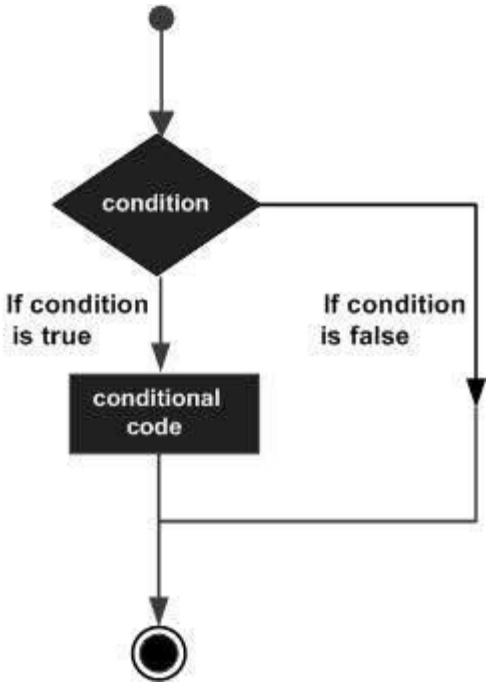
R - Decision making

Decision making structures require the programmer to specify one or more conditions to be evaluated or tested by the program, along with a statement or statements to be executed if the condition is determined to be **true**, and optionally, other statements to be executed if the condition is determined to be **false**.

Following is the general form of a typical decision making structure found in most of the programming languages −



R provides the following types of decision-making statements. Click the following links to check their detail.

Sr.No.	Statement & Description
1	**if statement** An **if** statement consists of a Boolean expression followed by one or more statements.
2	**if...else statement** An **if** statement can be followed by an optional **else** statement, which executes when the Boolean expression is false.
3	**switch statement** A **switch** statement allows a variable to be tested for equality against a list of values.

R – IF Statement

An **if** statement consists of a Boolean expression followed by one or more statements.

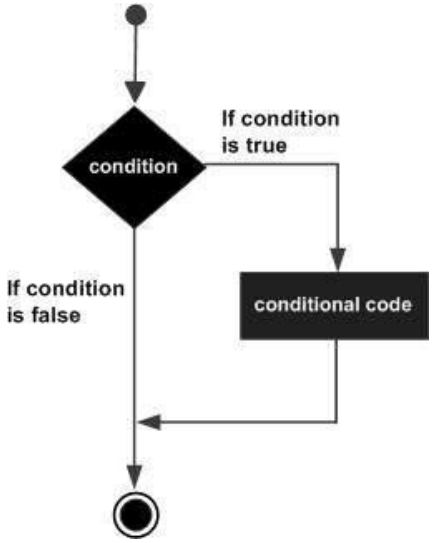
Syntax

The basic syntax for creating an **if** statement in R is −

```
if(boolean_expression) {
   // statement(s) will execute if the boolean expression is true.
}
```

If the Boolean expression evaluates to be **true**, then the block of code inside the if statement will be executed. If Boolean expression evaluates to be **false**, then the first set of code after the end of the if statement (after the closing curly brace) will be executed.

Flow Diagram



Example

```
x <- 30L

if(is.integer(x)) {

   print("X is an Integer")

}
```

When the above code is compiled and executed, it produces the following result −

```
[1] "X is an Integer"
```

R – IF ELSE Statement

An **if** statement can be followed by an optional **else** statement which executes when the boolean expression is false.

