URJIT UJWAL TEMBHURNIKAR

17070122069

SDL Assignment

Exercise - Basic Operarions on R

In [4]:
```
#The screen prompt > is an invitation to put R to work

log(42/7.3)
```

1.74979527012902

In [6]:
```
#Two or more expressions can be placed on a single line so long as they are separ

2+3; 5*7; 3-7
```

5

35

-4

Statastics

- Statistics for relatively advanced users: R has thousands of packages, designed, maintained, and widely used by statisticians.
  - Statistical graphics: try doing some of our plots in Stata and you won't have much fun.
- Flexible code: R has a rather liberal syntax, and variables don't need to be declared as they would in (for example) C++, which makes it very easy to code in.
- This also has disadvantages in terms of how safe the code is.
- Vectorization: R is designed to make it very easy to write functions which are applied pointwise to every element of a vector. This is extremely useful in statistics.
- R is powerful: if a command doesn't exist already, you can code it yourself

Basic Operations

In [7]:
```
3+4
```

7

In [8]:
```
345*5
```

1725

In [9]:
```
234/9
```

26

In [10]:
```
x<-15
p<-x+10
print(p)
```

```
[1] 25
```

In [11]:
```
### Conditions
```

In [13]:
```
x+4 <- 15
print(x)
```

```
Error in x + 4 <- 15: could not find function "+<-"
Traceback:
```

In [14]:
```
x = 5
5*x -> x
```

In [15]:
```
x
```

25

Vectors

The key feature which makes R very useful for statistics is that it is vectorized. This means that many operations can be performed point-wise on a vector. The function c() is used to create vectors:

In [17]:
```
x <- c(1.3, 345, -123, 4)
```

In [18]:
```
x
```

1.3 345 -123 4

In [19]:
```
x+2
```

3.3 347 -121 6

In [20]:
```
x*2
```

2.6 690 -246 8

In [21]:
```
sum((x-mean(x))*2)
```

2.8421709430404e-14

- Some useful vectors can be created quickly with R. The colon operator is used to generate integer sequences

In [22]:
```
1:10
```

1 2 3 4 5 6 7 8 9 10

In [23]:
```
100:50
```

100 99 98 97 96 95 94 93 92 91 90 89 88 87 86 85 84 83 82 81
80 79 78 77 76 75 74 73 72 71 70 69 68 67 66 65 64 63 62 61 60
59 58 57 56 55 54 53 52 51 50

In [24]:
```
-3:4
```

-3 -2 -1 0 1 2 3 4

- More generally, the function seq() can generate any arithmetic progression.

In [26]:
```
seq(from =2, to=4, by=1)
```

2 3 4

In [28]:
```
seq(from =-1, to=1, length=6)
```

-1 -0.6 -0.2 0.2 0.6 1

- Sometimes it's necessary to have repeated values, for which we use rep()

In [29]:
```
rep(5,3)
```

5 5 5

In [30]:
```
rep(2:5, each=3)
```

2 2 2 3 3 3 4 4 4 5 5 5

In [31]:
```
2*(0:10)
```

```
0  2  4  6  8  10  12  14  16  18  20
```

In [34]:
```
1:3 + rep(seq(from=0, to=3, by=10), each=3)
```

```
1  2  3
```

- The last example demonstrates recycling, which is also an important part of vectorization.
- If we perform a binary operation (such as +) on two vectors of different lengths, the shorter one is used over and over again until the operation has been applied to every entry in the longer one.
- If the longer length is not a multiple of the shorter length, a warning is given.

In [36]:
```
1:10 * c(-1,1)
```

```
-1  2  -3  4  -5  6  -7  8  -9  10
```

In [37]:
```
1:7*1:4
```

```
Warning message in 1:7 * 1:4:
"longer object length is not a multiple of shorter object length"
```

```
1  4  9  16  5  12  21
```

Vector calculus

In [39]:
```
apples <-4
oranges<-6
```

In [40]:
```
apples+oranges
```

```
10
```

In [41]:
```
earnings<-c(50,60,100)
```

In [42]:
```
earnings*3
```

```
150  180  300
```

In [44]:
```
expences<-c(20,30,40)
```

In [45]:
```
earnings-expences
```

```
30  30  60
```

In [46]:
```
savings<-earnings-expences
```

In [47]:
```
savings
```

```
30  30  60
```

sub setting

In [48]:
```
# Create vector
x <- 1:15

# Print vector
cat("Original vector: ", x, "\n")

# Subsetting vector
cat("First 5 values of vector: ", x[1:5], "\n")

cat("Without values present at index 1, 2 and 3: ",
                                x[-c(1, 2, 3)], "\n")
```

```
Original vector:  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
First 5 values of vector:  1 2 3 4 5
Without values present at index 1, 2 and 3:  4 5 6 7 8 9 10 11 12 13 14 15
```

In [49]:
```
# Dataset
cat("Original dataset: \n")
print(mtcars)

# Subsetting data frame
cat("HP values of all cars:\n")
print(mtcars['hp'])

# First 10 cars
cat("Without mpg and cyl column:\n")
print(mtcars[1:10, -c(1, 2)])
```

```
Original dataset:
                     mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
HP values of all cars:
                    hp
Mazda RX4          110
Mazda RX4 Wag      110
Datsun 710          93
Hornet 4 Drive     110
Hornet Sportabout  175
Valiant            105
Duster 360         245
Merc 240D           62
Merc 230            95
```

```
Merc 280            123
Merc 280C           123
Merc 450SE          180
Merc 450SL          180
Merc 450SLC         180
Cadillac Fleetwood  205
Lincoln Continental 215
Chrysler Imperial   230
Fiat 128             66
Honda Civic          52
Toyota Corolla       65
Toyota Corona        97
Dodge Challenger    150
AMC Javelin         150
Camaro Z28          245
Pontiac Firebird    175
Fiat X1-9            66
Porsche 914-2        91
Lotus Europa        113
Ford Pantera L      264
Ferrari Dino        175
Maserati Bora       335
Volvo 142E          109
Without mpg and cyl column:
                   disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        108.0  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    258.0 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 360.0 175 3.15 3.440 17.02  0  0    3    2
Valiant           225.0 105 2.76 3.460 20.22  1  0    3    1
Duster 360        360.0 245 3.21 3.570 15.84  0  0    3    4
Merc 240D         146.7  62 3.69 3.190 20.00  1  0    4    2
Merc 230          140.8  95 3.92 3.150 22.90  1  0    4    2
Merc 280          167.6 123 3.92 3.440 18.30  1  0    4    4
```

