1. di<-seq(-2,2,4/((NI+3)))

d<-di[-seq (11,13,1)] ##--In the sequence di, we omit No. 11-13

d<-di[-seq(11,14,2)] ##--In the sequence di, we omit every other element from No.11-14 (omitting 11 & 13)

1. **R while Loop**

**一种循环；只要 i 符合某个条件就不断进行计算**

while() loop will execute a block of commands until the condition is no longer satisfied.

while(cond) expr

cond: condition  
expr: expression

> x <- 1

> while(x < 5) {x <- x+1; print(x);}

[1] 2

[1] 3

[1] 4

next can skip one step of the loop.  
break will end the loop abruptly.  
  
Let's break the loop when x=3:

> x <- 1

> while(x < 5) {x <- x+1; if (x == 3) break; print(x); }

[1] 2

Let's skip one step when x=3:

> x <- 1

> while(x < 5) {x <- x+1; if (x == 3) next; print(x);}

[1] 2

[1] 4

[1] 5

### R unlist Function

**就是把一个 vector 里所有的元素都列出来。**

unlist(x) function simplifies a list to produce a vector which contains all the atomic components which occur in x.

unlist(x, recursive = TRUE, use.names = TRUE)

x: list or vector  
recursive: logical, should unlisting be applied to list components of x  
use.names: logical, should names be preserved

> BOD

Time demand

1 1 8.3

2 2 10.3

3 3 19.0

4 4 16.0

5 5 15.6

6 7 19.8

> is.list(BOD)

[1] TRUE

> unlist(BOD)

Time1 Time2 Time3 Time4 Time5 Time6 demand1

1.0 2.0 3.0 4.0 5.0 7.0 8.3

demand2 demand3 demand4 demand5 demand6

10.3 19.0 16.0 15.6 19.8

> unlist(BOD,use.names=FALSE)

[1] 1.0 2.0 3.0 4.0 5.0 7.0 8.3 10.3 19.0 16.0 15.6 19.8

### R apply Function

其实就是把一个 function 应用到所有选中的列、行或者 array 上去。但是 sapply 和 lapply 给出的结果的格式会有所不同。Lapply 列出来，sapply变成vector 列出来。Array 可以命令 R 对一串数字进行排列，排成矩阵也可以。

apply() function applies a function to margins of an array or matrix.

apply(x,margin,func, ...)

• x: array  
• margin: subscripts, for matrix, 1 for row, 2 for column  
• func: the function  
...

>BOD #R built-in dataset, Biochemical Oxygen Demand

Time demand

1 1 8.3

2 2 10.3

3 3 19.0

4 4 16.0

5 5 15.6

6 7 19.8

Sum up for each row:

> apply(BOD,1,sum)

[1] 9.3 12.3 22.0 20.0 20.6 26.8

Sum up for each column:

> apply(BOD,2,sum)

Time demand

22 89

Multipy all values by 10:

> apply(BOD,1:2,function(x) 10 \* x)

Time demand

[1,] 10 83

[2,] 20 103

[3,] 30 190

[4,] 40 160

[5,] 50 156

[6,] 70 198

Used for array, margin set to 1:

> x <- array(1:9)

> apply(x,1,function(x) x \* 10)

[1] 10 20 30 40 50 60 70 80 90

Two dimension array, margin can be 1 or 2:

> x <- array(1:9,c(3,3))

> x

[,1] [,2] [,3]

[1,] 1 4 7

[2,] 2 5 8

[3,] 3 6 9

> apply(x,1,function(x) x \* 10) #or apply(x,2,function(x) x \* 10)

[1] 10 20 30 40 50 60 70 80 90

lapply() function can handle data frame with similar results, return is a list:

> lapply(BOD,sum)

$Time

[1] 22

$demand

[1] 89

> lapply(BOD,mean)

$Time

[1] 3.666667

$demand

[1] 14.83333

sapply() has similar function, it defines "simplify=TRUE" by default, thus return a vector:

> sapply(BOD,sum)

Time demand

22 89

> sapply(BOD,sum,simplify=FALSE)

$Time

[1] 22

$demand

[1] 89