[S3 Introduction][https://www.youtube.com/watch?v=VZkD7DXQ-fk&list=PLI098pDmWQZqjayjwmeSJ2Zgt1M1Yi_RE&index=8&t=2961s]

Scaling the language/ Easy to extent the language

- Object-Oriented Programming

当对象接受并传递信息时所具有特征：

Encapsulation：信息是隐藏的，避免被code细节所干扰，方便使用

Inheritance：code(脚本，代码)能够很容易在不同对象中分享，generic 函数容易在不同函数中使用

Polymorphism：该过程能够接受并返回不同的对象类型，针对不同的class类型

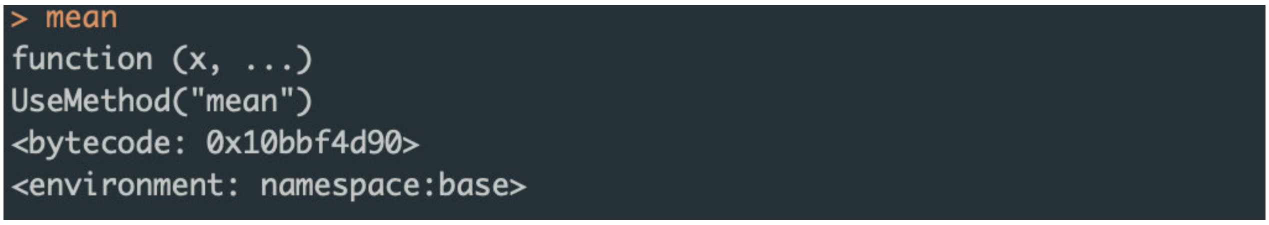
- S3 OO is class-base and impure/hybrid

class：对象属性，指定对象所能接受或返回的信息

method：用于指定class的函数

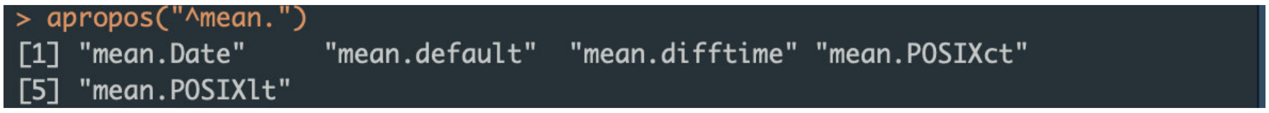
dispatch：为某一class特殊的函数选择，appropriate version of function

