S3 Class

Andrew Robinson
ACERA & University of Melbourne

August 15, 2012

Outline

Challenge: Scaling the Language

Conundrum: how to design a computer language for easy growth?

We want to handle all kinds of different challenges.

This introduces the problem of naming things.

```
> length(apropos("^print."))
[1] 47
```

Object-Oriented Programming

Think about objects receiving and passing messages.

Vocabulary

- Encapsulation means that information is hidden.
- Inheritance means that code can be easily shared among objects.
- Polymorphism means that procedures can accept or return objects of more than one type.

S3: Super-informal OO

S3 OO is class-based and impure / hybrid.

Vocabulary

- A class is an attribute of an object that dictates what messages the object can receive and return.
- A method is a function that is designed for a specific class.
- Dispatch is selection of the class-specific method.

S3 OO hinges on classes and methods.

Discovering Infrastructure: Generic Method

```
> mean

function (x, ...)
UseMethod("mean")
<bytecode: 0x10326dcd8>
<environment: namespace:base>
```

Discovering Infrastructure: Generic Method

```
> apropos("^mean.")
[1] "mean.Date"          "mean.POSIXct"
[3] "mean.POSIXlt"       "mean.data.frame"
[5] "mean.default"       "mean.difftime"
```

Discovering Infrastructure: Generic Method

```
> mean.default
function (x, trim = 0, na.rm = FALSE, ...)
{
    if (!is.numeric(x) && !is.complex(x) && !is.logical(x)) {
        warning("argument is not numeric or logical: returning NA")
        return(NA_real_)
    }
    if (na.rm)
        x <- x[!is.na(x)]
    if (!is.numeric(trim) || length(trim) != 1L)
        stop("'trim' must be numeric of length one")
    n <- length(x)
    if (trim > 0 && n) {
        if (is.complex(x))
            stop("trimmed means are not defined for complex data")
        if (any(is.na(x)))
            return(NA_real_)
        if (trim >= 0.5)
            return(stats::median(x, na.rm = FALSE))
        lo <- floor(n * trim) + 1
        hi <- n + 1 - lo
        x <- sort.int(x, partial = unique(c(lo, hi)))[lo:hi]
    }
    .Internal(mean(x))
}
<bytecode: 0x10336fc38>
```

Discovering Infrastructure: Classes for a Method

```
> methods(mean)

[1] mean.Date        mean.POSIXct
[3] mean.POSIXlt     mean.data.frame
[5] mean.default     mean.difftime
```

Discovering Infrastructure: Methods for a Class

```
> methods(class = "nls")

 [1] anova.nls*        coef.nls*
 [3] confint.nls*      deviance.nls*
 [5] df.residual.nls*  fitted.nls*
 [7] formula.nls*      logLik.nls*
 [9] nobs.nls*         predict.nls*
[11] print.nls*        profile.nls*
[13] residuals.nls*    summary.nls*
[15] vcov.nls*         weights.nls*

   Non-visible functions are asterisked
```

Discovering Infrastructure: Methods for a Class

```
> nobs

function (object, ...)
UseMethod("nobs")
<bytecode: 0x1037f5b88>
<environment: namespace:stats>
```

Discovering Infrastructure: Methods for a Class

```
> nobs.nls
Error: object 'nobs.nls' not found
```

Discovering Infrastructure: getAnywhere

```
> getAnywhere(nobs.nls)

A single object matching 'nobs.nls' was found
It was found in the following places
  registered S3 method for nobs from namespace stats
  namespace:stats
with value

function (object, ...)
if (is.null(w <- object$weights))
   length(object$m$resid()) else sum(w! = 0)
<bytecode: 0x10a8d78a0>
<environment: namespace:stats>
```

Discovering Infrastructure: Naming the Namespace

```
> stats:::nobs.nls

function (object, ...)
if (is.null(w <- object$weights))
   length(object$m$resid()) else sum(w! = 0)
<bytecode: 0x10a8d78a0>
<environment: namespace:stats>
```

Deploying a New Class

```
> x <- 1:3
> class(x) <- "digits"
```

Now ...

```
> attributes(x)
```

```
$class
[1] "digits"
```

Look at that assignment twice.

