Introduction to R Software

Basics of Calculations

::::

Conditional Executions and Loops

Shalabh

Department of Mathematics and Statistics

Indian Institute of Technology Kanpur

1. Conditional execution

Syntax

```
if ( condition ) {executed commands if condition is
                                        TRUE}
if ( condition ) {executed commands if condition is
                                        TRUE}
else { executed commands if condition is FALSE }
```

2. Conditional execution

Syntax

```
ifelse(test, yes, no)
```

- **Vector-valued evaluation of conditions .**
- **For the components in the vector-valued logical expression `test` which provide the value `TRUE`, the operations given by `yes` are executed.**
- **For the components in the vector-valued logical expression `test` which provide the value `FALSE`, the operations given by `no` are executed.**

2. Conditional execution

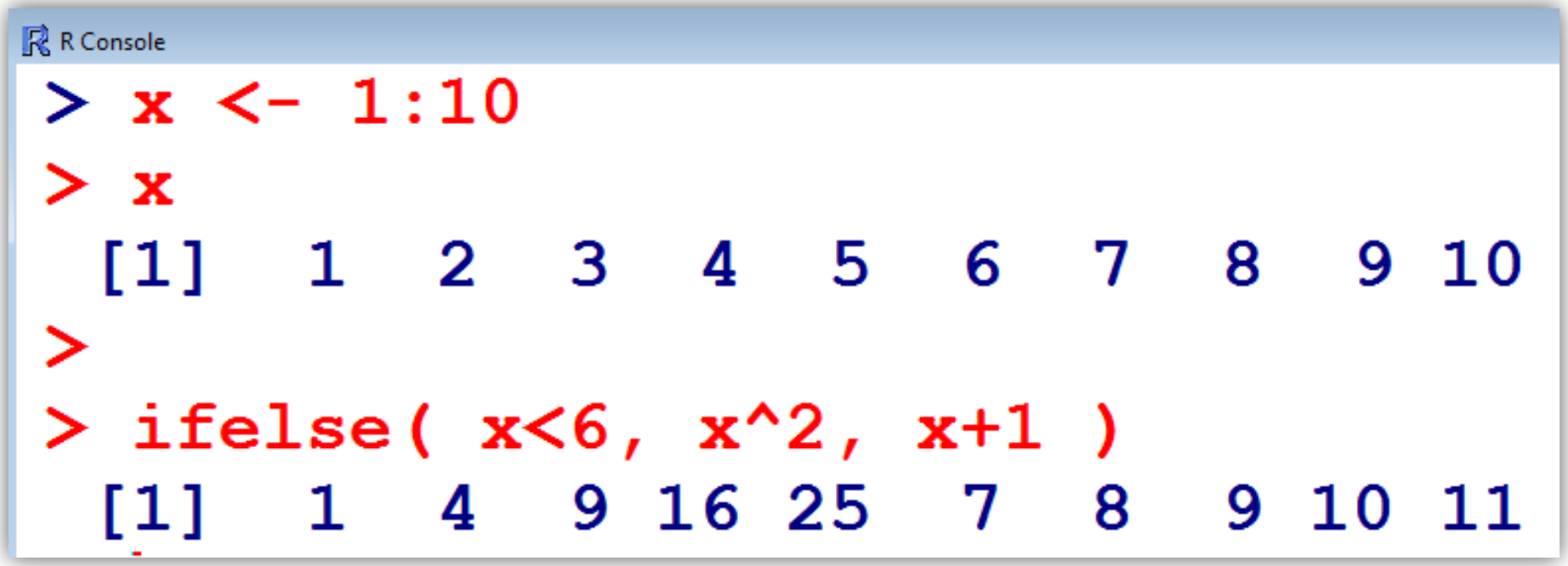
Example

```
> x <- 1:10
>x
 [1] 1 2 3 4 5 6 7 8 9 10
> ifelse( x<6, x^2, x+1 )
 [1] 1 4 9 16 25 7 8 9 10 11
```

Interpretation

- If $x < 6$ (TRUE), then $x = x^2$ (YES) .
- If $x \geq 6$ (FALSE), then $x = x + 1$ (NO).