- Discovering Infrastructure: Generic Method



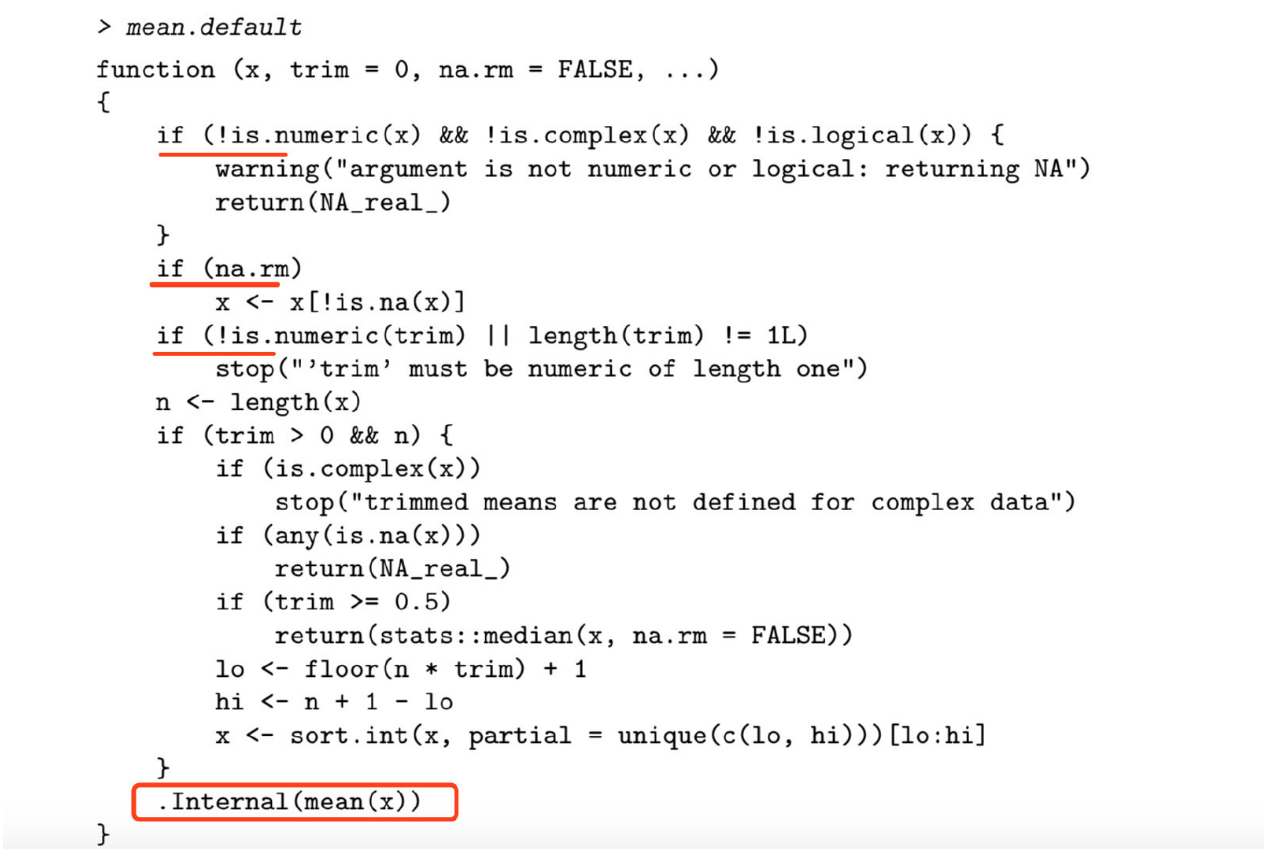
```
> mean
function (x, ...)
UseMethod("mean")
<bytecode: 0x10bbf4d90>
<environment: namespace:base>
```

当使用mean时，首先查看function(x,...)中的x的class，一般为第一位置值，然后对应使用mean.class_value，例如x的class为data.frame，那么使用对应的mean.data.frame函数



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

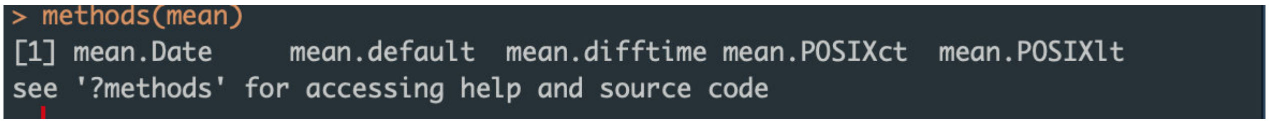
拥有不同的class类型处理策略，如果x的class不在其中，采用mean.default的函数



```
> 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))
}
```

- Discovering Infrastructure: Classes for a Method

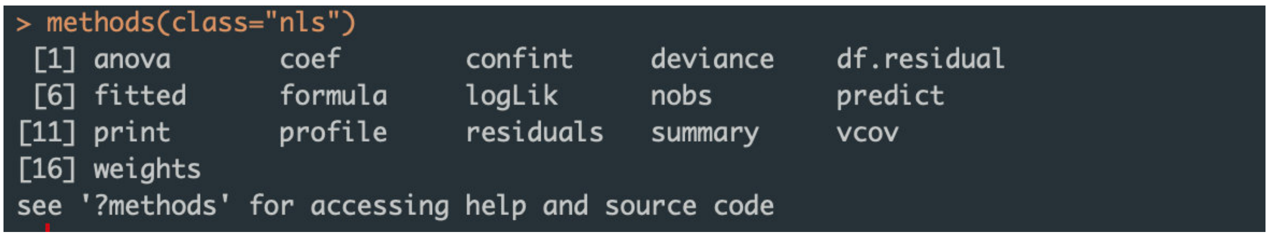
查看mean的函数的classes：



```
> methods(mean)
[1] mean.Date     mean.default  mean.difftime mean.POSIXct  mean.POSIXlt
see '?methods' for accessing help and source code
```

