Practical 1

Jumping Rivers

Practical 1

The aim of this practical is to understand the syntax of functions and loops. In practical 2, we will use this knowledge in a larger example.

Basic functions

Consider the following simple function

```
v = 5
Fun1 = function() {
    v = 0
    return(v)
}
Fun1()
```

1. Why does the final line return 0 and not 5.
2. Delete line 3 in the above piece of code. Now change `Fun1()` to allow `v` to be passed as an argument, i.e. we can write `Fun1(5)`. Call this function to make sure it works.

Default arguments:

```
Fun2 = function(x = 10) {
    return(x)
}

Fun3 = function(x) {
    return(x)
}
```

1. Why does

   ```
   Fun2()
   ```

   work, but this raises an error

   ```
   Fun3()
   ```

2. Change `Fun2` so that it returns `x*x`.

`if` statements.

```
Fun4 = function(x) {
    if (x == 5) {
        y = 0
    } else {
        y = 1
    }
    return(y)
}
```

Change `Fun4` so that it:

- returns 1 if x is positive;
- returns -1 if x is negative;
- returns 0 if x is zero.

Change `Fun4()` so it errors if x is positive

`for` loops.

```
total = 0
for (i in 1:5) {
    total = total + i
}
total
```

The `for` loop above calculates

$$\sum_{i=1}^{5} i = 1 + 2 + 3 + 4 + 5$$

1. What is the final value of `total` in the above piece of code?

2. Change the above loop to calculate the following summations:

$$\text{(i)} \sum_{i=1}^{20} (i + 1)$$

$$\text{(ii)} \sum_{j=-10}^{15} j$$

1. Rewrite the two loops using the `sum()` function. For example, the for loop in the first example can be written as `sum(1:5)`

More functions, for loops and signalling conditions:

```
a = 2
total = 0
for (blob in a:5) {
    total = total + blob
}
```

1. In the code above, delete line 1. Now put the above code in a function called `Fun5`, where `a` is passed as an argument, i.e. we can call `Fun5(1)`
2. Alter the code so that the `for` loop goes from `a` to `b`, rather than `a` to 5. Allow `b` to be passed as an argument, i.e. we can call `Fun5(1,5)`.
3. Change `Fun5` so that it has default arguments of `a = 1` and `b = 10`.
4. Change `Fun5` so that it messages the user the total after each iteration and stops the function if the total has surpassed 50.

   ```
   ## Current total is 5

   ## Current total is 11

   ## Current total is 18

   ## Current total is 26

   ## Current total is 35

   ## Current total is 45
   ```

Multiple t-tests

In the below code, I've attempted to loop through a data frame and extract the maximum values.

```
dd = data.frame(w = rnorm(10), x = letters[1:10], y = rnorm(10),
    z = rnorm(10))

max_cols = rep(NA, ncol(dd))
for (i in seq_along(dd)) {
    max_cols[i] = max(dd[, i])
}
```

```
## Error in Summary.factor(structure(1:10, .Label = c("a", "b", "c", "d", : 'max' not meaningful for fa
```

```
max_cols
```

However, there's something wrong. The second column isn't numeric and so the for loop breaks when we get to there. Of course, we could just change the iterations to c(1,3,4) to leave out the second column. But imagine we have tens of columns. Use the `try()` function to bypass the error.

Solutions

The solutions can be viewed via

```
library("jrAdvPackage")
vignette("solutions1", package = "jrAdvPackage")
```