- So for $x = 1, 2, 3, 4, 5$, we get $x = x^2 = 1, 4, 9, 16, 25$
- For $x = 6, 7, 8, 9, 10$, we get $x = x+1 = 7, 8, 9, 0, 11$


R Console
> x <- 1:10
> x
[1] 1 2 3 4 5 6 7 8 9 10
>
> ifelse(x<6, x^2, x+1)
[1] 1 4 9 16 25 7 8 9 10 11

Control structures in R :

Loops

Repetitive commands are executed by loops

- **for loop**
- **while loop**
- **repeat loop**

1. The `for` loop

If the number of repetitions is known in advance (e.g. if all commands have to be executed for all cases $i = 1,2,\ldots,n$ in the data), a `for()` loop can be used.

Syntax

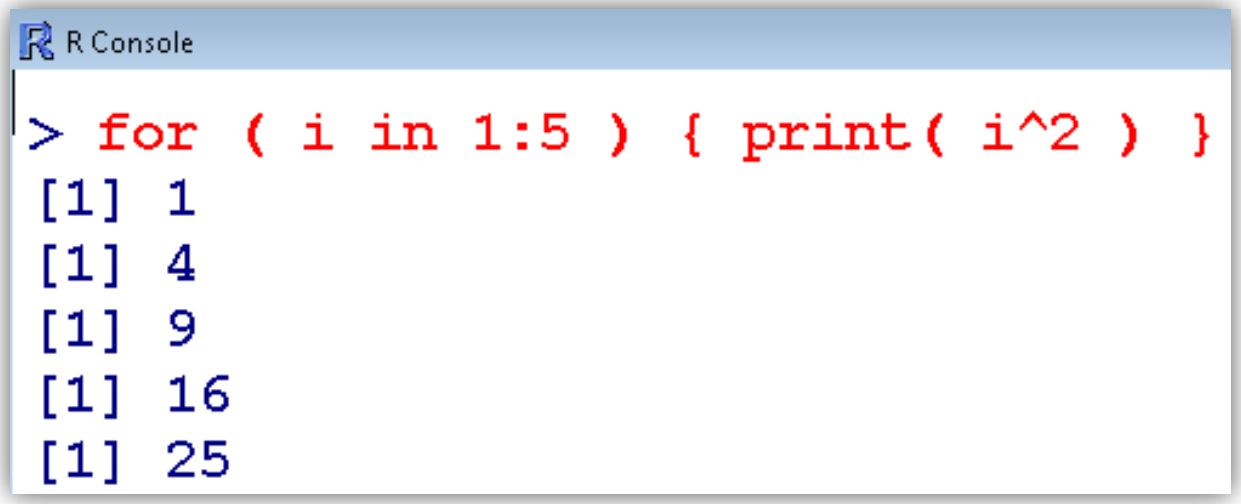
```
for (name in vector) {commands to be executed}
```

A variable with name `name` is sequentially set to all values, which contained in the vector vector.

All operations/commands are executed for all these values.

Example

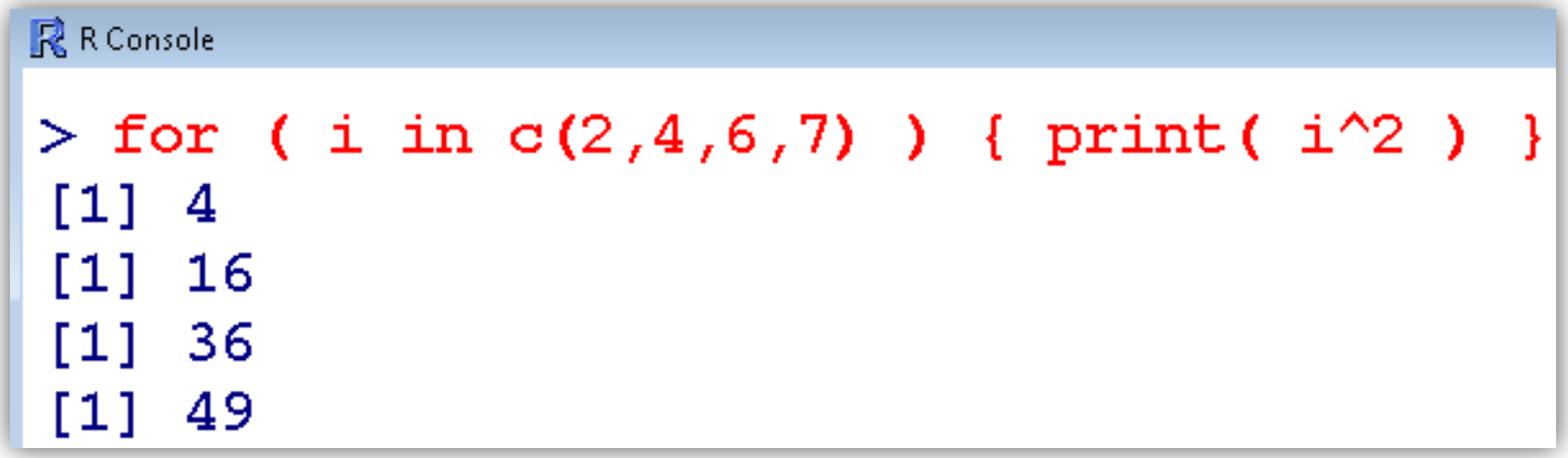
```
> for ( i in 1:5 ) { print( i^2 ) }
[1] 1
[1] 4
[1] 9
[1] 16
[1] 25
```



