

Clases y Métodos

Oscar Perpiñán Lamigueiro \ <http://oscarperpinan.github.io>

OOP en R

Programación Orientada a
Objetos (OOP)

Clases y métodos S3

Clases

Métodos

Clases y métodos S4

Clases en S4

Métodos en S4

Clases S3 con clases y
métodos S4

OOP en R

Programación Orientada a Objetos (OOP)

Clases y métodos S3

Clases y métodos S4

Programación Orientada a Objetos (OOP)

Clases y Métodos

Oscar Perpiñán
Lamigueiro \ [http://
oscarperpinan.
github.io](http://oscarperpinan.github.io)

- ▶ 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.

OOP en R

Programación Orientada a
Objetos (OOP)

Clases y métodos
S3

Clases

Métodos

Clases y métodos
S4

Clases en S4

Métodos en S4

Clases S3 con clases y
métodos S4

Referencias

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

OOP en R

Programación Orientada a
Objetos (OOP)

Clases y métodos S3

Clases

Métodos

Clases y métodos S4

Clases en S4

Métodos en S4

Clases S3 con clases y
métodos S4

OOP en R

Clases y métodos S3

Clases

Métodos

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
```

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) <- 'Task'
task1
```

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

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

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

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

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

Clases y Métodos

Oscar Perpiñán
Lamigueiro \ [http://
oscarperpinan.
github.io](http://oscarperpinan.github.io)

OOP en R

Programación Orientada a
Objetos (OOP)

Clases y métodos
S3

Clases

Métodos

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] "Task"

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

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

$priority
[1] "High"

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

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

Clases y Métodos

Oscar Perpiñán
Lamigueiro \ [http://
oscarperpinan.
github.io](http://oscarperpinan.github.io)

OOP en R

Programación Orientada a
Objetos (OOP)

Clases y métodos
S3

Clases

Métodos

Clases y métodos
S4

Clases en S4

Métodos en S4

Clases S3 con clases y
métodos S4

OOP en R

Clases y métodos S3

Clases

Métodos

Clases y métodos S4

Métodos con S3

- ▶ Sencillos de usar e implementar.
- ▶ Poco robustos.
- ▶ Se definen a partir de un método genérico, añadiendo a la función el nombre de la clase con un punto como separador.

```
print print.data.frame  
summary summary.lm
```

Clases y Métodos

Oscar Perpiñán
Lamigueiro \ [http://
oscarperpinan.
github.io](http://oscarperpinan.github.io)

OOP en R

Programación Orientada a
Objetos (OOP)

Clases y métodos
S3

Clases

Métodos

Clases y métodos
S4

Clases en S4

Métodos en S4

Clases S3 con clases y
métodos S4

Métodos genéricos: UseMethod

- UseMethod sirve para elegir el método correspondiente a la clase del objeto empleado como argumento en la función.
- Se debe definir un método genérico, incluyendo llamada a UseMethod.

summary

```
function (object, ...)
UseMethod("summary")
<bytecode: 0x5640bb025728>
<environment: namespace:base>
```

- Si no hay un método definido para la clase del objeto, UseMethod ejecuta la función por defecto:

summary.default

```
function (object, ..., digits)
{
  if (is.factor(object))
    return(summary.factor(object, ...))
  else if (is.matrix(object)) {
    if (missing(digits))
      return(summary.matrix(object, ...))
    else return(summary.matrix(object, digits = digits, ...))
  }
}
```

Métodos genéricos: UseMethod

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

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

```
Funcion genérica
[1] -1.58026882 -1.61878387 -1.02965731  0.29374595 -0.40879692  2.27929624
[7] -0.69849094 -0.45129074 -1.27135509  0.08937423
```

```
myFun(task1)
```

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

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

$priority
[1] "Low"

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

Clases y Métodos

Oscar Perpiñán
Lamigueiro \ <http://oscarperpinan.github.io>

OOP en R

Programación Orientada a
Objetos (OOP)

