

# Clases y Métodos

Oscar Perpiñán Lamigueiro

Marzo de 2013

# Contenidos

Clases y Métodos

Oscar Perpiñán  
Lamigueiro

## OOP en R

### Programación Orientada a Objetos (OOP)

#### OOP en R

Programación Orientada a  
Objetos (OOP)

#### Clases y métodos S3

Clases

Métodos con S3

Métodos genéricos con S3

#### Clases y métodos S3

Clases

Métodos con S3

Métodos genéricos con S3

#### Clases y métodos S4

Clases en S4

Métodos en S4

Clases S3 con clases y métodos S4

#### Clases y métodos S4

Clases en S4

Métodos en S4

Clases S3 con clases y  
métodos S4

# Programación Orientada a Objetos (OOP)

Clases y Métodos

Oscar Perpiñán  
Lamigueiro

OOP en R

Programación Orientada a  
Objetos (OOP)

Clases y métodos  
S3

Clases

Métodos con S3

Métodos genéricos con S3

Clases y métodos  
S4

Clases en S4

Métodos en S4

Clases S3 con clases y  
métodos S4

- ▶ Características básicas del paradigma OOP:
  - ▶ Los objetos encapsulan información y control de su comportamiento (*objects*).
  - ▶ Las clases describen propiedades de un grupo de objetos (*class*).
  - ▶ Se pueden definir clases a partir de otras (*inheritance*).
  - ▶ Una función genérica se comporta de forma diferente atendiendo a la clase de uno (o varios) de sus argumentos (*polymorphism*).
- ▶ En R coexisten dos implementaciones de la OOP:
  - ▶ S3: elaboración informal con énfasis en las funciones genéricas y el polimorfismo.
  - ▶ S4: elaboración formal de clases y métodos.

## Referencias

- ▶ Software for Data Analysis
- ▶ How Methods Work
- ▶ S4 classes in 15 pages
- ▶ R Programming for Bioinformatics
- ▶ S4 System Development in Bioconductor

# Contenidos

Clases y Métodos

Oscar Perpiñán  
Lamigueiro

## OOO en R

### Programación Orientada a Objetos (OOO)

#### OOO en R

Programación Orientada a  
Objetos (OOO)

## Clases y métodos S3

### Clases

### Métodos con S3

### Métodos genéricos con S3

#### Clases y métodos S3

Clases

Métodos con S3

Métodos genéricos con S3

#### Clases y métodos S4

Clases en S4

Métodos en S4

Clases S3 con clases y  
métodos S4

## Clases y métodos S4

### Clases en S4

### Métodos en S4

### Clases S3 con clases y métodos S4

- ▶ Los objetos básicos en R tienen una clase implícita definida en S3. Es accesible con `class`.

```
x <- rnorm(10)
class(x)
```

```
[1] "numeric"
```

- ▶ Pero no tienen atributo ni se consideran formalmente objetos:

```
attr(x, 'class')
```

```
NULL
```

```
is.object(x)
```

```
[1] FALSE
```

## OOO en R

Programación Orientada a  
Objetos (OOO)

## Clases y métodos S3

Clases

Métodos con S3

Métodos genéricos con S3

## Clases y métodos S4

Clases en S4

Métodos en S4

Clases S3 con clases y  
métodos S4

# Clases

- Se puede redefinir la clase de un objeto S3 con `class`

```
class(x) <- 'myNumeric'  
class(x)
```

```
[1] "myNumeric"
```

- Ahora sí es un objeto y su atributo está definido:

```
attr(x, 'class')
```

```
[1] "myNumeric"
```

```
is.object(x)
```

```
[1] TRUE
```

- Sin embargo, su modo de almacenamiento (clase intrínseca) no cambia:

```
mode(x)
```

```
[1] "numeric"
```

# Definición de Clases

```
task1 <- list(what='Write an email',  
              when=as.Date('2013-01-01'),  
              priority='Low')  
class(task1) <- 'task3'  
task1
```

```
$what  
[1] "Write an email"
```

```
$when  
[1] "2013-01-01"
```

```
$priority  
[1] "Low"
```

```
attr(,"class")  
[1] "task3"
```

```
task2 <- list(what='Find and fix bugs',  
              when=as.Date('2013-03-15'),  
              priority='High')  
class(task2) <- 'task3'
```

