Practical 1

Jumping Rivers

This practical is aimed to served as a revision of functions, whilst introducing some new concepts which will be useful for the rest of the day!

Argument matching

R allows a variety of ways to match function arguments. For example, by position, by complete name, or by partial name. We haven't covered argument matching in the lecture, so let's try and figure out the rules from the examples below. First we'll create a little function to help

```
arg_explore = function(arg1, rg2, rg3){
    paste("a1, a2, a3 = ", arg1, rg2, rg3)
}
```

Next we'll create a few examples. Try and predict what's going to happen before calling the functions. One of these examples will raise an error - why?

```
arg_explore(1, 2, 3)
arg_explore(2, 3, arg1 = 1)
arg_explore(2, 3, a = 1)
arg_explore(1, 3, rg = 1)
```

Can you write down a set of rules that R uses when matching arguments?

```
## SOLUTION
## See http://goo.gl/NKsved for the offical document
## To summeriase, matching happens in a three stage pass:
#1. Exact matching on tags
#2. Partial matching on tags.
#3. Positional matching
```

Following on from the above example, can you predict what will happen with

```
sample(size = 10, TRUE, x = c(1, 2, 3, 4))
```

and

```
sample(10, TRUE, x = c(1, 2, 3, 4))
```

```
## SOLUTION
#sample(size = 10, TRUE, x = c(1,2,3,4)) is equivilent to
sample(x = c(1,2,3,4), size = 10, replace = TRUE)
#sample(10, TRUE, x = c(1,2,3,4)) is also equivilent to
sample(x = c(1,2,3,4), size = 10, replace = TRUE)
```

Variable scope

Scoping can get tricky. **Before** running the example code below, predict what is going to happen

1. A simple one to get started

```
f = function(x) return(x + 1)
f(10)
```

```
##Nothing strange here. We just get
f(10)
```

2. A bit more tricky

```
f = function(x) {
    f = function(x) {
        x + 1
    }
    x = x + 1
    return(f(x))
}
f(10)
```

3. More complex

```
f = function(x) {
    f = function(x) {
        f = function(x) {
            x + 1
        }
        x = x + 1
        return(f(x))
    }
    x = x + 1
    return(f(x))
}
f(10)
```

```
## Solution: The easiest way to understand is
## to use print statements
```

```
f = function(x) {
    f = function(x) {
        f = function(x) {
            message("f1: = ", x)
            x + 1
        }
        message("f2: = ", x)
        x = x + 1
        return(f(x))
    }
    message("f3: = ", x)
    x = x + 1
    return(f(x))
}
f(10)
```

```
f = function(x) {
    f = function(x) {
        x = 100
        f = function(x) {
            x + 1
        }
        x = x + 1
        return(f(x))
    }
    x = x + 1
    return(f(x))
}
f(10)
```

```
## Solution: The easiest way to understand is
## to use print statements as above
```

Solutions

Solutions are contained within the course package

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