Clases y métodos
S3

Clases

Métodos

Clases y métodos
S4

Clases en S4

Métodos en S4

Clases S3 con clases y
métodos S4

methods

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

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

```
[1] myFun.default  
see '?methods' for accessing help and source code
```

```
methods('summary')
```

```
[1] summary.aov  
[3] summary.aspell*  
[5] summary.connection  
[7] summary.Date  
[9] summary.ecdf*  
[11] summary.glm  
[13] summary.lm  
[15] summary.manova  
[17] summary.nlm*  
[19] summary.packageStatus*  
[21] summary.PDF_Stream*  
[23] summary.POSIXlt  
[25] summary.prcomp*  
[27] summary.proc_time  
[29] summary.srcfile  
[31] summary.stepfun  
[33] summary.table  
[35] summary.tukeysmooth*  
see '?methods' for accessing help and source code  
  
summary.aovlist*  
summary.check_packages_in_dir*  
summary.data.frame  
summary.default  
summary.factor  
summary.infl*  
summary.loess*  
summary.matrix  
summary.nls*  
summary.PDF_Dictionary*  
summary.POSIXct  
summary.ppr*  
summary.princomp*  
summary.shingle*  
summary.srcref  
summary.stl*  
summary.trellis*
```

Clases y Métodos

Oscar Perpiñán
Lamigueiro \ [http://
oscarperpinan.
github.io](http://oscarperpinan.github.io)

OOP en R

Programación Orientada a
Objetos (OOP)

Clases y métodos
S3

Clases

Métodos

Clases y métodos
S4

Clases en S4

Métodos en S4

Clases S3 con clases y
métodos S4

Definición del método para Task

```
myFun.Task <- 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.Task
see '?methods' for accessing help and source code
```

```
methods(class='Task')
```

```
[1] myFun
see '?methods' for accessing help and source code
```

Clases y Métodos

Oscar Perpiñán
Lamigueiro \
[http://
oscarperpinan.
github.io](http://oscarperpinan.github.io)

OOP en R

Programación Orientada a
Objetos (OOP)

Clases y métodos
S3

Clases

Métodos

Clases y métodos
S4

Clases en S4

Métodos en S4

Clases S3 con clases y
métodos S4

NextMethod

Incluyendo NextMethod en un método específico
llamamos al método genérico (default).

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

```
print(task1)
```

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

Clases y Métodos

Oscar Perpiñán
Lamigueiro \
[http://
oscarperpinan.
github.io](http://oscarperpinan.github.io)

OOP en R

Programación Orientada a
Objetos (OOP)

Clases y métodos
S3

Clases

Métodos

Clases y métodos
S4

Clases en S4

Métodos en S4

Clases S3 con clases y
métodos S4

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] "Task"
```

```
[[2]]
```

```
Task:
```

```
$what
```

```
[1] "Find and fix bugs"
```

```
$when
```

```
[1] "2013-03-15"
```

Clases y Métodos

Oscar Perpiñán
Lamigueiro \
[http://
oscarperpinan.
github.io](http://oscarperpinan.github.io)

OOP en R

Programación Orientada a
Objetos (OOP)

Clases y métodos S3

Clases

Métodos

Clases y métodos S4

Clases en S4

Métodos en S4

Clases S3 con clases y
métodos S4

Definición de un método S3 para Task

```
print.Task <- 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

Clases y Métodos

Oscar Perpiñán
Lamigueiro \ [http://
oscarperpinan.
github.io](http://oscarperpinan.github.io)

OOP en R

Programación Orientada a
Objetos (OOP)

Clases y métodos
S3

Clases

Métodos

Clases y métodos
S4

Clases en S4

Métodos en S4

Clases S3 con clases y
métodos S4

Definición de un método S3 para ToDo3

- Definimos un método más sofisticado para la clase ToDo3 **sin** tener en cuenta el método definido para la clase Task.

```
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  
-----
```

Redefinición del método para ToDo3

- Podemos aligerar el código teniendo en cuenta el método definido para la clase Task.

```
print.ToDo3 <- function(x, ...){  
  cat('This is my ToDo list:\n')  
  ## Cada uno de los elementos de un  
  ## objeto ToDo3 son Task. Por tanto,  
  ## x[[i]] es de clase Task y  
  ## print(x[[i]]) ejecuta el metodo  
  ## print.Task  
  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  
-----
```

OOP en R

Clases y métodos S3

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.
- ▶ `slots`: 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 `slots`.
- ▶ `validity`: a función que comprueba la validez de la clase creada con la información suministrada.