## OOP en R

Programación Orientada a  
Objetos (OOP)

## Clases y métodos S3

### Clases

Métodos con S3

Métodos genéricos con S3

## Clases y métodos S4

Clases en S4

Métodos en S4

Clases S3 con clases y  
métodos S4



# Definición de Clases

```
myToDo <- list(task1, task2)
class(myToDo) <- c('ToDo3')
myToDo
```

```
[[1]]
$what
[1] "Write an email"

$when
[1] "2013-01-01"

$priority
[1] "Low"

attr(,"class")
[1] "task3"

[[2]]
$what
[1] "Find and fix bugs"

$when
[1] "2013-03-15"

$priority
[1] "High"

attr(,"class")
[1] "task3"

attr(,"class")
[1] "ToDo3"
```

# Métodos con S3: NextMethod

```
print.task3 <- function(x, ...){  
  cat('Task:\n')  
  NextMethod(x, ...)  
}
```

```
print(task1)
```

```
Task:  
$what  
[1] "Write an email"  
  
$when  
[1] "2013-01-01"  
  
$priority  
[1] "Low"  
  
attr(,"class")  
[1] "task3"
```

# Métodos con S3: NextMethod

```
print.ToDo3 <- function(x, ...){  
  cat('This is my ToDo list:\n')  
  NextMethod(x, ...)  
  cat('-----\n')  
}
```

```
print(myToDo)
```

```
This is my ToDo list:  
[[1]]  
Task:  
$what  
[1] "Write an email"  
  
$when  
[1] "2013-01-01"  
  
$priority  
[1] "Low"  
  
attr(,"class")  
[1] "task3"  
  
[[2]]  
Task:  
$what  
[1] "Find and fix bugs"  
  
$when  
[1] "2013-03-15"  
  
$priority
```

Clases y Métodos

Oscar Perpiñán  
Lamigueiro

OOP en R

Programación Orientada a  
Objetos (OOP)

Clases y métodos  
S3

Clases

Métodos con S3

Métodos genéricos con S3

Clases y métodos  
S4

Clases en S4

Métodos en S4

Clases S3 con clases y  
métodos S4

# Definición de un método S3 para ToDo3

```
print.ToDo3 <- function(x, ...){
  cat('This is my ToDo list:\n')
  for (i in seq_along(x)){
    cat('Task no.', i, ':\n')
    cat('What:', x[[i]]$what,
        ' - When:', as.character(x[[i]]$when),
        ' - Priority:', x[[i]]$priority,
        '\n')
  }
  cat('-----\n')
}
```

```
print(myToDo)
```

```
This is my ToDo list:
Task no. 1 :
What:  Write an email - When: 2013-01-01 - Priority: Low
Task no. 2 :
What:  Find and fix bugs - When: 2013-03-15 - Priority: High
-----
```

# Definición de un método S3 para task3

```
print.task3 <- function(x, number,...){  
  if (!missing(number)) cat('Task no.', number, ':\n')  
  cat('What: ', x$what,  
      '- When: ', as.character(x$when),  
      '- Priority: ', x$priority,  
      '\n')  
}
```

```
print(task1)
```

What: Write an email - When: 2013-01-01 - Priority: Low

```
print(myToDo[[2]])
```

What: Find and fix bugs - When: 2013-03-15 - Priority: High

# Redefinición del método para ToDo3

```
print.ToDo3 <- function(x, ...){  
  cat('This is my ToDo list:\n')  
  for (i in seq_along(x)) print(x[[i]], i)  
    cat('-----\n')  
}
```

```
print(myToDo)
```

```
This is my ToDo list:  
Task no. 1 :  
What:  Write an email - When: 2013-01-01 - Priority: Low  
Task no. 2 :  
What:  Find and fix bugs - When: 2013-03-15 - Priority: High  
-----
```

# Métodos genéricos con S3: UseMethod

Clases y Métodos

Oscar Perpiñán  
Lamigueiro

OOP en R

Programación Orientada a  
Objetos (OOP)

