

# Advanced R programming: practical 2

Dr Colin Gillespie

December 30, 2013

## 1 S3 objects

1. Following the cohort example in the notes, suppose we want to create a mean method.

- List all S3 methods associated with the mean function.
- Examine the source code of `mean.default`.
- What are the arguments of `mean.default`?
- Create a function called `mean.cohort` that returns a vector containing the mean weight and mean height.<sup>1</sup>

<sup>1</sup> Ensure that you can pass in the standard mean arguments, i.e. `na.rm`.

2. Let's now make a similar function for the standard deviation

- Look at the arguments of the `sd` function.
- Create an function call `sd.cohort` that returns a vector containing the weight and height standard deviation.<sup>2</sup>
- Create a default `sd` function. Look at `cor.default` in the notes for a hint.

<sup>2</sup> Ensure that you can pass in the standard `sd` arguments, i.e. `na.rm`.

## 2 S4 objects

1. Following the Cohort example in the notes, suppose we want to make a generic for the mean function.

- Using the `isGeneric` function, determine if the mean function is an S4 generic. If not, use `setGeneric` to create an S4 generic.
- Using `setMethod`, create a mean method for the Cohort class.<sup>3</sup>

I've intentionally mirrored the functions from section 1 of this practical to highlight the differences.

<sup>3</sup> Be careful to match the arguments.

2. Repeat the above steps for the `sd` function.

## 3 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.<sup>4</sup>
- 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.

<sup>4</sup> The reference class version, not the function closure generator.

```
r = randu(calls = 0, seed = 10, state = 10)
r$r()
## [1] 0.0003052
```

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$get_state()

## [1] 655390

r$get_seed()

## [1] 10
```

- 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] 0.0003052

r$r()

## [1] 0.001831

r$get_num_calls()

## [1] 2
```

## *Solutions*

Solutions are contained within the course package

```
library("nclRadvanced")
vignette("solutions2", package = "nclRadvanced")
```