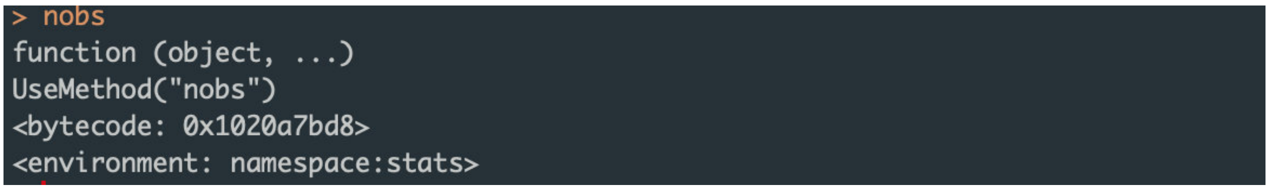
- Discovering Infrastructure: Methods for a Class

查看class拥有的methods：



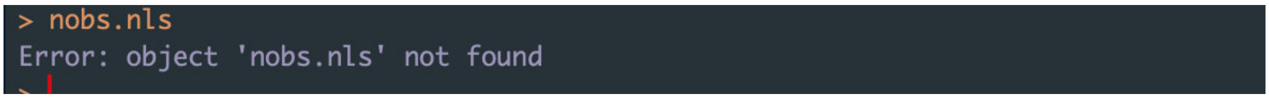
```
> methods(class="nls")
 [1] anova       coef        confint     deviance    df.residual
 [6] fitted      formula     logLik      nobs        predict
[11] print       profile     residuals   summary     vcov
[16] weights
see '?methods' for accessing help and source code
```

例如查看nobs的method



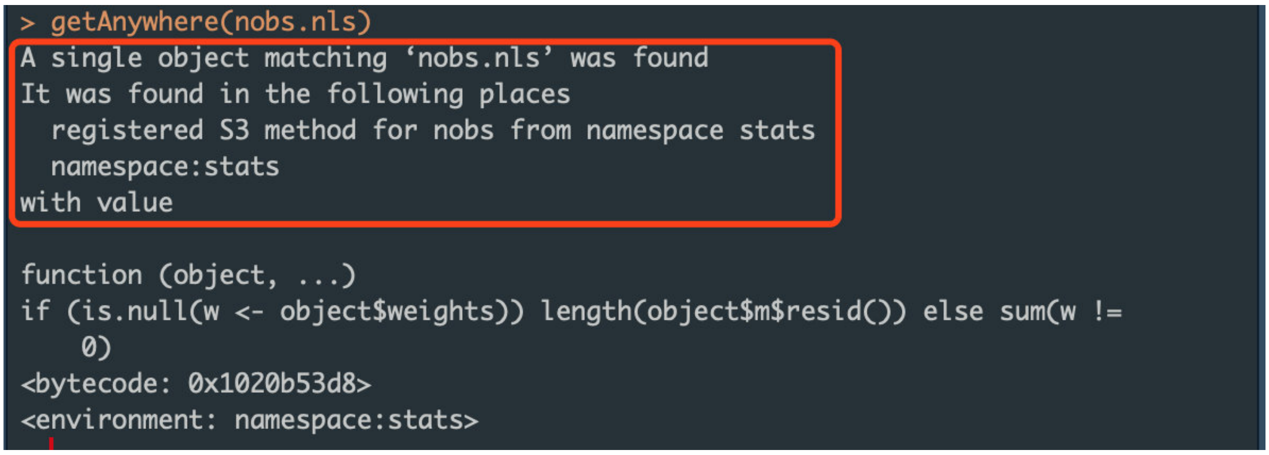
```
> nobs
function (object, ...)
UseMethod("nobs")
<bytecode: 0x1020a7bd8>
<environment: namespace:stats>
```