Clases y métodos  
S3

Clases

Métodos con S3

Métodos genéricos con S3

Clases y métodos  
S4

Clases en S4

Métodos en S4

Clases S3 con clases y  
métodos S4

```
myFun <- function(x, ...)UseMethod('myFun')
myFun.default <- function(x, ...){
  cat('Funcion genérica\n')
  print(x)
}
```

myFun(x)

```
Funcion genérica
[1] 0.9999021 0.7701458 -1.1414183 -0.7296333 0.6176794 -0.8534673
[7] 0.2304325 0.3848272 1.0438842 -0.5581884
attr("class")
[1] "myNumeric"
```

myFun(task1)

What: Write an email - When: 2013-01-01 - Priority: Low

- Con `methods` podemos averiguar los métodos que hay definidos para una función particular:

```
methods('myFun')
```

```
[1] myFun.default myFun.task3
```

```
head(methods('print'))
```

```
[1] "print.acf"      "print.anova"    "print.aov"      "print.aovlist"  
[5] "print.ar"       "print.Arima"
```



# Definición del método para task3 con UseMethod

```
myFun.task3 <- function(x, number,...){  
  if (!missing(number)) cat('Task_no.', number, ':\n')  
  cat('What: ', x$what,  
      '- When: ', as.character(x$when),  
      '- Priority: ', x$priority,  
      '\n')  
}
```

```
myFun(task1)
```

```
What: Write an email - When: 2013-01-01 - Priority: Low
```

```
methods(myFun)
```

```
[1] myFun.default myFun.task3
```

```
methods(class='task3')
```

```
[1] myFun.task3 print.task3
```

Clases y Métodos

Oscar Perpiñán  
Lamigueiro

OOP en R

Programación Orientada a  
Objetos (OOP)

Clases y métodos  
S3

Clases

Métodos con S3

Métodos genéricos con S3

Clases y métodos  
S4

Clases en S4

Métodos en S4

Clases S3 con clases y  
métodos S4

# Contenidos

Clases y Métodos

Oscar Perpiñán  
Lamigueiro

## OOO en R

### Programación Orientada a Objetos (OOO)

#### OOO en R

Programación Orientada a  
Objetos (OOO)

## Clases y métodos S3

### Clases

### Métodos con S3

### Métodos genéricos con S3

#### Clases y métodos S3

Clases

Métodos con S3

Métodos genéricos con S3

## Clases y métodos S4

### Clases en S4

### Métodos en S4

### Clases S3 con clases y métodos S4

#### Clases y métodos S4

Clases en S4

Métodos en S4

Clases S3 con clases y  
métodos S4

- ▶ Se construyen con `setClass`, que acepta varios argumentos
  - ▶ `Class`: nombre de la clase.
  - ▶ `representation`: una lista con las clases de cada componente. Los nombres de este vector corresponden a los nombres de los componentes (`slot`).
  - ▶ `contains`: un vector con las clases que esta nueva clase extiende.
  - ▶ `prototype`: un objeto proporcionando el contenido por defecto para los componentes definidos en `representation`.
  - ▶ `validity`: a función que comprueba la validez de la clase creada con la información suministrada.
- ▶ Una vez que la clase ha sido definida con `setClass`, se puede crear un objeto nuevo con `new`.

# Definición de una nueva clase

```
setClass('task',  
  representation=list(what='character',  
    when='Date',  
    priority='character')  
)
```

```
getClass('task')
```

```
Class "task" [in ".GlobalEnv"]  
  
Slots:  
  
Name:      what      when  priority  
Class: character    Date character
```

```
getSlots('task')
```

```
      what      when  priority  
"character" "Date" "character"
```

```
slotNames('task')
```

```
[1] "what" "when" "priority"
```

# Creación de un objeto con la clase definida:

## new

```
task1 <- new('task', what='Find_and_fix_bugs',  
             when=as.Date('2013-03-15'),  
             priority='High')
```

```
task1
```

```
An object of class "task"  
Slot "what":  
[1] "Find and fix bugs"  
  
Slot "when":  
[1] "2013-03-15"  
  
Slot "priority":  
[1] "High"
```



# Definición de la clase ToDo

```
setClass('ToDo',  
  representation=list(tasks='list')  
)
```

```
myList <- new('ToDo', tasks=list(task1, task2))
```

