*Advanced R programming: practical 2*

*Dr Colin Gillespie*

*May 8, 2014*

## 1  Rprofile

1. Create an `.Rprofile` file. Add the line

```r
cat("Successfully loaded .Rprofile at", date(), "\n")

## Successfully loaded .Rprofile at Thu May  8 08:42:49 2014
```

   to the file and restart R. Does the welcome message appear?

   An easy way of creating the file is to use the R function `file.create`, so

```r
file.exists("~/.Rprofile")
file.create("~/.Rprofile")
```

2. Try adding my suggestions to your `.Rprofile`, e.g.

```r
options(prompt="R> ", digits=4,
        show.signif.stars=FALSE)
```

   and setting the CRAN mirror:

```r
r = getOption("repos")
r["CRAN"] = "http://cran.rstudio.com/"
options(repos = r)
rm(r)
```

3. I've put a few other functions functions that I use online

   http://goo.gl/nf82zX

   feel free to use them. Also take a look at this stackoverflow question

   http://goo.gl/TLFLQR

## 2  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`.
   - What are the arguments of `mean`?

- Create a function called `mean.cohort` that returns a vector containing the mean weight and mean height.[1]

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.[2]
  - Create a default `sd` function. Look at `cor.default` in the notes for a hint.

3. Create a `hist` method for the `cohort` class. When the `hist` function is called on a `cohort`, it should produce a single plot showing two histograms - one for height and another for weight.

4. Create a `[` method for the `cohort` class. This method should return a `cohort` object, but with the relevant rows sub setted. For example, if `cc` was a cohort object, then

```
cc[1:3, ]
```

would return the first three rows of the data frame.

5. Create a `[<-` method for the `cohort` class. This method should allow us to replace values in the `details` data frame, i.e.

```
cc[1, 1] = 10
```

## 3  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.[3]

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

3. Create a `hist` method for the `cohort` class. When the `hist` function is called on a `cohort`, it should produce a single plot showing two histograms - one for height and another for weight.

4. Create a `[` method for the `cohort` class. This method should return a `cohort` object, but with the relevant rows sub setted.

5. Create a `<-` method for the `cohort` class. This method should allow us to replace values in the `details` data frame.

[1] Ensure that you can pass in the standard mean arguments, i.e. `na.rm`.

[2] Ensure that you can pass in the standard `sd` arguments, i.e. `na.rm`.

I've intentionally mirrored the functions from section 2 of this practical to highlight the differences.

[3] Be careful to match the arguments.

## 4    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.[4]

- 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.

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] 0.0003052

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")
```