查看对应的class的function：



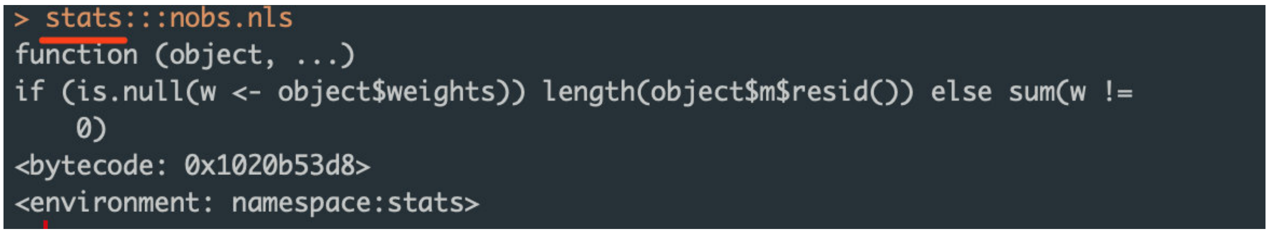
```
> nobs.nls
Error: object 'nobs.nls' not found
```

- Discovering Infrasture: getAnywhere



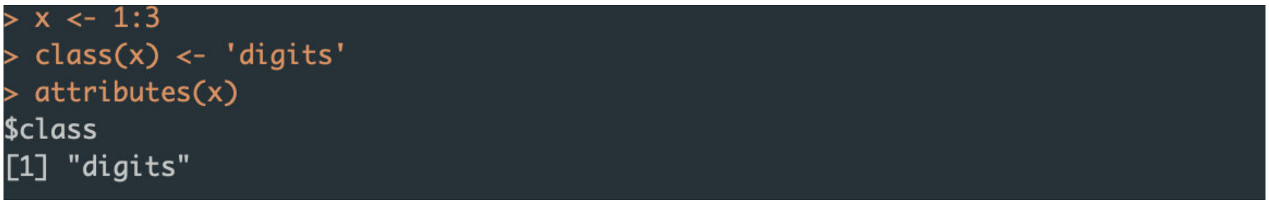
```
> getAnywhere(nobs.nls)
A single object matching ‘nobs.nls’ was found
It was found in the following places
  registered S3 method for nobs from namespace stats
  namespace:stats
with value

function (object, ...)
if (is.null(w <- object$weights)) length(object$m$resid()) else sum(w !=
    0)
<bytecode: 0x1020b53d8>
<environment: namespace:stats>
```

使用getAnywhere函数查看隐藏methods的信息或者根据namespace



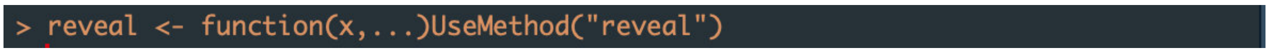
```
> stats:::nobs.nls
function (object, ...)
if (is.null(w <- object$weights)) length(object$m$resid()) else sum(w !=
    0)
<bytecode: 0x1020b53d8>
<environment: namespace:stats>
```

- Deploying a New Class



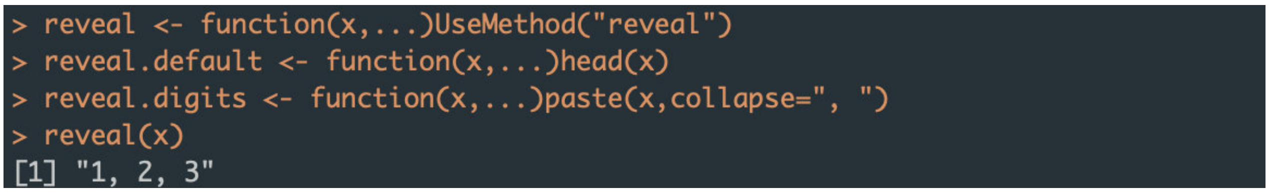
```
> x <- 1:3
> class(x) <- 'digits'
> attributes(x)
$class
[1] "digits"
```

- Writing a New Generic Function



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

- Writing a Method



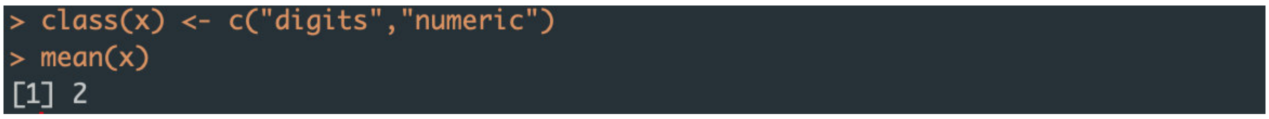
```
> reveal <- function(x,...)UseMethod("reveal")
> reveal.default <- function(x,...)head(x)
> reveal.digits <- function(x,...)paste(x,collapse=", ")
> reveal(x)
[1] "1, 2, 3"
```

