Session 6
Control Flow 2

R for Social Data Science

Jeffrey Ziegler, PhD

Assistant Professor in Political Science & Data Science
Trinity College Dublin

Fall 2022

Road map for today

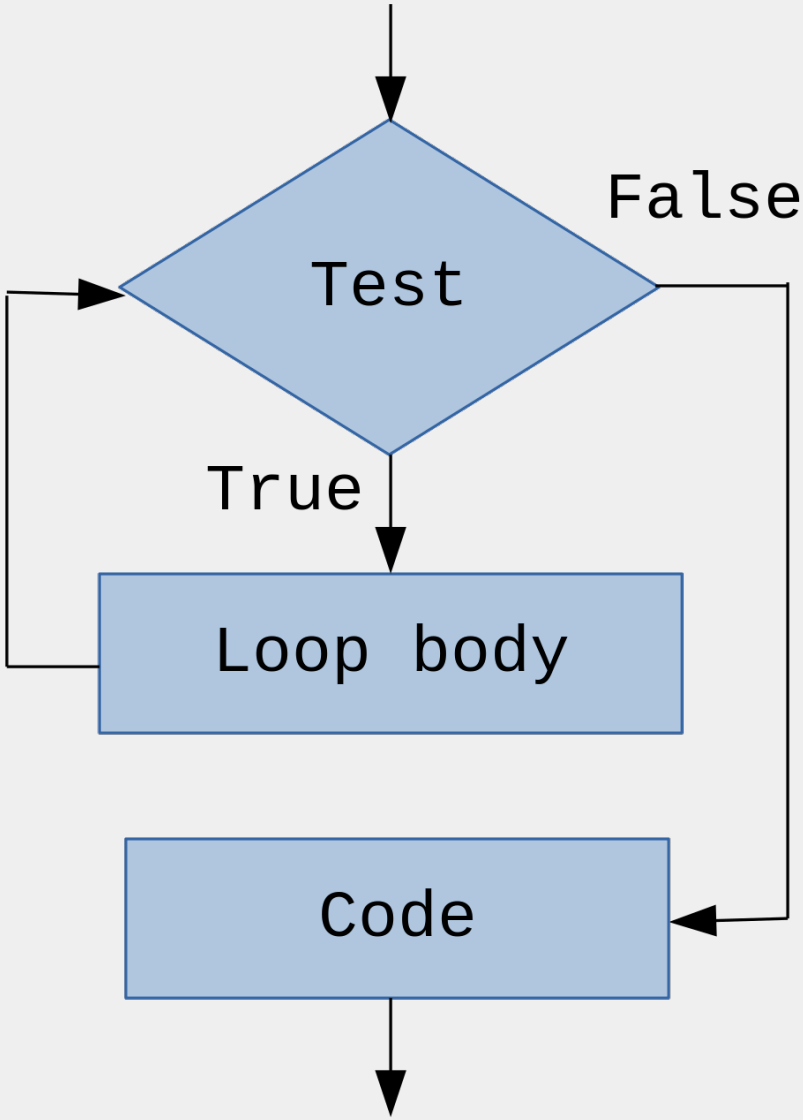
Last time:

- Straight-line and branching programs
- Algorithms
- Conditional statements

This time:

- Loops and Iteration

Iteration (looping)



Iteration: 'while'

- 'while' - defines a condition under which some code (loop body) is executed repeatedly

```
while (<boolean_expression>) {
<some_code>
}
```

```
# Calculate a factorial with decrementing function
# E.g. 5! = 1 * 2 * 3 * 4 * 5 = 120
x <- 5
factorial <- 1
while (x > 0) {
  factorial <- factorial * x
  x <- x - 1
}
factorial
```

```
[1] 120
```

Iteration: ‘for‘

- ‘for‘ - defines elements and sequence over which some code is executed iteratively

```
for (<element> in <sequence>) {
<some_code>
}
```

```
1 test <- c("t", "e", "s", "t")
2 for (i in test) {
3   # cat() function concatenates and prints objects'
      representations
4   cat(pasteo(i, "!"), "")
5 }
```

```
t! e! s! t!
```

Iteration with conditional statements

```
v <- c(3, 27, 9, 42, 10, 2, 5)
max_val <- NA
for (i in v) {
  if (is.na(max_val) | i > max_val) {
    max_val <- i
  }
}
```

```
[1] 42
```

Generating sequences for iteration

- ‘seq()‘ function that we encountered in subsetting can be used in looping
- As well as its cousins: ‘seq_len()‘ and ‘seq_along()‘

```
seq(<from>, <to>, <by>)
seq_len(<length>)
seq_along(<object>)
```

Generating sequences for iteration examples

```
s <- seq(1, 20)
s
```

```
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
seq_len(length(s))
```

```
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
seq_along(s)
```

```
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
seq_along(letters[1:20])
```

```
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

Generating sequences for iteration examples

```r
s2 <- vector(mode = "double", length = length(s))
for (i in seq_len(length(s))) {
  s2[i] <- s[i] * 2
}
s2
```

```
 [1]  2  4  6  8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40
```

```r
s3 <- vector(mode = "double", length = length(s))
for (i in seq_along(s)) {
  s3[i] <- s[i] * 3
}
s3
```

```
 [1]  3  6  9 12 15 18 21 24 27 30 33 36 39 42 45 48 51 54 57 60
```

Iteration: ‘next‘

- ‘next‘ - exits the iteration of a loop in which it is contained

```
for (i in seq(1,6)) {
  if (i %% 2 == 0) {
    break
  }
}
print(i)
```

```
[1] 1
```

Iteration: 'break'

- 'break' - terminates the loop in which it is contained

```
for (i in seq(1,6)) {
  if (i %% 2 == 0) {
    break
  }
}
print(i)
```

```
[1] 1
[1] 3
[1] 5
```

Infinite loops

- Loops that have no explicit limits for the number of iterations are called *infinite*
- They have to be terminated with a ‘break‘ statement (or Ctrl/Cmd-C in interactive session)
- Such loops can be unintentional (bug) or desired (e.g. waiting for user’s input, some event)

```
i <- 1
while (TRUE) {
  i <- i + 1
  if (i > 10) {
    break
  }
}
i
```

```
[1] 11
```

Iteration: ‘repeat‘

- ‘repeat‘ - defines code which is executed iteratively until loop is explicitly terminated
- Is equivalent to ‘while (TRUE)‘

```
repeat {
<some_code>
}
```

```
1  i <- 1
2  repeat {
3    i <- i + 1
4    if (i > 10) {
5      break
6    }
7  }
```

```
[1] 11
```

Infinite loop

```
while (TRUE)
```

Tutorial: Iterations and Loops

- Below you see a matrix of random 30 observations of 5 variables
- Inspect visually the matrix
- Which variable(s) do you think has(ve) the highest standard deviation?
- First, try subsetting individual rows and columns from this matrix
    1. Remove second row
    2. Remove third column

```
mat <- mapply(
  function(x) cbind(rnorm(n=30,mean=0,sd=x)),
  runif(n=5,min=0,max=10)
)
mat
```

Tutorial: Iterations and Loops

- Check dimensions of matrix using dim(), nrow() and ncol() functions
- Write a loop that goes over each variable and calculates its standard deviation
  - You can use sd() function to calculate standard deviation
  - Save these calculated standard deviations in a vector
- Find variable with maximum standard deviation using max() or which.max() functions
  - Is it the one you thought it would be?

Overview

This week:

- Straight-line and branching programs
- Algorithms
- Conditional statements
- Loops and Iteration

Next week:

- Functions in R