

## Syntax

```
name.of.function <- function(params) {  
  
}
```

```
func1 <- function() {  
  r <- sqrt(7)  
  pi * r^2  
}  
  
func1()
```

What happens if I want to change the values?

```
func2 <- function(radius) {  
  pi * radius^2  
}  
  
func2(10)
```

What happens if I don't pass any value?

```
#func2()  
  
func3 <- function(radius = 7) {  
  pi * radius^2  
}  
  
func3() ## picks default  
  
func3(8) ## uses the passed value
```

## Returning values

### implicitly

```
add.nos <- function(a,b) {  
  a + b  
}  
  
ret.val1 <- add.nos(5,8)  
ret.val1  
  
ret.val2 <- add.nos(15,20)  
ret.val2
```

### explicitly using return statement

```
key.func <- function(a = 10, b = 50, c = 20, key = 1) {  
  
  if(key == 1) {
```

```
    ret.val <- a + b
  }
  else {
    ret.val <- b + c
  }

  return(ret.val)
}

key.func()

key.func(key=0)
```

---

## Control Structures

---

### 1. if

```
2. if(TRUE) {
3.   do something
4. }

func.test <- function(mean = 10) {
  ret.val <- 0

  if(mean > 5) {
    ret.val <- 20
  }
  return(ret.val)
}

func.test()

func.test(mean = 1)
```

### 1. if-else

```
2. if(TRUE) {
3.   do something
4. } else {
5.   do something else
6. }

if(7 > 10) {
  print("Are you serious?")
} else {
  print("Yeah, thought so.")
}
```

### 1. ifelse

```
2. ifelse(CONDITION, IF TRUE, IF FALSE)
```

```
values <- c(0.5,1,-1)

values <- ifelse(values > 0.5, 1, -1)

table(values)
```

Use it with something cool

```
values <- ifelse(sample(0:1,10000,replace = T) > 0.5, 1, -1)

plot(cumsum(values), type = "l", col = "red")
lines(c(0,10000),c(0,0))
```

### 1. for

```
2. for(counter in vector) {
3.     iterate and do something
4. }
```

Lets play Scrabble

```
alphabets <- sample(LETTERS, 7, replace = T)

for(i in 1:7) {
  print(alphabets[i])
}
```

### 1. while

```
2. while(CONDITION IS TRUE) {
3.     do something
4. }

num <- 10

while(num > 0) {
  print(num)
  num <- num - 1 ## Be careful about iterator
}
```

---

## apply family

### apply

- Syntax: `apply(X, MARGIN, FUN)`

```
mat <- matrix(round(runif(50,-10,10),0), nrow = 5)
mat

apply(mat, 2, max)
```

```
apply(mat, 1, max)
```

### **lapply and sapply**

```
list1 <- list(num = 10,  
             truefalse = ifelse(runif(10,0,1)>.5,T,F),  
             colors = c("red", "blue", "green", "yellow"))
```

```
list1
```

```
lapply(list1, length)
```

```
sapply(list1, length)
```

### **tapply**

```
library(ISLR)  
data("Hitters")
```

```
head(Hitters)
```

```
hitters.complete <- Hitters[complete.cases(Hitters),]
```

```
length(hitters.complete[,1])
```

```
tapply(hitters.complete$Salary,  
       list(hitters.complete$Division, hitters.complete$League),  
       sum)
```

### **Using dplyr**

```
library(dplyr)  
hitters.complete %>%  
  group_by(Division, League) %>%  
  summarise(s = sum(Salary))
```