# Funciones para crear objetos

```
createToDo <- function(){  
  new('ToDo')  
}
```

```
addTask <- function(object, task){  
  stopifnot(is(task, 'task'))  
  object@tasks <- c(object@tasks, task)  
  object  
}
```



# Problema con los slots definidos como list

Clases y Métodos

Oscar Perpiñán  
Lamigueiro

OOP en R

Programación Orientada a  
Objetos (OOP)

Clases y métodos  
S3

Clases

Métodos con S3

Métodos genéricos con S3

Clases y métodos  
S4

Clases en S4

Métodos en S4

Clases S3 con clases y  
métodos S4

- Dado que el slot `tasks` es una `list`, podemos añadir cualquier cosa.

```
myListOps <- new('ToDo',  
                 tasks=list(t1='Tarea1',  
                           task1, task2))
```

- Para obligar a que sus elementos sean de clase task debemos añadir una función de validación.

```
valida <- function (object) {  
  if (any(sapply(object@tasks, function(x) !is(x, "task"))))  
    stop("not a list of task objects")  
  return(TRUE)  
}  
  
setClass('ToDo',  
  representation=list(tasks='list'),  
  validity=valida  
)
```

```
myListOps <- new('ToDo',  
  tasks=list(t1='Tarea1',  
    task1, task2))
```

Error en validityMethod(object) : not a list of task objects

# Métodos en S4

- ▶ Normalmente se definen con `setMethod`.
- ▶ Hay que definir:
  - ▶ la signature (clase de los argumentos para *esta* definición del método)
  - ▶ la función a ejecutar (definition).
- ▶ Es necesario que exista un método genérico ya definido. Si no existe, se define con `setGeneric`.

```
isGeneric('print')
```

```
[1] TRUE
```

- ▶ Si ya existe un método genérico, la función definition debe tener todos los argumentos de la función genérica y en el mismo orden.

```
getGeneric('print')
```

```
standardGeneric for "print" defined from package "base"
```

```
function (x, ...)  
  standardGeneric("print")  
<environment: 0xa937894>  
Methods may be defined for arguments: x  
Use showMethods("print") for currently available ones.
```

# Definición de un método print para task

```
setMethod('print', signature='task',  
  definition=function(x,...){  
    cat('What:␣', x@what,  
      '␣When:', as.character(x@when),  
      '␣Priority:', x@priority,  
      '\n')  
  })
```

```
[1] "print"
```

```
print(task1)
```

```
What: Find and fix bugs - When: 2013-03-15 - Priority: High
```

# Definición de un método print para task

```
setMethod('print', signature='ToDo',  
          definition=function(x,...){  
            cat('This is my ToDo list:\n')  
            tasksList <- x@tasks  
            for (i in seq_along(tasksList)) {  
              cat('No.', i, ':')  
              print(tasksList[[i]])  
            }  
            cat('-----\n')  
          })
```

```
[1] "print"
```

```
print(myList)
```

```
This is my ToDo list:  
No. 1 :What: Find and fix bugs - When: 2013-03-15 - Priority: High  
No. 2 :What: Write an email - When: 2013-01-01 - Priority: Low  
-----
```

# Clases S3 con clases y métodos S4

- ▶ Para usar objetos de clase S3 en signatures de métodos S4 o como contenido de slots de una clase S4 hay que registrarlos con `setOldClass`:

```
setOldClass('lm')
```

- ▶ Definimos un método genérico para `xyplot`

```
setGeneric('xyplot')
```

```
[1] "xyplot"
```

- ▶ Definimos un método para la clase `lm` usando `xyplot`.

```
setMethod('xyplot',  
  signature=c(x='lm', data='missing'),  
  definition=function(x, data,  
    ...){  
    fitted <- fitted(x)  
    residuals <- residuals(x)  
    xyplot(residuals ~ fitted,...)  
  })
```

# Ejemplo con lm y xyplot

- Recuperamos la regresión que empleamos en el apartado de Estadística:

```
lmFertEdu <- lm(Fertility ~ Education, data = swiss)
```

```
xyplot(lmFertEdu, col='red', pch=19, type=c('p', 'g'))
```

