Advanced R programming: solutions 4

Dr Colin Gillespie

Reference classes

The example in the notes created a random number generator using a reference class.

- Reproduce the `randu` generator from the notes and make sure that it works as advertised.[1]
- When we initialise the random number generator, the very first state is called the `seed`. Store this variable and create a new function called `get_seed` that will return the initial seed, i.e.

[1] The reference class version, not the function closure generator.

Reference classes also have an initialise method - that way we would only specify the seed and would then initialise the other variables. I'll give you an example in the solutions.

```
r = randu(calls=0, seed=10, state=10)
r$r()

## [1] 0.0003051898

r$get_state()

## [1] 655390

r$get_seed()

## [1] 10
```

```
##Solutions - see below
```

- Create a variable that stores the number of times the generator has been called. You should be able to access this variable with the function `get_num_calls`

```
r = randu(calls=0, seed=10, state=10)
r$get_num_calls()

## [1] 0

r$r()

## [1] 0.0003051898

r$r()

## [1] 0.001831097

r$get_num_calls()

## [1] 2
```

```r
## Solutions ##
randu = setRefClass("randu",
                    fields = list(calls = "numeric",
                                  seed="numeric",
                                  state="numeric"))
randu$methods(get_state = function() state)
randu$methods(set_state = function(initial) state <<- initial)
randu$methods(get_seed = function() seed)
randu$methods(get_num_calls = function() calls)
randu$methods(r = function() {
  calls <<- calls + 1
  state <<- (65539*state) %% 2^31
  return(state/2^31)
})
```

Solutions

Solutions are contained within the course package

```r
library("nclRadvanced")
vignette("solutions4", package="nclRadvanced")
```