Advanced R for Econometricians (Summer 2025)

Object Oriented R Programming — Solutions to Exercises

Martin Arnold, Martin Schmelzer

Slide 18

1.

Because of S3's `generic.class` naming scheme, both functions may initially look similar, while they are in fact unrelated.

`t.test()` is a generic function that performs a t-test. `t.data.frame()` is a method that gets called by the generic `t()` to transpose dataframe input. Due to R's S3 dispatch rules, `t.test()` would also get called when `t()` is applied to an object of class test

2.

We obtain an object of class `ecdf` (short for empirical cumulative distribution function) with the superclasses `stepfun` and `function`. The `ecdf` object is built on the base type `closure` (a function). The expression which was used to create it (`rpois(100, 10)`), is stored in the `call` attribute.

3.

The code returns a `table` object, which is built upon the integer type. The attribute `dimnames` is used to name the elements of the integer vector.

Slide 36

1.

We see that `t.test()` is a generic because it calls `UseMethod()`:

```
t.test
```

```
## function (x, ...)
## UseMethod("t.test")
## <bytecode: 0x139452890>
## <environment: namespace:stats>
```

```
# or simply call
sloop::ftype(t.test)
```

```
## [1] "S3"      "generic"
```

Interestingly, R also provides helpers, which list functions that look like methods, but in fact are not:

```
tools::nonS3methods("stats")
```

```
## [1] "anova.lmlist"        "expand.model.frame"  "fitted.values"
## [4] "influence.measures"  "lag.plot"            "t.test"
## [7] "plot.spec.phase"     "plot.spec.coherency"
```

When we create an object with class test, `t()` dispatches to the `t.default()` method. This is because `UseMethod()` simply searches for functions named `paste0("generic", ".", c(class(x), "default"))`.

```
x <- structure(1:10, class = "test")

t(x)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## [1,]    1    2    3    4    5    6    7    8    9    10
## attr(,"class")
## [1] "test"
```

However, in older versions of `R` (pre 4.0.0) this behaviour was slightly different. Instead of dispatching to the `t.default()` method, the `t.test()` generic was erroneously treated as a method of `t()` which then dispatched to `t.test.default()` or (if defined) to `t.test.test()`.

2.

This is a simple application of `sloop::s3_methods_class()`:

```
s3_methods_class("table")
```

```
## # A tibble: 10 x 4
##    generic       class visible source
##    <chr>         <chr> <lgl>   <chr>
##  1 [             table TRUE    base
##  2 aperm         table TRUE    base
##  3 as.data.frame table TRUE    base
##  4 Axis          table FALSE   registered S3method
##  5 lines         table FALSE   registered S3method
##  6 plot          table FALSE   registered S3method
##  7 points        table FALSE   registered S3method
##  8 print         table TRUE    base
##  9 summary       table TRUE    base
## 10 tail          table FALSE   registered S3method
```

Interestingly, the `table` class has a number of methods designed to help plotting with base graphics.

3.

```
prime <- function(x = integer()) {
  validate_prime(new_prime(x))
}

validate_prime <- function(x = list()) {
  sapply(
    unique(unlist(x)), function(z) {
      if(!all(z==2 || z %% 2:(z-1) > 0)) {
        stop('Input contains non-prime number(s)!', call. = F)
      }
    }
  )
  x
}

new_prime <- function(x = integer()) {
  stopifnot(is.integer(x))
```

```
  x <- list(x)
  class(x) <- 'prime'
  x
}
```

3.1

```
summary.prime <- function(x) {
  if(class(x)  == "prime") {
    x <- unlist(x)
    out <- list(
        name = quote(x),
        n = length(x),
        min = min(x),
        max = max(x)
        )
    class(out) <- "summary.prime"
    return(out)
  } else {
    message("Object not of class prime!")
  }
}

my_primes <- new_prime(c(3L, 5L, 7L))

summary(my_primes)
```

```
## $name
## x
##
## $n
## [1] 3
##
## $min
## [1] 3
##
## $max
## [1] 7
##
## attr(,"class")
## [1] "summary.prime"
```

3.2

```
print.summary.prime <- function(x) {
  if(class(x)  == "summary.prime") {
    cat(
      "summary for", x$name, ":\n\n",
      x$n, "prime numbers between", x$min, "to", x$max
    )
  } else {
    message("Object not of class summary.prime!")
  }
```

```
}

# now with print method
summary(my_primes)
```

```
## summary for x :
##
##  3 prime numbers between 3 to 7
```