x的class为digit，对应采用reveal.digits的function

- Inheritance

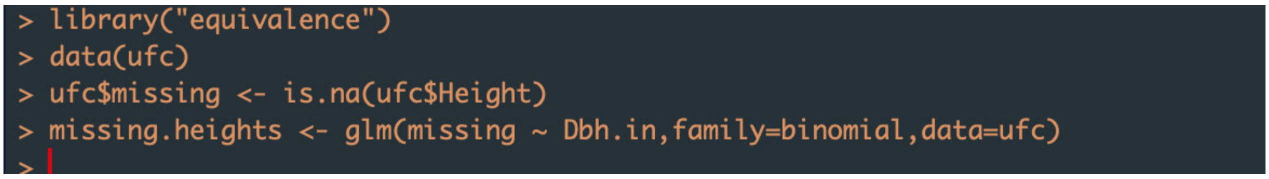
使我们的objects能够从generic classes去借code

very artifical example



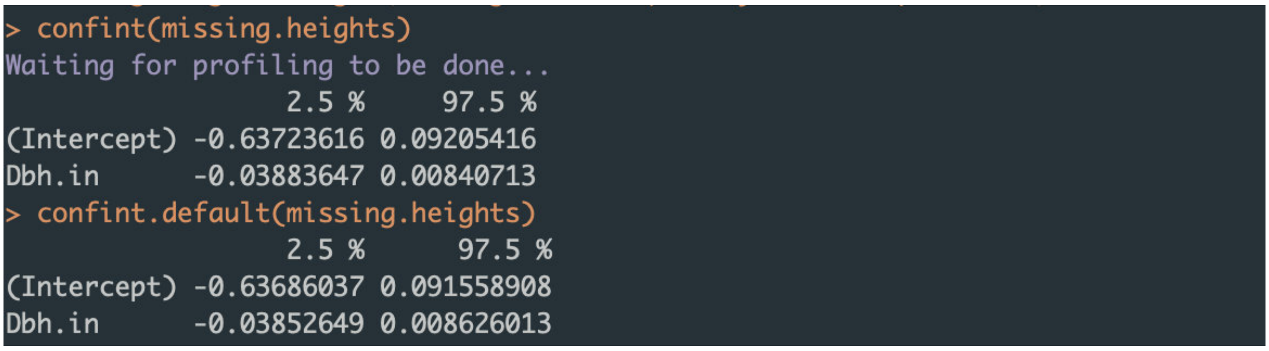
```
> class(x) <- c("digits","numeric")
> mean(x)
[1] 2
```

- Methods can be Overridden



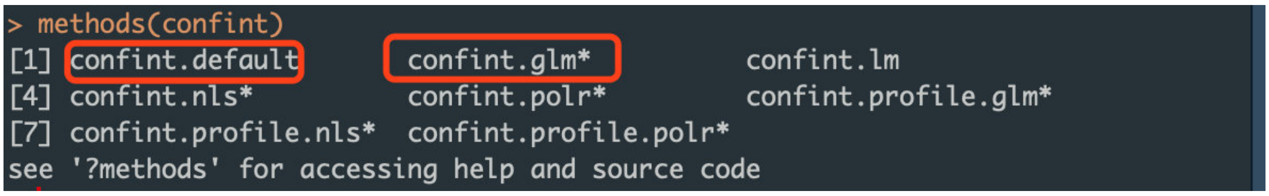
```
> library("equivalence")
> data(ufc)
> ufc$missing <- is.na(ufc$Height)
> missing.heights <- glm(missing ~ Dbh.in,family=binomial,data=ufc)
> 
```

打印不同指定不同method时的confint值



```
> confint(missing.heights)
Waiting for profiling to be done...
                  2.5 %     97.5 %
(Intercept) -0.63723616 0.09205416
Dbh.in      -0.03883647 0.00840713
> confint.default(missing.heights)
                  2.5 %      97.5 %
(Intercept) -0.63686037 0.091558908
Dbh.in      -0.03852649 0.008626013
```

