2. Control Structures

Contents

1 2. Control Structures

1.1 Control Structures - Introduction

Control structures in R allow you to control the flow of execution of the program, depending on runtime conditions. Common structures are · if, else: testing a condition · for: execute a loop a fixed number of times · while: execute a loop while a condition is true · repeat: execute an infinite loop · break: break the execution of a loop · next: skip an interation of a loop · return: exit a function Most control structures are not used in interactive sessions, but rather when writing functions or longer expresisons.

1.2 Control Structures - If-else

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

1.2.1 if

This is a valid if/else structure.

```
x <-0
```

```
if(x>3) {
  y <- 10
} else{
```

```
  y <- 0
}
```

So is this one.

```
y <- if (x > 3) {
  10
} else{
  0
}
```

Of course, the else clause is not necessary.

```
if(<condition1>) {

}

if(<condition2>) {

}
```

1.3 Control Structures - For loops

for loops take an interator variable and assign it successive values from a sequence or vector. For loops are most commonly used for iterating over the elements of an object (list, vector, etc.)

```
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
```

This loop takes the i variable and in each iteration of the loop gives it values 1, 2, 3, ..., 10, and then exits.

These three loops have the same behavior.

```
x <- c("a", "b", "c", "d")

for (i in 1:4) {
  print(x[i])
}
```

```
## [1] "a"
## [1] "b"
## [1] "c"
## [1] "d"
```

```
for(i in seq_along(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"
```

```
for(i in 1:4) print(x[i])
```

```
## [1] "a"
## [1] "b"
## [1] "c"
## [1] "d"
```

1.3.1 Nested for loops

for loops can be nested.

```
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
```

Be careful with nesting though. Nesting beyond 2–3 levels is often very difficult to read/understand.

1.4 Control Structures - While loops

While loops begin by testing a condition. If it is true, then they execute the loop body. Once the loop body is executed, the condition is tested again, and so forth.

```
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
```

While loops can potentially result in infinite loops if not written properly. Use with care!

Sometimes there will be more than one condition in the test.

```
z <- 5
while (z >= 3 && z <= 10) {
  print(z)
  coin <- rbinom(1, 1, 0.5)
  if (coin == 1) {
    ## random walk z<-z+1
  } else{
    z <- z - 1
  }
}
```

```
## [1] 5
## [1] 4
## [1] 4
## [1] 4
## [1] 4
## [1] 4
## [1] 4
## [1] 3
```

Conditions are always evaluated from left to right.

1.5 Control Structures - Repeat, Next, Break

1.5.1 repeat

Repeat initiates an infinite loop; these are not commonly used in statistical applications but they do have their uses. The only way to exit a repeat loop is to call break.

```
x0 <- 1
tol <- 1e-8
repeat {
  x1 <- computeEstimate()
  if (abs(x1 - x0) < tol) {
    break
  } else{
    x0 <- x1
  }
}
```

```
## Error in computeEstimate(): could not find function "computeEstimate"
```

The loop in the previous slide is a bit dangerous because there's no guarantee it will stop. Better to set a hard limit on the number of iterations (e.g. using a for loop) and then report whether convergence was achieved or not.

1.5.2 next, return

next is used to skip an iteration of a loop

```
for(i in 1:100) {
  if (i <= 20) {
    ## Skip the first 20 iterations
    next
  }
  ## Do something here
  print(i)
}
```

```
## [1] 21
## [1] 22
## [1] 23
## [1] 24
## [1] 25
## [1] 26
## [1] 27
## [1] 28
## [1] 29
## [1] 30
## [1] 31
## [1] 32
## [1] 33
## [1] 34
## [1] 35
## [1] 36
## [1] 37
## [1] 38
## [1] 39
## [1] 40
## [1] 41
## [1] 42
## [1] 43
## [1] 44
## [1] 45
## [1] 46
## [1] 47
## [1] 48
## [1] 49
## [1] 50
## [1] 51
## [1] 52
## [1] 53
## [1] 54
## [1] 55
## [1] 56
## [1] 57
## [1] 58
## [1] 59
## [1] 60
## [1] 61
## [1] 62
```

```
## [1] 63
## [1] 64
## [1] 65
## [1] 66
## [1] 67
## [1] 68
## [1] 69
## [1] 70
## [1] 71
## [1] 72
## [1] 73
## [1] 74
## [1] 75
## [1] 76
## [1] 77
## [1] 78
## [1] 79
## [1] 80
## [1] 81
## [1] 82
## [1] 83
## [1] 84
## [1] 85
## [1] 86
## [1] 87
## [1] 88
## [1] 89
## [1] 90
## [1] 91
## [1] 92
## [1] 93
## [1] 94
## [1] 95
## [1] 96
## [1] 97
## [1] 98
## [1] 99
## [1] 100
```

return signals that a function should exit and return a given value

1.5.3 Control Structures

Summary · Control structures like if, while, and for allow you to control the flow of an R program · Infinite loops should generally be avoided, even if they are theoretically correct. · Control structures mentiond here are primarily useful for writing programs; for command-line interactive work, the *apply functions are more useful.