Practical 3

Jumping Rivers

S4 objects [1]

[1] I've intentionally mirrored the functions from previous practical to highlight the differences.

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

2. Using the `isGeneric` function, determine if the `mean` function is an S4 generic. If not, use `setGeneric` to create an S4 generic.

```
isGeneric("mean")

## [1] TRUE

setGeneric("mean")

## [1] "mean"
```

3. Using `setMethod`, create a `mean` method for the `Cohort` class.[2]

[2] Be careful to match the arguments.

```
setMethod("mean", signature = c("Cohort"), definition = function(x,
    ...) {
    m1 = mean(x@details[, 1], ...)
    m2 = mean(x@details[, 2], ...)
    return(c(m1, m2))
})
```

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

```
isGeneric("sd")

## [1] FALSE

setGeneric("sd")

## [1] "sd"

setMethod("sd", signature = c("Cohort"), definition = function(x,
    na.rm = FALSE) {
    m1 = sd(x@details[, 1], na.rm = na.rm)
    m2 = sd(x@details[, 2], na.rm = na.rm)
    return(c(m1, m2))
})
```

5. Create a `summary` method for the `cohort` class

6. Use `isGeneric` to determine if an S4 generic exists.

7. Use `setGeneric` to set the generic method (if necessary).

8. Create an S4 summary method.

```
isGeneric("summary")
```

```
## [1] TRUE
```

```
setGeneric("summary")
```

```
## [1] "summary"
```

```
setMethod("summary", signature = c("Cohort"),
    definition = function(object, ...) {
        summary(object@details)
    })
```

9. 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.

```
isGeneric("hist")
```

```
## [1] FALSE
```

```
setGeneric("hist")
```

```
## [1] "hist"
```

```
setMethod("hist", signature = c("Cohort"), definition = function(x,
    ...) {
    dd = x@details
    Weight = ggplot(dd, aes(x = weight)) + geom_histogram() +
        labs(title = "Weight")
    Height = ggplot(dd, aes(x = height)) + geom_histogram() +
        labs(title = "Height")
    gridExtra::grid.arrange(Weight, Height)
})
```

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

```
isGeneric("[")
```

```
## [1] TRUE
```

```
getGeneric("[")
```

```
## standardGeneric for "[" defined from package "base"
##
## function (x, i, j, ..., drop = TRUE)
## standardGeneric("[", .Primitive("["))
## <bytecode: 0x561f78800898>
## <environment: 0x561f787f51b0>
## Methods may be defined for arguments: x, i, j, drop
## Use  showMethods("[")  for currently available ones.
```

```
## Can you determine what drop does?
setMethod("[", signature = c("Cohort"), definition = function(x,
    i, j, ..., drop = TRUE) {
    x@details = x@details[i, j, ..., drop = drop]
    x
})
```

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

```
isGeneric("[<-")
```

```
## [1] TRUE
```

```
setGeneric("[<-")
```

```
## [1] "[<-"
```

```
setMethod("[<-", signature = c("Cohort"), definition = function(x,
    i, j, value) {
    x@details[i, j] = value
    x
})
coh_s4[1, ] = 5
```

Solutions

Solutions are contained within the course package

```
library("jrOOP")
vignette("solutions3", package = "jrOOP")
```