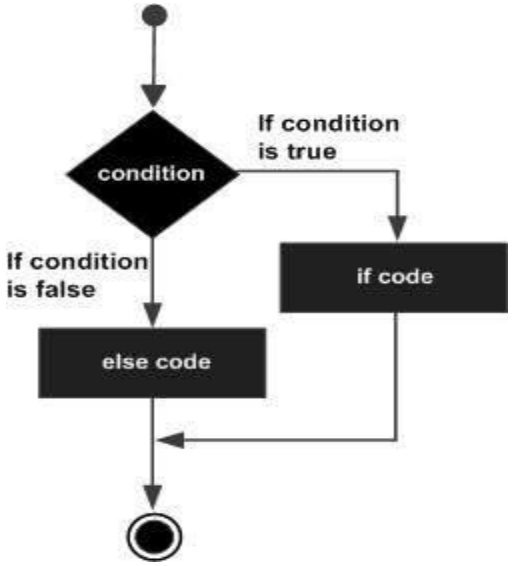
Syntax

The basic syntax for creating an **if...else** statement in R is −

```
if(boolean_expression) {
   // statement(s) will execute if the boolean expression is true.
} else {
   // statement(s) will execute if the boolean expression is false.
}
```

If the Boolean expression evaluates to be **true**, then the **if block** of code will be executed, otherwise **else block** of code will be executed.

Flow Diagram



Example

```
x <- c("what","is","truth")

if("Truth" %in% x) {

   print("Truth is found")

} else {

   print("Truth is not found")

}
```

When the above code is compiled and executed, it produces the following result −

```
[1] "Truth is not found"
```

Here "Truth" and "truth" are two different strings.

The if...else if...else Statement

An **if** statement can be followed by an optional **else if...else** statement, which is very useful to test various conditions using single if...else if statement.

When using **if**, **else if**, **else** statements there are few points to keep in mind.

- An **if** can have zero or one **else** and it must come after any **else if**'s.
- An **if** can have zero to many **else if's** and they must come before the else.
- Once an **else if** succeeds, none of the remaining **else if**'s or **else**'s will be tested.

Syntax

The basic syntax for creating an **if...else if...else** statement in R is −

```
if(boolean_expression 1) {
   // Executes when the boolean expression 1 is true.
} else if( boolean_expression 2) {
   // Executes when the boolean expression 2 is true.
} else if( boolean_expression 3) {
   // Executes when the boolean expression 3 is true.
} else {
   // executes when none of the above condition is true.
}
```

Example

```
x <- c("what","is","truth")

if("Truth" %in% x) {

   print("Truth is found the first time")

} else if ("truth" %in% x) {

   print("truth is found the second time")

} else {

   print("No truth found")

}
```

When the above code is compiled and executed, it produces the following result −

```
[1] "truth is found the second time"
```

R – SWITCH Statement

A **switch** statement allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each case.

Syntax

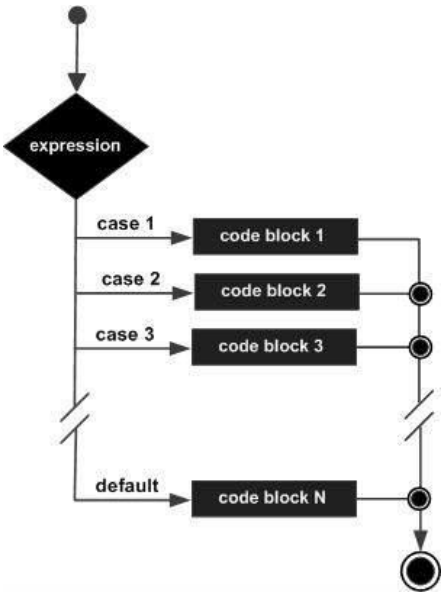
The basic syntax for creating a switch statement in R is −

```
switch(expression, case1, case2, case3....)
```

The following rules apply to a switch statement −

- If the value of expression is not a character string it is coerced to integer.
- You can have any number of case statements within a switch. Each case is followed by the value to be compared to and a colon.
- If the value of the integer is between 1 and nargs()−1 (The max number of arguments)then the corresponding element of case condition is evaluated and the result returned.
- If expression evaluates to a character string then that string is matched (exactly) to the names of the elements.
- If there is more than one match, the first matching element is returned.
- No Default argument is available.
- In the case of no match, if there is a unnamed element of ... its value is returned. (If there is more than one such argument an error is returned.)

Flow Diagram



Example

```
x <- switch(
   3,
   "first",
   "second",
   "third",
   "fourth"
)
print(x)
```

When the above code is compiled and executed, it produces the following result −

```
[1] "third"
```