    a[2,1,2], a[2,2,2], a[2,3,2], a[2,4,2])
a[,,]        # the entire array
x <- array(1:20, dim=c(4,5))   #  4 by 5 array
i <- array(c(1:3,3:1), dim=c(3,2)) # 3x2 array
x[i] <- 0       # set elements 9, 6, 3 of x to 0
```

Lists (and data.frames)

- Lst <- list(name="Fred", wife="Mary", no.children=3, child.ages=c(4,7,9))

Always numbered

- Lst[[4]] # returns [1] 4 7 9
- Lst[[4]][2] # returns 7
- Lst[4][2] # returns Null
- Lst[3] # returns $no.children [1] 3

Matrix operations

- A * B # element by element matrix product
- A %*% B # matrix multiplication
- x <- array(1:16, dim=c(4,4))
- y = array(rnorm(16),dim=c(4,4))
- x*y
- x%*%y

Reading from files

- HousePrice <- read.table("houses.data", header=TRUE)
- read.table(file, header = FALSE, sep = "", quote = "\"'",
  dec = ".", row.names, col.names, as.is = !stringsAsFactors,
  na.strings = "NA", colClasses = NA, nrows = -1, skip = 0, check.names = TRUE, fill = !blank.lines.skip, strip.white = FALSE, blank.lines.skip = TRUE,
  comment.char = "#", allowEscapes = FALSE, flush = FALSE,
  stringsAsFactors = default.stringsAsFactors(), fileEncoding = "",
  encoding = "unknown")

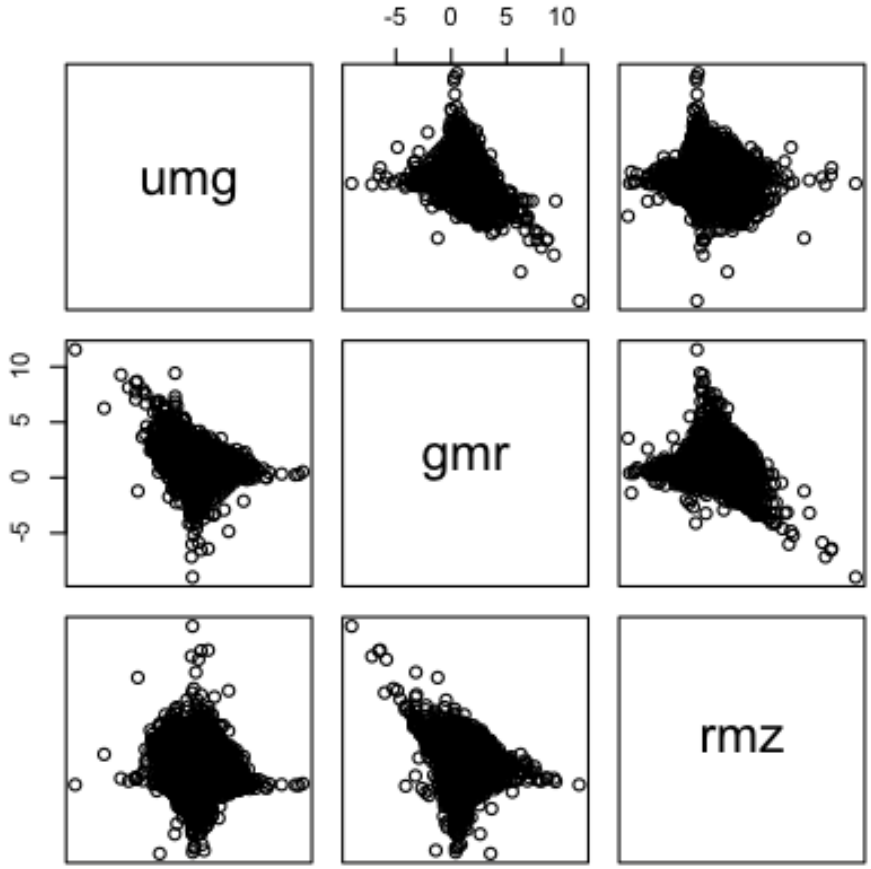
read.csv, read.csv2, read.delim, read.delim2

colors and classification

- d2 = read.table("dataset2.Rframe",header=TRUE)
- objects()
- names(d2)
- umg
- d2$umg
- attach(d2)
- umg
- plot(umg,gmr)
- pairs(d2)
- help(pairs)
- example(pairs)

Reading from a URL and some ops

- d2 = read.table("http://www.astro.caltech.edu/~aam/datasets/SDSS_colors.dat",header=TRUE)
- help(d2)
- names(d2)
- size(d2)
- apropos(d2)
- ??d2
- ?d2
- mean(d2)
- min(d2)
- d2[1,]
- d2[,1]
- d2[1,1]
- d2[0,0]



Next time …

- Built in datasets
- Packages
- Debugging
- Basic plotting