原因：confint针对不同的class拥有对应的method，星号表隐藏



```
> methods(confint)
[1] confint.default      confint.glm*         confint.lm
[4] confint.nls*         confint.polr*        confint.profile.glm*
[7] confint.profile.nls* confint.profile.polr*
see '?methods' for accessing help and source code
```

- Object-Oriented Programming Redux

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

- Debugging

  `str`

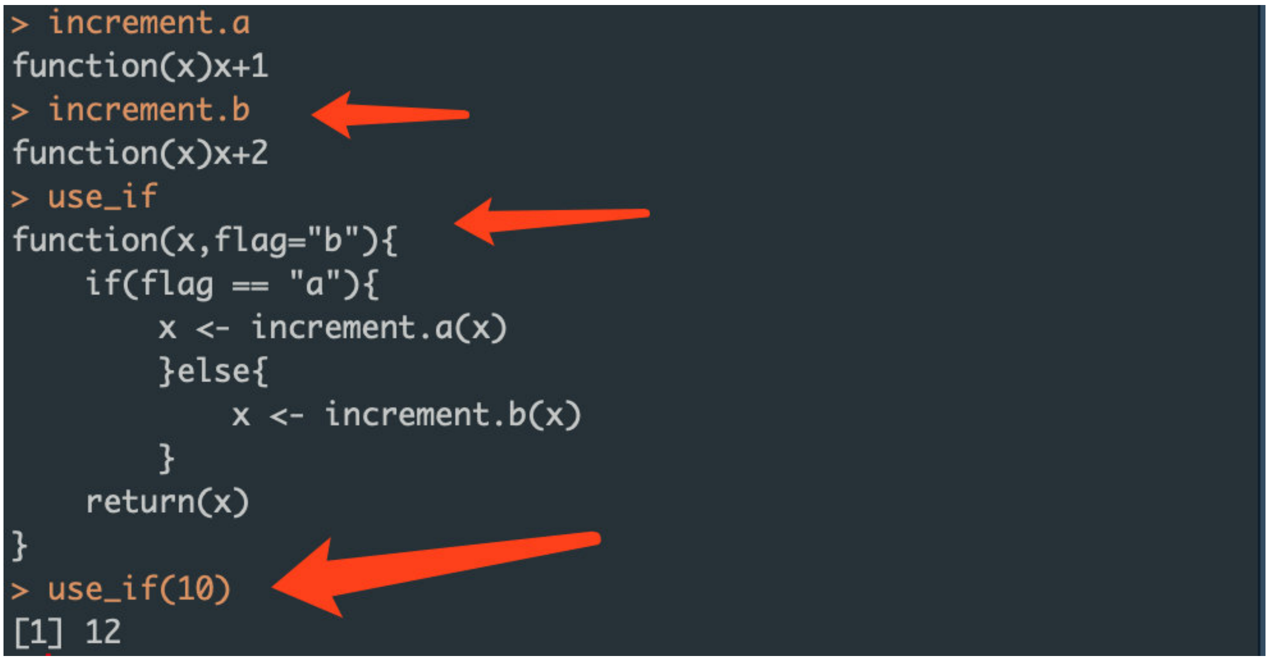
  - ▶ What is the class?
  - ▶ What is the dimension?
  - ▶ What are the components?
  - ▶ What are the classes of the components?

第一件事就是查看数据结构，查找错误原因，str()

- Drawbacks

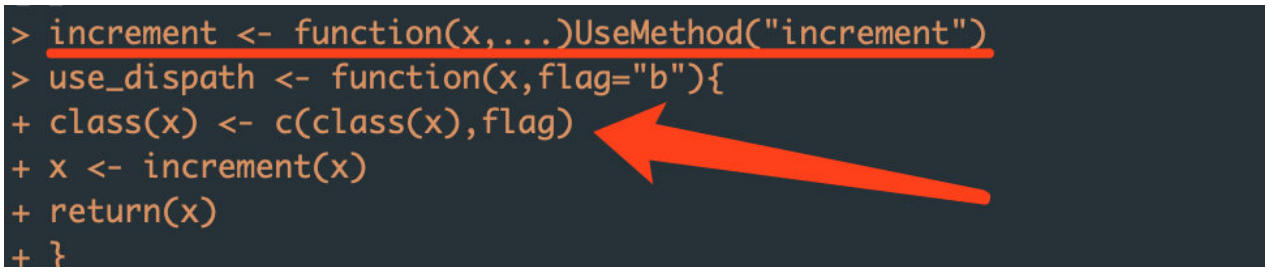
当function面对不兼容的objects类别时，将会强制转变类别格式来采用funciton