Writing a New Generic Function

```
> reveal <- function(x, ...) UseMethod("reveal")
```

Writing a Method

```
> reveal.default <- function(x, ...) head(x)
> reveal.digits <-
+   function(x, ...) paste(x, collapse = ", ")

> reveal(x)

[1] "1, 2, 3"
```

Inheritance

VERY artificial example!

```
> class(x) <- c("digits","numeric")
> mean(x)
[1] 2
```

Methods can be Overridden

```
> library(equivalence)
> data(ufc)
> ufc$missing <- is.na(ufc$Height)
> missing.heights <- glm(missing ~ Dbh.in,
+                        family = binomial,
+                        data = ufc)
```

Methods can be Overridden

```
> confint(missing.heights)          # Profiling
                  2.5 %     97.5 %
(Intercept) -0.63723616 0.09205416
Dbh.in      -0.03883647 0.00840713
> confint.default(missing.heights) # Wald
                  2.5 %      97.5 %
(Intercept) -0.63686037 0.091558908
Dbh.in      -0.03852649 0.008626013
```

Object-Oriented Programming Redux

Think about objects receiving and passing messages.

Vocabulary

- Encapsulation: non-visible functions via namespaces.
- Inheritance: add to the vector of class labels.
- Polymorphism: method dispatch.

Debugging

`str`

- What is the class?
- What is the dimension?
- What are the components?
- What are the classes of the components?

Drawbacks

No control.

- Namespaces
- S4 classes

Using Method Dispatch for Flow Control

```
> increment.a <- function(x) x + 1
> increment.b <- function(x) x + 2
```

Using Method Dispatch for Flow Control

```
> use_if <- function(x, flag = "b") {
+   if (flag == "a") {
+     x <- increment.a(x)
+     } else {
+       x <- increment.b(x)
+     }
+   return(x)
+ }
> use_if(10)
```

```
[1] 12
```

Using Method Dispatch for Flow Control

```
> increment <- function(x, ...) UseMethod("increment")
```

Using Method Dispatch for Flow Control

```
> use_dispatch <- function(x, flag = "b") {
+   class(x) <- c(class(x), flag)
+   x <- increment(x)
+   return(x)
+ }
> use_dispatch(10)

[1] 12
attr(,"class")
[1] "numeric" "b"
```

Using Method Dispatch for Flow Control

```
> system.time(sapply(1:1000, use_if))

   user  system elapsed
  0.003   0.000   0.003

> system.time(sapply(1:1000, use_dispatch))

   user  system elapsed
  0.019   0.000   0.018
```

Gaming the System

```
> class(x) <- c("glm","family","link")
```

Two Views

Method-centric vs. Object-centric Programming

Example: Method Focus

```
> jll <- function(y, mu, m, a) UseMethod("jll")
> linkFn <- function(mu, m, a) UseMethod("linkFn")
> lPrime <- function(mu, m, a) UseMethod("lPrime")
> delink <- function(y, eta, m, a) UseMethod("delink")
> variance <- function(mu, m, a) UseMethod("variance")
```

Example: Method Focus

Family-specific functions

```
> variance.binomial <- function(mu, m, a)
+   mu * (1 - mu/m)
> jll.binomial <- function(y, mu, m, a)
+   dbinom(x = y, size = m, prob = mu / m, log = TRUE)
```

Link-specific functions

```
> linkFn.logit <- function(mu, m, a)
+   log(mu / (m - mu))
> lPrime.logit <- function(mu, m, a)
+   m / (mu * (m - mu))
> delink.logit <- function(y, eta, m, a)
+   m / (1 + exp(-eta))
```

Example: Method Focus

Probit Link instead of Logit Link.

```
> linkFn.probit <- function(mu, m, a)
+   qnorm(mu / m)
> lPrime.probit <- function(mu, m, a)
+   1 / (m * dnorm(qnorm(mu/m)))
> delink.probit <- function(y, eta, m, a)
+   m * pnorm(eta)
```

Example: Object Focus

http://www.metacritic.com

```
> str(games, vec.len = 2)

'data.frame': 36548 obs. of  7 variables:
 $ url       : Factor w/ 2173 levels "100000pyramid",..: 1 1 1 1 1 ...
 $ score     : num  94 80 64 62 60 ...
 $ source    : Factor w/ 206 levels "1UP","2404.org",..: 82 123 76 72 118 ...
 $ name      : Factor w/ 2173 levels "$100,000 Pyramid",..: 1 1 1 1 1 ...
 $ year      : num  2001 2001 ...
 $ pub.score: num  69 69 69 69 69 ...
 $ old.rank : int  1260 1260 1260 1260 1260 ...
```

Example: Object Focus

```
> class(games) <- c("meta", class(games))
```

Example: Object Focus

```
> print.meta <- function(x, n) {
+   cat(paste("The meta score from ", x$source[n],
+             "\nof ", x$name[n], "\nis ",
+             x$pub.score[n], ".\n", sep = ""))
+ }
> print(games, 15)

The meta score from Eurogamer
of 1503 A.D. - The New World
is 74.
```

Recap: Key Points

- S3 Classes — objects and (Generic) methods
- `foo.bar()`
- `methods`, `getAnywhere`
- `str`

. . . what was the question?

Introduction

Discussion Points

Other Angles

Examples