Example

Note: `print` **is a function to print the argument**

```
> for ( i in c(2,4,6,7) ) { print( i^2 ) }
[1] 4
[1] 16
[1] 36
[1] 49
```



2. The `while()` loop

If the number of loops is not known in before, e.g. when an iterative algorithm to maximize a likelihood function is used, one can use a `while()` loop.

Syntax

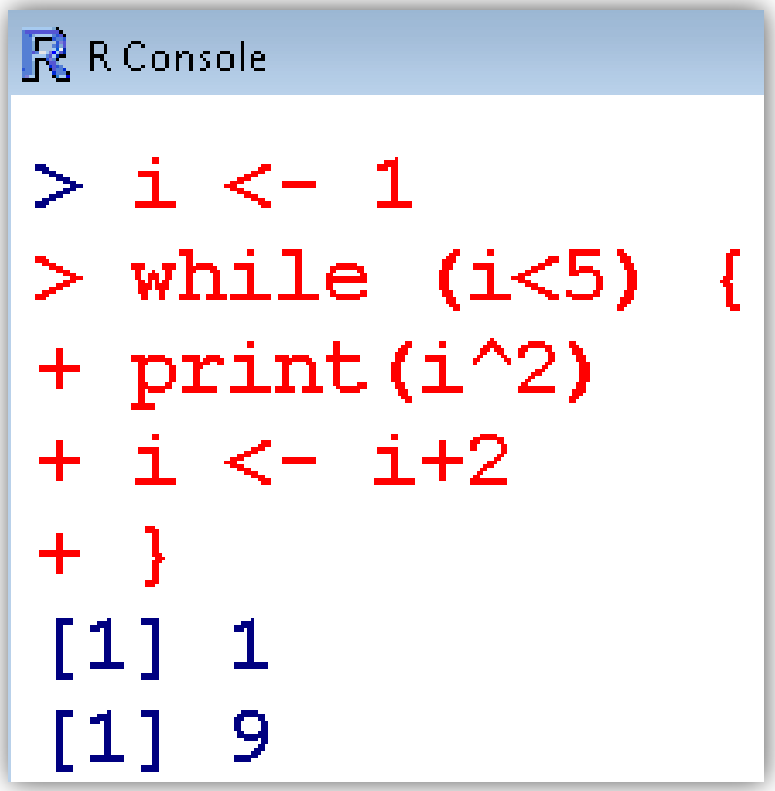
```
while(condition){ commands to be executed as
long as condition is TRUE }
```

If the condition is not true *before entering* the loop, no commands within the loop are executed.

Example

```
> i <- 1
> while (i<5) {
+ print(i^2)
+ i <- i+2
+}
[1] 1
[1] 9
```

The programmer itself has to be careful that the counting variable `i` within the loop is incremented. Otherwise an infinite loop occurs.


R Console
> i <- 1
> while (i<5) {
+ print(i^2)
+ i <- i+2
+ }
[1] 1
[1] 9

3. The `repeat` loop

The repeat loop doesn't test any condition — in contrast to the `while()` loop — *before entering* the loop and also not during the execution of the loop.

Again, the programmer is responsible that the loop terminates after the appropriate number of iterations. For this the `break` command can be used.

