R Introduction 1

J.M. Ponciano

August 17, 2007

1 Objects and types

Location, definition of, and permanence of data

- "Objects" in Splus language include _____,_____and _____.
- Objects are created by assignment statements:

  A vector: `y<-c(38,18,22)`

  A list:

  ```
  x<-list(ncolors=2,colors=c("red","blue"))
  ```

  A function:

  ```
  std.dev<-function(x){return(sqrt(var(x)))}
  ```

- To list objects in current data directory, type _____or _____.
  To narrow search, use _____.
- Cleaning up by removing: _____, or _____the same names.
- Displaying object content:_____
- Executing a function object: _____, and Quitting R _____.

2 Vectors

- element types:_____, _____, _____.

- 4 ways to acces elements:_____,_____,_____, _____.

  Examples: `y<-c(18,32,15,-7,12,19)`

- Matrices: creation, naming, accessing

```
> y<- c(18,32,15,-7,12,19)
> x<- matrix(data=y,nrow=2,ncol=3)
> x
     [,1] [,2] [,3]
[1,]   18   15   12
[2,]   32   -7   19
> z<-matrix(data=y,nrow=2,ncol=3,byrow=T)
> z
     [,1] [,2] [,3]
[1,]   18   32   15
[2,]   -7   12   19

> x<- matrix(data=rep(0,5*8),nrow=5,ncol=8) # a matrix of zeros
> matrix(data=rep(c(10,20,30),times=3,each=3),nrow=3,ncol=3)
     [,1] [,2] [,3]
[1,]   10   20   30
[2,]   10   20   30
[3,]   10   20   30
> matrix(data=rep(c(10,20,30),times=3,each=3),nrow=3,ncol=3,byrow=T)
     [,1] [,2] [,3]
[1,]   10   10   10
[2,]   20   20   20
[3,]   30   30   30
```

Now recall that we did `x<-matrix(data=y,nrow=2,ncol=3)`. Now type:

```
> x<-matrix(data=y,nrow=2,ncol=3)
> x
     [,1] [,2] [,3]
[1,]   18   15   12
[2,]   32   -7   19
> which(x<0,arr.ind=TRUE)
     row col
[1,]   2   2
> a <- which(x<0,arr.ind=TRUE) # which element of the matrix 'x' is negative?
> a # here's the location of the negative element in a
     row col
[1,]   2   2
> a[1] # object 'a' is a 2 elements vector
[1] 2
> a[2]
[1] 2
> x[a] <- 8 # replace -7 in 'x' by 8
> x
     [,1] [,2] [,3]
[1,]   18   15   12
[2,]   32    8   19
```

- Arrays: creation, naming, accessing

```
> y<-c(1:8,11:18,111:118)
> x<-array(y,dim=c(2,4,3))
```

- Factors: creation, accessing

```
colors <- c("red","red","green","yellow","yellow","blue")
colors <- factor(x=colors,levels=c("red","green","yellow","blue"))
table(colors)
```

- vector *atributes*: always either one of 2: _____(numeric,logical,etc...) and _____.
- Some special 'values' and coercion
  - Missing values:

- infinite values:
- "place" holder value:
- logical check for type:
- coercion to type:

- Arithmetic and logical operations and operators
  - Precedence (See Venables and Ripley), from highest to lowest:

```
function(..)
^                   #exponentiation
* /                  #multiplication, division
+ -                 #addition, subtraction
> < <= == != #comparison operators
```

    but can always use parentheses to force a precendence (best)

```
x <- 36
y <- 7
sqrt(x)*7-2 <  x+y^2
(((sqrt(x))*7)-2) < (x+(y^2))  #identical to above but "clear" precedence
```

  - element by element operations on two vectors

```
x <- 1:10
y <- 11:20
```

  - recycling:

– Integer division and modulo reduction:

```
7 %/% 3       #answer 2, number of times 3 goes into 7
7 %%  3       #answer 1, 7 - (7 %/% 3)*3
```

for example, to pick up each 3rd element in x, try:

```
> x<-1:20
> x%%3
 [1] 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2
> x[x%%3==0] <- 6 # replace each third element in x by 6
> x
 [1]  1  2  6  4  5  6  7  8  6 10 11  6 13 14  6 16 17  6 19 20
```

– Logical expressions and functions with logical arguments

```
x <- c(1,3,5)
y <- c(2,2,4)


x < y
x <= y
x != y


any(x<y)  #if any term in vector x<y is T, return T
all(x<y)  #if all terms in vector x<y is T, return T


(x==3 | y==2)  # union of vector x==3 with vector y==2
               # T|T or T|F or F|T returns T
               # only F|F returns F


(x==3 & y==2)  # intersection of vector x==3 with vector y==2
               # only T&T returns T#
```

3 Lists

- assignment
- accessing components
- accessing elements within components
- unlisting

- Special case: data.frame list object-creation, access and extension

```
muscle        <- rep(seq(0,19,by=2),2);
sex              <- factor(rep(c("M","F"),c(10,10)));
speed          <- rep(0,20); # a vector of zeros
speed[1:10]   <-  rnorm(n=10,mean=(30+2*muscle[1:10]),sd=0.1) # filling-in speed
speed[11:20] <- rnorm(10,(40+2*muscle[11:20]),0.1)
mydata        <- data.frame(y=speed,x1=muscle,x2=sex)
print(mydata)
names(mydata)
temp            <- lm(y~x1+x2,data=mydata)
summary(temp)
# attaching the dataframe:
> names(mydata)
[1] "y"  "x1" "x2"
> x1
Error: object "x1" not found
> y
Error: object "y" not found
> x2
Error: object "x2" not found
> attach(mydata)
> y
 [1] 29.91764 34.00737 38.07257 41.95327 45.91410 50.11541
 [7] 53.98323 57.77561 61.82590 65.98829 39.91799 44.05325
[13] 47.87027 52.00124 55.91240 59.93873 63.90140 67.96487
[19] 71.90786 75.91131
> x1
 [1]  0  2  4  6  8 10 12 14 16 18  0  2  4  6  8 10 12 14 16 18
> x2
 [1] M M M M M M M M M M F F F F F F F F F F
Levels: F M

> levels(x2) # asking R what are the levels of x2:
[1] "F" "M"
> detach("mydata")
```

4 Functions

- Creation: specified or unspecified arguments

  unspecified

  specified

- Argument passing for specified case (names,order,default)

5 Simple Functions

- element by element: (same as before, e.g. define `x<-3:12` and add a number to it)
- operate on entire vector and return a scalar: `mean(x)`
- operate on entire vector and return a vector:

  ```
  cumsum(x)
  cumprod(x)
  ```

- conversion to integers, rounding, ...

  ```
  x<-103.652
  ```

- extreme values:

  ```
  x <- c(3,5,5,9); y <- c(5,3,5,11)
  ```

- ordering

```
x <- c(12,5,6,6,-1)
```

- simple statistics

```
x <- c(8,12,5,4,19,2,1)
y <- rnorm(length(x), mean=3+2*x,sd=1)
```

6 Univariate random variable generators, etc

- random number seed; fix for reproducible simulations
- random sampling: `z <- c(18,22,14,25,36)`

- univariate random variable generation: e.g, uniform, binomial, Poisson, Normal, log-normal, Gamma, F, Chi-square

- percentages and percentiles

- probability density functions or probability mass functions

7 Some useful links

Well, besides the links in the class web page, now a days, basically you can "google" any question you have in R and you will get an answer. That's one of the great advantages of R.