Week 3 Tutorial: Control Flow in R

POP77001 Computer Programming for Social Scientists

Module website: tinyurl.com/POP77001

Note on code formatting

- Use consistent style and indentation (RStudio indents by 2 whitespaces, Jupyter Notebook by 4)
- Even though it does not affect how programs are executed in R

Note on code formatting

- Use consistent style and indentation (RStudio indents by 2 whitespaces, Jupyter Notebook by 4)
- Even though it does not affect how programs are executed in R

In [2]:

```r
# Good style
is_positive <- function(num) {
  if (num > 0) {
    res <- TRUE
  } else {
    res <- FALSE
  }
  return(res)
}
```

Note on code formatting

- Use consistent style and indentation (RStudio indents by 2 whitespaces, Jupyter Notebook by 4)
- Even though it does not affect how programs are executed in R

In [2]:
```r
# Good style
is_positive <- function(num) {
  if (num > 0) {
    res <- TRUE
  } else {
    res <- FALSE
  }
  return(res)
}
```

In [3]:
```r
# Bad style
is_positive <- function(num) {
if (num > 0) {
res <- TRUE
}
else {
res <- FALSE
}
return(res)
}
```

Exercise 1: Conditional Statements

- Below you will find a code snippet for finding the maximum value in vector `v` using exhaustive enumeration.
- Modify it in such a way that it finds the minimum (rather than maximum) value.
- Check that your code works correctly by applying the built-in function `min()`.

In [4]:

```
set.seed(2022)
v <- sample(1:1000, 50)
max_val <- v[1]
for (i in v) {
  if (i > max_val) {
    max_val <- i
  }
}
max_val
```

```
[1] 998
```

- Now let's make this code more robust.
- Re-write the code above so that it can handle vectors that contain NAs in them.
- Test your code on the vector below.

- Now let's make this code more robust.
- Re-write the code above so that it can handle vectors that contain NAs in them.
- Test your code on the vector below.

In [8]:
```
set.seed(2022)
v <- sample(c(1:500, rep(NA, 500)), 25)
```

Exercise 2: Iteration

- 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
- Check the dimensionality of the 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 the standard deviation
- Save these calculated standard deviations in a vector
- Find the variable with the maximum standard deviation using `max()` or `which.max()` functions
- Is it the one you thought it would be?

In [4]:

```r
# When dealing with random number generation it's always a good idea to
# by setting the seed with set.seed(function)
set.seed(2021)
# Here we create a matrix of 30 observations of 5 variables
# where each variable is a random draw from a normal distribution with
# and standard deviation drawn from a uniform distribution between 0 ar
mat <- mapply(
  function(x) cbind(rnorm(n = 30, mean = 0, sd = x)),
  runif(n = 5, min = 0, max = 10)
)
```

In [4]:
```r
# When dealing with random number generation it's always a good idea to
# by setting the seed with set.seed(function)
set.seed(2021)
# Here we create a matrix of 30 observations of 5 variables
# where each variable is a random draw from a normal distribution with
# and standard deviation drawn from a uniform distribution between 0 ar
mat <- mapply(
  function(x) cbind(rnorm(n = 30, mean = 0, sd = x)),
  runif(n = 5, min = 0, max = 10)
)
```

In [5]:
```r
mat
```

```
      [,1]       [,2]        [,3]         [,4]       [,5]
 [1,]  2.3839361  -9.3570121  -3.38211062 -1.6188191   2.434582
8
 [2,] -2.8108812   3.6398922   9.12221934  2.1498588   2.114532
7
 [3,]  9.5657865 -13.0553064  -5.65808900 -1.5074431  -0.487091
6
 [4,]  4.4413551   3.4095297  -5.55809112 -4.7089982 -11.591002
1
 [5,]  0.7666776  12.0176689   0.86723020  0.8672294 -11.614873
0
 [6,] -3.0214758  -0.8865633 -11.54534021  6.1809994  -1.319497
3
 [7,]  5.0308587  -4.3594766  10.55399527  0.5751735   1.679959
0
 [8,]  0.5180495  20.3242580  -1.64929984 -0.2931002  -1.184991
1
```

```
 [9,]  7.6669813  -1.8116540  16.75477035 -4.4174753  -6.163295
5
[10,] -2.7861283   3.2815657  -4.84693805  5.0580929   3.186592
1
[11,]  6.1104984  -6.1228633  -1.06513883 -2.3779707  -3.927849
6
[12,]  5.3133132   7.2338976   8.62467646 -0.9139698  -1.522958
5
[13,]  4.0453120  -2.0328074  -0.09658005 -4.4241866  -2.791951
5
[14,]  7.1206917   3.6832976 -12.75733596 -2.2678010  -7.499024
0
[15,]  5.8374139   0.4654195   1.84243188 -2.2238920   5.395620
2
[16,] -3.4106540   9.9545362   3.51130060 -3.1272271   6.959347
5
[17,] -3.1863625   4.7673280  -5.40019237 -5.7072101 -12.604807
8
[18,] -0.5974791   4.0041569   3.99436039  8.4112405   0.173272
1
[19,]  4.6128663  -4.4334811  -4.27529981  3.5846152   4.572060
5
[20,]  7.7628644   9.3434631   1.21902325  2.7199821  -3.411513
8
[21,]  3.6998460  -5.6681773   6.81095739 -3.0746508   2.689981
9
[22,] -9.6012377  -1.3070740  -3.12625864  3.0189988  -2.058110
7
[23,]  4.1883348   2.6332821  -0.73465194  0.4420367  -1.754610
3
[24,]  1.8300096  -6.5693717   5.11940776 -5.0718646   0.631413
```

Week 3: Assignment 1

- Practice subsetting, conditional statements and iterations in R
- Due by 23:59 on Friday, 30th September