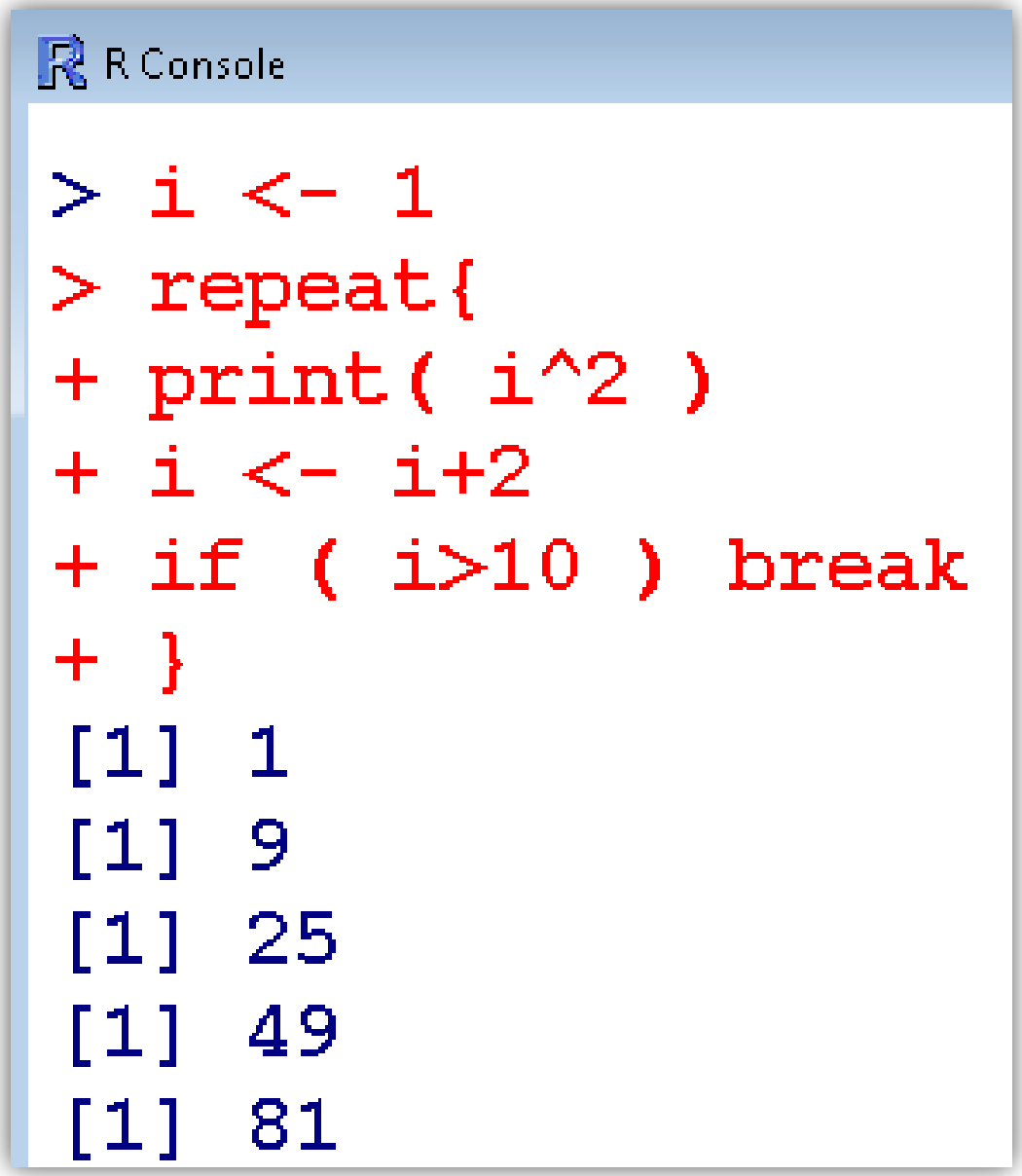
Syntax

```
repeat{ commands to be executed }
```

Example:

```
> i <- 1

> repeat{
  + print( i^2 )
  + i <- i+2
  + if ( i > 10 ) break
  +}
[1] 1
[1] 9
[1] 25
[1] 49
[1] 81
```


R Console
> i <- 1
> repeat{
+ print(i^2)
+ i <- i+2
+ if (i>10) break
+ }
[1] 1
[1] 9
[1] 25
[1] 49
[1] 81

Example:

Additionally, the command `next` is available, to return to the beginning of the loop (to return to the first command in the loop).

```
> i <- 1

> repeat{
  + i <- i+1
  + if (i < 10) next
  + print(i^2)
  + if (i >= 13) break
  +}
[1] 100
[1] 121
[1] 144
[1] 169
```