- Using Method Dispath for Flow Control

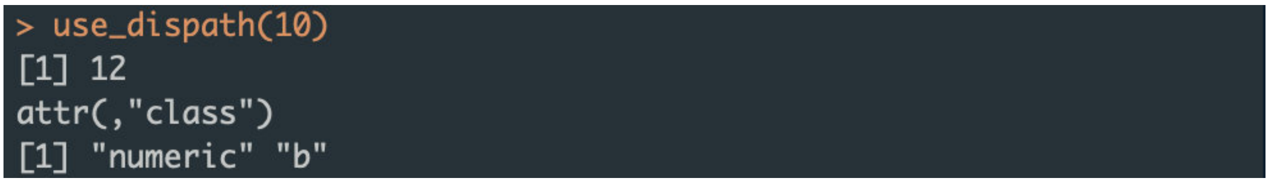

```
> increment.a
function(x)x+1
> increment.b
function(x)x+2
> use_if
function(x,flag="b"){
    if(flag == "a"){
        x <- increment.a(x)
        }else{
            x <- increment.b(x)
        }
    return(x)
}
> use_if(10)
[1] 12
```

Using method dispath:

首先构建generic function：

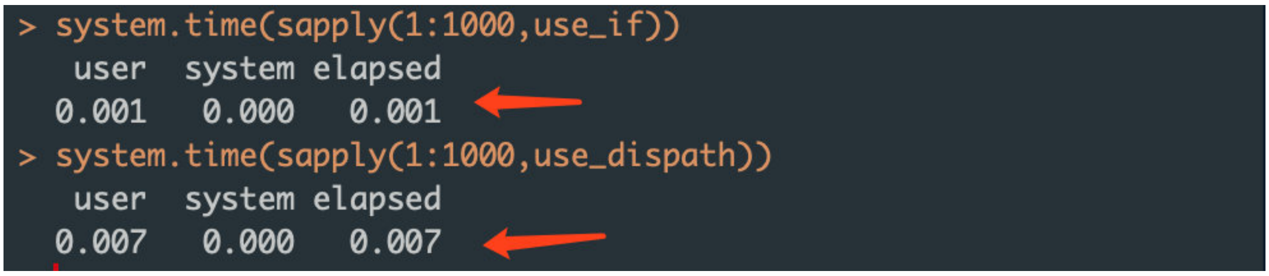

```
> increment <- function(x,...)UseMethod("increment")
> use_dispath <- function(x,flag="b"){
+ class(x) <- c(class(x),flag)
+ x <- increment(x)
+ return(x)
+ }
```

调用之前的increment.a/increment.b：


```
> use_dispath(10)
[1] 12
attr(,"class")
[1] "numeric" "b"
```

可以根据条件，构建多种class判断

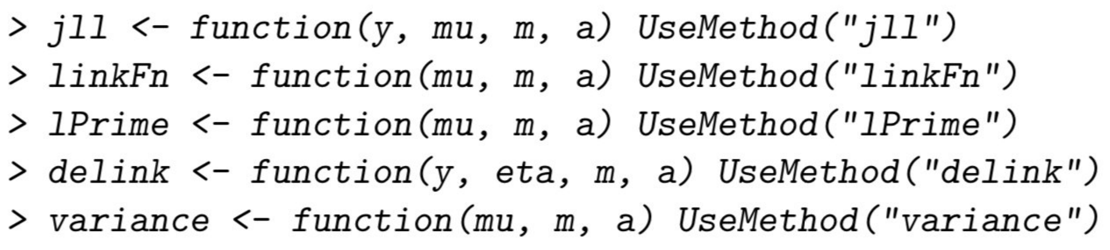
使用dispath方式速度慢


```
> system.time(sapply(1:1000,use_if))
   user  system elapsed
  0.001   0.000   0.001
> system.time(sapply(1:1000,use_dispath))
   user  system elapsed
  0.007   0.000   0.007
```