Definición de una nueva clase

```
setClass('task',  
  slots = c(  
    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"
```

Clases y Métodos

Oscar Perpiñán
Lamigueiro \
[http://
oscarperpinan.
github.io](http://oscarperpinan.github.io)

OOP en R

Programación Orientada a
Objetos (OOP)

Clases y métodos
S3

Clases

Métodos

Clases y métodos
S4

Clases en S4

Métodos en S4

Clases S3 con clases y
métodos S4

Creación de un objeto con la clase definida:

`new`

Una vez que la clase ha sido definida con `setClass`, se puede crear un objeto nuevo con `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"
```

Clases y Métodos

Oscar Perpiñán
Lamigueiro \ [http://
oscarperpinan.
github.io](http://oscarperpinan.github.io)

OOP en R

Programación Orientada a
Objetos (OOP)

Clases y métodos
S3

Clases
Métodos

Clases y métodos
S4

Clases en S4

Métodos en S4

Clases S3 con clases y
métodos S4

Funciones para crear objetos

Es habitual definir funciones que construyen y modifican objetos para evitar el uso de new:

```
createTask <- function(what, when, priority){  
  new('task',  
    what = what,  
    when = when,  
    priority = priority)  
}
```

```
task2 <- createTask(what = 'Write an email',  
  when = as.Date('2013-01-01'),  
  priority = 'Low')
```

```
createTask('Oops', 'Hoy', 3)
```

```
Error in validObject(.Object) :
```

```
  invalid class "task" object: 1: invalid object for slot "when" in class "task": got class "character", should be "Date"  
invalid class "task" object: 2: invalid object for slot "priority" in class "task": got class "numeric", should be "character"
```

Clases y Métodos

Oscar Perpiñán
Lamigueiro \ [http://
oscarperpinan.
github.io](http://oscarperpinan.github.io)

OOP en R

Programación Orientada a
Objetos (OOP)

Clases y métodos
S3

Clases
Métodos

Clases y métodos
S4

Clases en S4

Métodos en S4

Clases S3 con clases y
métodos S4

Definición de la clase ToDo

Clases y Métodos

Oscar Perpiñán
Lamigueiro \ [http://
oscarperpinan.
github.io](http://oscarperpinan.github.io)

```
setClass('ToDo',  
  slots = c(tasks = 'list')  
)
```

```
myList <- new('ToDo',  
  tasks = list(  
    t1 = task1,  
    t2 = task2))
```

OOP en R

Programación Orientada a
Objetos (OOP)

Clases y métodos
S3

Clases

Métodos

Clases y métodos
S4

Clases en S4

Métodos en S4

Clases S3 con clases y
métodos S4

Acceso a los slots

Para extraer información de los *slots* hay que emplear @ (a diferencia de \$ en listas y `data.frame`)

```
myList@tasks
```

```
$t1  
An object of class "task"  
Slot "what":  
[1] "Find and fix bugs"
```

```
Slot "when":  
[1] "2013-03-15"
```

```
Slot "priority":  
[1] "High"
```

```
$t2  
An object of class "task"  
Slot "what":  
[1] "Write an email"
```

```
Slot "when":  
[1] "2013-01-01"
```

```
Slot "priority":  
[1] "Low"
```

Clases y Métodos

Oscar Perpiñán
Lamigueiro \
[http://
oscarperpinan.
github.io](http://oscarperpinan.github.io)

OOP en R

Programación Orientada a
Objetos (OOP)

Clases y métodos
S3

Clases
Métodos

Clases y métodos
S4

Clases en S4

Métodos en S4

Clases S3 con clases y
métodos S4

Acceso a los slots

El *slot* tasks es una lista: empleamos \$ para acceder a sus elementos

```
myList@tasks$t1
```

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

Cada elemento de tasks es un objeto de clase task: empleamos @ para extraer sus *slots*.

```
myList@tasks$t1@what
```

```
[1] "Find and fix bugs"
```

Clases y Métodos

Oscar Perpiñán
Lamigueiro \ [http://
oscarperpinan.
github.io](http://oscarperpinan.github.io)

OOP en R

Programación Orientada a
Objetos (OOP)

Clases y métodos
S3

Clases
Métodos

Clases y métodos
S4

Clases en S4

Métodos en S4

Clases S3 con clases y
métodos S4

Problema con los slots definidos como list

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

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

Clases y Métodos

Oscar Perpiñán
Lamigueiro \ [http://
oscarperpinan.
github.io](http://oscarperpinan.github.io)

OOP en R

Programación Orientada a
Objetos (OOP)

Clases y métodos
S3

Clases

Métodos

Clases y métodos
S4

Clases en S4

Métodos en S4

Clases S3 con clases y
métodos S4

Validación

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',  
  slots = c(tasks = 'list'),  
  validity=valida  
)
```

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

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

Funciones para crear y modificar objetos

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

```
addTask <- function(object, task){  
  ## La siguiente comprobación sólo es necesaria si  
  la  
  ## definición de la clase *no* incorpora una  
  función  
  ## validity  
  stopifnot(is(task, 'task'))  
  object@tasks <- c(object@tasks, task)  
  object  
}
```

Clases y Métodos

Oscar Perpiñán
Lamigueiro \
[http://
oscarperpinan.
github.io](http://oscarperpinan.github.io)

OOP en R

Programación Orientada a
Objetos (OOP)

Clases y métodos
S3

Clases

Métodos

Clases y métodos
S4

Clases en S4

Métodos en S4

Clases S3 con clases y
métodos S4

OOP en R

Clases y métodos S3

Clases y métodos S4

- Clases en S4

- Métodos en S4

- Clases S3 con clases y métodos S4

Métodos en S4: setMethod

- ▶ 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` y `standardGeneric`

```
setGeneric('myMethod',  
          function(x, y, ...)  
            standardGeneric('myMethod')  
          )
```

```
[1] "myMethod"
```

```
setGeneric('print')
```

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

Métodos en S4: setGeneric y getGeneric

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

```
library(lattice)
isGeneric('xyplot')
```

```
[1] TRUE
```

```
getGeneric('xyplot')
```

```
standardGeneric for "xyplot" defined from package "lattice"
```

```
function (x, data, ...)
  standardGeneric("xyplot")
<bytecode: 0x5640bc7712d0>
<environment: 0x5640bc58d960>
Methods may be defined for arguments: x, data
Use  showMethods("xyplot")  for currently available ones.
```

Clases y Métodos

Oscar Perpiñán
Lamigueiro \ [http://
oscarperpinan.
github.io](http://oscarperpinan.github.io)

OOP en R

Programación Orientada a
Objetos (OOP)

Clases y métodos
S3

Clases
Métodos

Clases y métodos
S4

Clases en S4

Métodos en S4

Clases S3 con clases y
métodos S4

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
```

Clases y Métodos

Oscar Perpiñán
Lamigueiro \
[http://
oscarperpinan.
github.io](http://oscarperpinan.github.io)

OOP en R

Programación Orientada a
Objetos (OOP)

Clases y métodos
S3

Clases
Métodos

Clases y métodos
S4

Clases en S4

Métodos en S4

Clases S3 con clases y
métodos S4

Definición de un método print para ToDo

