SCS 2211 - Laboratory II

Mr. D. T. Bamunuarachchi – dtb@ucsc.cmb.ac.lk
Ms. Piyumi Seneviratne – pws@ucsc.cmb.ac.lk

Recap – Data Types

- **Numeric -** The default type when dealing with numbers.
  - Examples: 1, 1.0, 42.5
- **Integer**
  - Examples: 1L, 2L, 42L
- **Complex**
- Example: 4 + 2i
- **Logical**
  - TRUE and FALSE
- **Character**
  - Examples: "a", "Statistics", "1 plus 2."

Recap – Data Structures

- Vector
- Matrix
- List
- Data Frame
- Array

Recap – Vector Operations

R Console

```
> vec1=1:10
> vec1
 [1]  1  2  3  4  5  6  7  8  9 10
> vec1+1
 [1]  2  3  4  5  6  7  8  9 10 11
> vec1^2
 [1]   1   4   9  16  25  36  49  64  81 100
> vec1*3
 [1]  3  6  9 12 15 18 21 24 27 30
> sqrt(vec1)
 [1] 1.000000 1.414214 1.732051 2.000000 2.236068 2.449490 2.645751 2.828427
 [9] 3.000000 3.162278
>
```

Logical Operators

Operator	Summary	Example	Result
`x < y`	x less than y	`3 < 42`	TRUE
`x > y`	x greater than y	`3 > 42`	FALSE
`x <= y`	x less than or equal to y	`3 <= 42`	TRUE
`x >= y`	x greater than or equal to y	`3 >= 42`	FALSE
`x == y`	xequal to y	`3 == 42`	FALSE
`x != y`	x not equal to y	`3 != 42`	TRUE
`!x`	not x	`!(3 > 42)`	TRUE
`x | y`	x or y	`(3 > 42) | TRUE`	TRUE
`x & y`	x and y	`(3 < 4) & ( 42 > 13)`	TRUE

Logical Operators

```
> x = c(1, 3, 5, 7, 8, 9)
> x > 3
[1] FALSE FALSE  TRUE  TRUE  TRUE  TRUE
> x == 3
[1] FALSE  TRUE FALSE FALSE FALSE FALSE
> x != 3
[1]  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
> x == 3 & x != 3
[1] FALSE FALSE FALSE FALSE FALSE FALSE
> x == 3 | x != 3
[1] TRUE TRUE TRUE TRUE TRUE TRUE
>
```

Try it out!

```
x = c(1, 3, 5, 7, 8, 9)
```

What would be the output of followings?

- `x[x > 3]`
- `x[x != 3]`

Try it out!

x = **c**(1, 3, 5, 7, 8, 9)

What would be the output of followings?

- x[x > 3]
- x[x != 3]

```
> x[x > 3]
[1] 5 7 8 9
> x[x != 3]
[1] 1 5 7 8 9
```

Control Structures

- if and else: testing a condition and acting on it
- for: execute a loop a fixed number of times
- while: execute a loop while a condition is true
- repeat: execute an infinite loop (must break out of it to stop)
- break: break the execution of a loop
- next: skip an iteration of a loop

If/else

```
if (...) {
  some R code
} else {
  more R code
}
```

```
> x = 1
> y = 3
> if (x > y) {
+ z = x * y
+ print("x is larger than y")
+ } else {
+ z = x + 5 * y
+ print("x is less than or equal to y")
+ }
[1] "x is less than or equal to y"
> z
[1] 16
```

If/else/ else if

```
if(<condition1>) {
## do something
} else if(<condition2>) {
## do something different
} else {
## do something different
}
```

for Loop

```
> for(i in 1:10) {
+ print(i)
+ }
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
[1] 10
>
```

```
> x <- c("a", "b", "c", "d")
> for(i in 1:4) {
+ ## Print out each element of 'x'
+ print(x[i])
+ }
[1] "a"
[1] "b"
[1] "c"
[1] "d"
>
```

for Loop

```
> x <- c("a", "b", "c", "d")
> for(i in 1:4) {
+ ## Print out each element of 'x'
+ print(x[i])
+ }
[1] "a"
[1] "b"
[1] "c"
[1] "d"
> for(letter in x) {
+ print(letter)
+ }
[1] "a"
[1] "b"
[1] "c"
[1] "d"
```

Nested for Loop

```
> x <- matrix(1:6, 2, 3)
> for(i in seq_len(nrow(x))) {
+ for(j in seq_len(ncol(x))) {
+ print(x[i, j])
+ }
+ }
[1] 1
[1] 3
[1] 5
[1] 2
[1] 4
[1] 6
```

While Loop

```
> count <- 0
> while(count < 10) {
+ print(count)
+ count <- count + 1
+ }
[1] 0
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
>
```

repeat/break

```
> y <- 12345
> x <- y/2
> repeat{
+ x <- (x + y/x)/2
+ if (abs(x*x-y) < 1e-10) break
+ }
> x
[1] 111.1081
```

Functions

- Functions are defined using the function() directive.
- Functions stored as R objects.
- return() function can be used to return a value from a function

```
> f <- function() {
+ ## This is an empty function
+ }
> ## Functions have their own class
> class(f)
[1] "function"
>
```

Functions

```
> f <- function() {
+ cat("Hello, world!\n")
+ }
> f()
Hello, world!
>
```

Functions

```
> f <- function(num) {
+ hello <- "Hello, world!\n"
+ for(i in seq_len(num)) {
+ cat(hello)
+ }
+ chars <- nchar(hello) * num
+ chars
+ }
> f(5
+ )
Hello, world!
Hello, world!
Hello, world!
Hello, world!
Hello, world!
[1] 70
```

Functions

```
> f2 <- function(num) {
+ txt <- "ABCD\n"
+ for(i in seq_len(num)) {
+ cat(txt)
+ }
+ chars <- nchar(txt) * num
+ chars
+ }
> f2(5)
ABCD
ABCD
ABCD
ABCD
ABCD
[1] 25
```

Try it out!

Write a function that takes temperature in Fahrenheit and return the Celsius temperature.

$$°C = (°F - 32) \times 5/9$$

Loop Functions

- lapply(): Loop over a list and evaluate a function on each element
- sapply(): Same as lapply but try to simplify the result
- apply(): Apply a function over the margins of an array
- tapply(): Apply a function over subsets of a vector
- mapply(): Multivariate version of lapply

Loop Functions

```
> lapply(airquality,mean,na.rm=T)
$Ozone
[1] 42.12931

$Solar.R
[1] 185.9315

$Wind
[1] 9.957516

$Temp
[1] 77.88235

$Month
[1] 6.993464

$Day
[1] 15.80392

> sapply(airquality,mean,na,rm=T)
Error in mean.default(X[[i]], ...) : object 'na' not found
> sapply(airquality,mean,na.rm=T)
     Ozone    Solar.R       Wind       Temp      Month        Day
 42.129310 185.931507   9.957516  77.882353   6.993464  15.803922
>
```