Subset vector in R

In [50]:
```
my_vector <- c(15, 21, 17, 25, 12, 51)
```

In [51]:
```r
# Returns the full vector
my_vector[]

# Third value
my_vector[3]

# Third value, simplified
my_vector[[3]]

# Elements one to three
my_vector[1:3]

# Second and fifth elements
my_vector[c(2, 5)]

# Second element twice
my_vector[c(2, 2)]

# All values except the fourth
my_vector[-4]
```

```
15  21  17  25  12  51

17

17

15  21  17

21  12

21  21

15  21  17  12  51
```

Subset a matrix in R

In [52]:
```r
set.seed(45)

my_matrix <- matrix(sample(1:9), ncol = 3)
colnames(my_matrix) <- c("one", "two", "three")
my_matrix
```

one	two	three
5	2	4
3	8	7
6	9	1

In [53]:
```
# Subset matrix with rows and columns index
my_matrix[c(1, 3), c(1, 2)]

# Subset with logical values
my_matrix[c(TRUE, FALSE, TRUE), c(TRUE, TRUE, FALSE)] # Equivalent

# You can also mix
my_matrix[c(1, 3), c(TRUE, TRUE, FALSE)] # Equivalent
```

one	two
5	2
6	9

one	two
5	2
6	9

one	two
5	2
6	9

Subsetting a list in R

In [54]:
```
my_list <- list(1:10, c(TRUE, FALSE), 1)
```

In [55]:
```
# Second object of the list
my_list[2]

# Second object of the list, simplified
my_list[[2]]

# Second object simplified, first element
my_list[[2]][1]

# Second object, first element, all simplified
my_list[[2]][[1]]
```

1. TRUE FALSE

TRUE FALSE

TRUE

TRUE

In [56]:
```
my_named_list <- list(x = 1:10, y = c(TRUE, FALSE), z = 1)
```

In [57]:
```
# First element
my_named_list["x"]
my_named_list$x # Equivalent

# Second element, simplified
my_named_list[["y"]]
```

$x =

1 2 3 4 5 6 7 8 9 10

1 2 3 4 5 6 7 8 9 10

TRUE FALSE

In [58]:
```
my_list[[1]] <- subset(my_list[[1]], my_list[[1]] > 5)
my_list
```

1. 6 7 8 9 10
2. TRUE FALSE
3. 1

* R Datatypes

* Vectors

* Lists

* Matrices

* Arrays

* Data Frames

In [60]:
```
getwd()
setwd("E:/R Basic/")
##R variables
x<- 10L #integer
class(x)
x<- 'a'#character
x<-"ab"
x<- 2.1#numeric
x<- TRUE #Logical
x<- 1+2i #Complex
is.complex(x)
x<- as.integer(x)
print(x)
class(x)
```

'C:/Users/atharvtembhurnikar'

```
Error in setwd("E:/R Basic/"): cannot change working directory
Traceback:

1. setwd("E:/R Basic/")
```

In [61]:
```
#VECTOR
v<-c(1,2,3,4)
print(v)
class(v)
v<-c('a','b','c')
v<-c(1,'a',1.2)
v[2]
```

```
[1] 1 2 3 4
```

'numeric'

'a'

In [62]:
```r
#combining vector
s<- c(1,2,3)
n<- c('a','b','c')
ns<- c(n,s)
ns
class(ns)
vec<- c(1,1.2,"a")
class(vec)
```

'a' 'b' 'c' '1' '2' '3'

'character'

'character'

In [63]:
```
#Vector arithmetic
p<-c (10,20,30)
s*p
s/p
s+p
s-p
m<-c(10,20,30,40,50,60)
p+m #Recycle
#accessing vector element
m[3]
m[-3] # except element at 3
m[10]
m[1:3]
m[c(1,3,5)]
#named vector
names(v)= c("First","second","Third")
v
v["First"]
```

10 40 90

0.1 0.1 0.1

11 22 33

-9 -18 -27

20 40 60 50 70 90

30

10 20 40 50 60

<NA>

10 20 30

10 30 50

First	'1'
second	'a'
Third	'1.2'

First: '1'

List

In [65]:
```
l<-list(1,2,3)
v<-list(1,'a',1.2)
s<- c("hi","how r u")
class(l)
l
print(l)
lis<- list(l,v,1,2,3,s)
lis[2]
lis[[2]]
names(lis)<- c('1','2','3','4','5','6')
lis$`2`[2]
lis$`6`
named_list <- list(x=c(1,2,3),y=c('1','2','3'))
names(named_list)
named_list$x
```

'list'

1. 1
2. 2
3. 3

```
[[1]]
[1] 1

[[2]]
[1] 2

[[3]]
[1] 3
```

1. A. 1
   B. 'a'
   C. 1.2

1. 1
2. 'a'
3. 1.2

1. 'a'

'hi' 'how r u'

'x' 'y'

1 2 3

END

In [ ]: