Introduce S3 and R6 Systems

Object-Oriented Programming in R

r08h41005 統計一 張和家

Outline

- Problems we are facing
- R's OOP systems
- S3 example
- R6 example
- Trade-offs

Problems we are facing

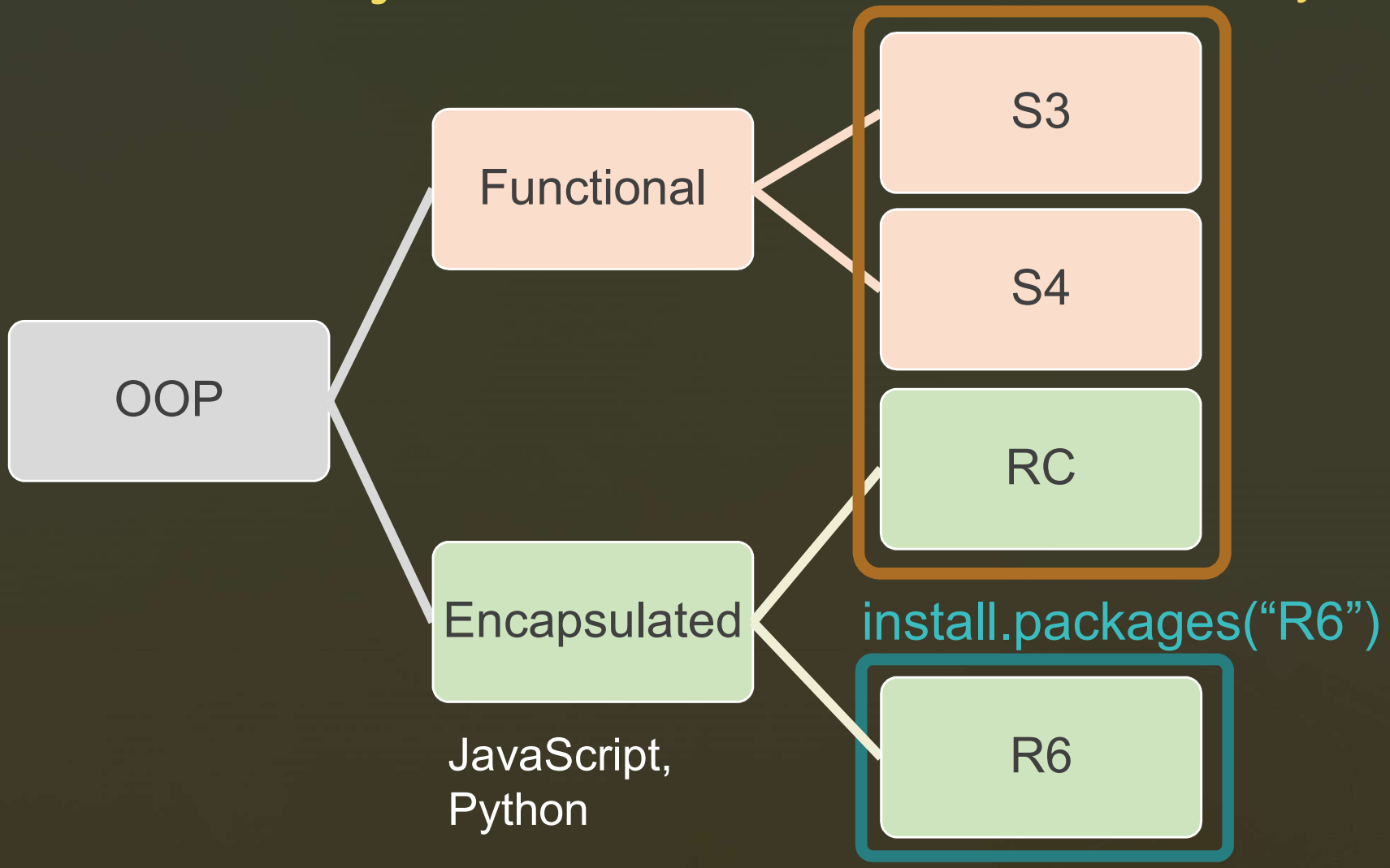
- Manage lots of things
- Naming things is hard
- Side effect or return a value

```
# Pseudo code

# methods belong to functions
t.test(normal)
ks.test(gamma)
chisq.test(gumbel)

# methods belongs to data
normal$test() # t.test()
gamma$test()  # ks.test()
gumbel$test() # chisq.test()
```

R's OOP systems



S3 example

```
x <- rnorm(10)
y <- t.test(x)
# class is htest
attributes(y)
#> $names
#>  [1] "statistic"   "parameter"   "p.value"
"conf.int"    "estimate"
#>  [6] "null.value"  "stderr"      "alternative" "method"
"data.name"
#>
#> $class
#> [1] "htest"
# rnorm() is a function
sloop::ftype(rnorm)
#> [1] "function"
# object x is a base object
sloop::otype(x)
#> [1] "base"
```

* Objects are not mutable

* Not all objects are OOP objects

```r
# t.test() is a S3 generic function
sloop::ftype(t.test)
#> [1] "S3"      "generic"
# object t_test is a S3 object
sloop::otype(y)
#> [1] "S3"
# an S3 generic is to perform method dispatch
# method dispatch is performed by UseMethod()
t.test
#> function (x, ...)
#> UseMethod("t.test")
#> <bytecode: 0x0000000012fe6720>
#> <environment: namespace:stats>
# => indicates the method that is called
# method t.test.default is called
sloop::s3_dispatch(t.test(x))
#>    t.test.double
#>    t.test.numeric
#> => t.test.default
```

* Methods are separated from objects and belong to functions

```
# list all methods that belong to t.test generic
sloop::s3_methods_generic("t.test")
#> # A tibble: 2 x 4
#>   generic class   visible source
#>   <chr>   <chr>   <lgl>   <chr>
#> 1 t.test  default FALSE   registered S3method
#> 2 t.test  formula FALSE   registered S3method
# you can also use formula in t.test()
t.test(extra ~ group, data = sleep)
#>
#>  Welch Two Sample t-test
#>
#> data:  extra by group
#> t = -1.8608, df = 17.776, p-value = 0.07939
#> alternative hypothesis: true difference in means is not equal to 0
#> 95 percent confidence interval:
#>  -3.3654832  0.2054832
#> sample estimates:
#> mean in group 1 mean in group 2
#>            0.75            2.33
```

```
# list all generics that have a method for htest class
sloop::s3_methods_class("htest")
#> # A tibble: 1 x 4
#>   generic class visible source
#>   <chr>   <chr> <lgl>   <chr>
#> 1 print   htest FALSE   registered S3method
# method print.htest is called
sloop::s3_dispatch(print(t.test(x)))
#> => print.htest
#>  * print.default
t.test(x)
#>
#>  One Sample t-test
#>
#> data:  x
#> t = -1.4537, df = 9, p-value = 0.18
#> alternative hypothesis: true mean is not equal to 0
#> 95 percent confidence interval:
#>  -0.7843667  0.1706541
#> sample estimates:
#>  mean of x
#>  -0.3068563
```

R6 example

```
# create a R6 class
TTest <- R6Class("TTest",
  public = list(
    result = NULL,
    test = function(...) {
      self$result <- t.test(...)
      invisible(self)
    }
))
# create a R6 object
x <- TTest$new()
# call its method
x$test(rnorm(10))
```

* Objects are mutable and contain data and methods

```
# call its field
x$result
#>
#>  One Sample t-test
#>
#> data:  x
#> t = 1.0489, df = 9, p-value = 0.3216
#> alternative hypothesis: true mean is not equal to 0
#> 95 percent confidence interval:
#>  -0.4051709  1.1057536
#> sample estimates:
#> mean of x
#> 0.3502913
# the same idea used in JavaScript and Python
x$test(rnorm(10))$result
#>
#>  One Sample t-test
# Also similar to the idea of pipe operator in tidyverse
rnorm(10) %>% t.test()
#>
#>  One Sample t-test
```

```r
# create a R6 class
TTest <- R6Class("TTest",
  public = list(
    data = NULL,
    test = function(...) {
      self$data <- list(...)
      t.test(...)
    }
))
# create a R6 object
x <- TTest$new()
# call its method
x$test(rnorm(10))
#>
#>  One Sample t-test
#>
# call its field
x$data
#> [[1]]
#>  [1] -0.7170052  0.3282419 -0.7891339  0.5054498
```

* R6 methods can return a value and modify an object simultaneously

Functional	Objects contain data
	Class methods are separated from objects
	Objects are not mutable
Encapsulated	Objects contain data and methods
	Objects are mutable

Trade-offs

- R6 Real case: Shiny
- R6 methods can return a value and modify an object simultaneously
- R6 is faster than S3 method dispatch
- R6 is easier to name the objects
- R6 is odd
- Debug

Reference

- R6 document
- The R6 class system