```
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 y Métodos

Oscar Perpiñán
Lamigueiro \
[http://
oscarperpinan.
github.io](http://oscarperpinan.github.io)

OOP en R

Programación Orientada a
Objetos (OOP)

Clases y métodos
S3

Clases

Métodos

Clases y métodos
S4

Clases en S4

Métodos en S4

Clases S3 con clases y
métodos S4

OOP en R

Clases y métodos S3

Clases y métodos S4

- Clases en S4

- Métodos en S4

- Clases S3 con clases y métodos S4

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')
```

```
getClass('lm')
```

```
Virtual Class "lm" [package "methods"]
```

```
Slots:
```

```
Name: .S3Class
```

```
Class: character
```

```
Extends: "oldClass"
```

```
Known Subclasses:
```

```
Class "mlm", directly
```

```
Class "aov", directly
```

```
Class "glm", directly
```

```
Class "maov", by class "mlm", distance 2
```

```
Class "glm.null", by class "glm", distance 2
```

Clases y Métodos

Oscar Perpiñán
Lamigueiro \ [http://
oscarperpinan.
github.io](http://oscarperpinan.github.io)

OOP en R

Programación Orientada a
Objetos (OOP)

Clases y métodos
S3

Clases

Métodos

Clases y métodos
S4

Clases en S4

Métodos en S4

Clases S3 con clases y
métodos S4

Ejemplo con lm y xyplot

Definimos un método genérico para xyplot

```
library(lattice)
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,...)
  })
```

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

Clases y Métodos

Oscar Perpiñán
Lamigueiro \ [http://
oscarperpinan.
github.io](http://oscarperpinan.github.io)

OOP en R

Programación Orientada a
Objetos (OOP)

Clases y métodos
S3

Clases
Métodos

Clases y métodos
S4

Clases en S4

Métodos en S4

Clases S3 con clases y
métodos S4

Ejemplo con lm y xyplot

Recuperamos la regresión que empleamos en el apartado de Estadística:

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

Call:

```
lm(formula = Fertility ~ Education, data = swiss)
```

Residuals:

Min	1Q	Median	3Q	Max
-17.036	-6.711	-1.011	9.526	19.689

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	79.6101	2.1041	37.836	< 2e-16 ***
Education	-0.8624	0.1448	-5.954	3.66e-07 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.446 on 45 degrees of freedom

Multiple R-squared: 0.4406, Adjusted R-squared: 0.4282

F-statistic: 35.45 on 1 and 45 DF, p-value: 3.659e-07

Clases y Métodos

Oscar Perpiñán
Lamigueiro \ [http://
oscarperpinan.
github.io](http://oscarperpinan.github.io)

OOP en R

Programación Orientada a
Objetos (OOP)

Clases y métodos
S3

Clases

Métodos

Clases y métodos
S4

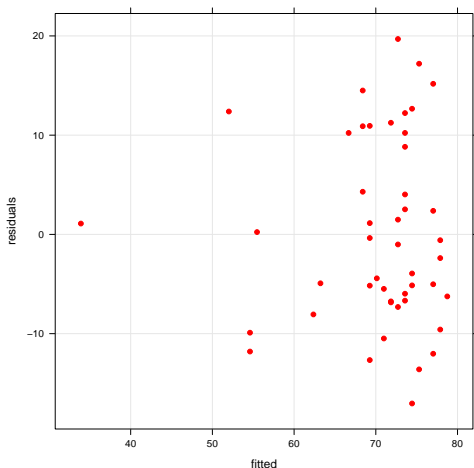
Clases en S4

Métodos en S4

Clases S3 con clases y
métodos S4

Ejemplo con lm y xyplot

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



Clases y Métodos

Oscar Perpiñán
Lamigueiro \ [http://
oscarperpinan.
github.io](http://oscarperpinan.github.io)

OOP en R

Programación Orientada a
Objetos (OOP)

Clases y métodos
S3

Clases

Métodos

Clases y métodos
S4

Clases en S4

Métodos en S4

Clases S3 con clases y
métodos S4