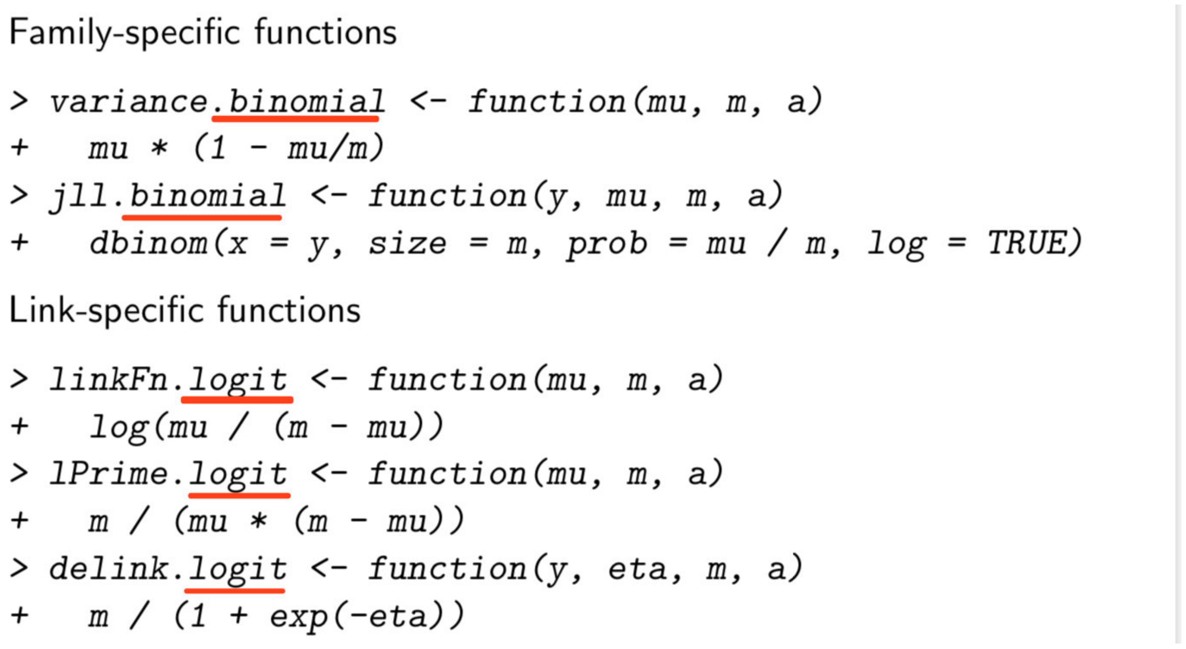
- Two Views

Method-centric vs. Object-centric Programming

- Example: Method Foucs



```
> 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")
```



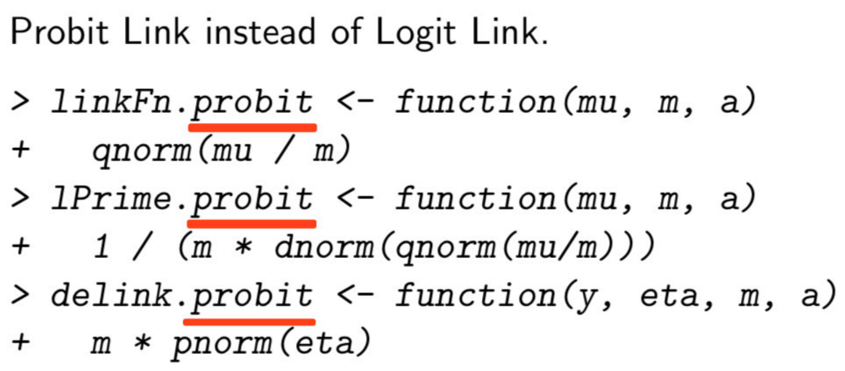
Family-specific functions

```
> 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))
```

根据不同的objects的class采用不同的functions



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)
```

对functions构建不同objects的class的dispath，分别处理

- 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 ...
```

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

```
> 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.
```

根据对象的不同，调整function的输出。