R7 > S3 + S4

and Bioconductor

Henrik Bengtsson, Jim Hester, Will Landau, Michael Lawrence, Martin Maechler, Luke Tierney, Hadley Wickham
library(rtracklayer)

loads

129 classes,

1114 generics and

6304 methods
S4 is essential for interoperable, domain-specific frameworks

- **Formal** class definitions with **self-documenting** structures
- **Automatic** validation
- **Multiple** (particularly binary) dispatch
But S4 challenges users and developers

- **Syntax is unnatural** (side effects)
- **Hard to understand** multiple inheritance and multiple dispatch
- **Lack of transparency** of object structure and methods
  - One of R’s strengths is that it makes data *tactile*: users can intuitively manipulate their data
  - But S4 (at least how we tend to use it) undermines that by making objects *opaque*
- **Not part of base**
- **Awkwardly coexists with S3**

Not to mention the *methods* implementation has

- Poor performance and is
- Difficult to maintain
Hypothesis

There can be an extension of S3 that adds many of the important S4 features without introducing (yet) another object-oriented system.
Problems

- Syntax is unnatural (side effects)
- Complex namespace declarations that few understand

Solutions

- Class objects constructed and exported like other objects
  
  ```r
  text <- new_class("text", parent = "character",
                    constructor = function(text) new_object(.data = text))
  ```

- The class object is the constructor: `object <- text("hi")`

- Method definition is simple and intuitive (could be same as S3)
  
  ```r
  method(foo, text) <- function(x, ...) paste0("foo-", x)
  ```
Problem

- Multiple inheritance and dispatch hard to understand

Solutions

- Single inheritance
- Multiple dispatch through nested single dispatch
Single inheritance is feasible via composition

**Use case:** CompressedFactorList contains both FactorList and CompressedList

Instead, we could store a slot of new type ListStore, with SimpleStore and CompressedStore subclasses.

**Challenge:** how to implement optimized `levels()` method that takes advantage of compression?

**One option:** mangled generics that take the store strategy, along with the data:

```r
setGeneric("levels_List", function(x, store) standardGeneric("levels_List"))

setMethod("levels_List", c("FactorList", "CompressedStore", function(x, store) {
    setNames(rep(CharacterList(levels(store@unlistData)), length(x)), names(x))
}))
```
Just an idea: compositional dispatch

Allows optimization of top-level methods based on composition:

```r
method(levels, "FactorList", when = c(store = "CompressedStore")) <- function(x) {
  setNames(rep(CharacterList(levels(x@store@unlistData)), length(x)), names(x))
}
```
Problem

- Lack of transparency of object structure and methods

Solutions

- Intuitive alternative to `selectMethod()`:
  
  ```r
  method(foo, text)
  ```

- Classes can have *properties* that expose a virtual structure without breaking encapsulation (need getter and setter)

```r
properties = list(
  start = "numeric",
  end = "numeric",
  new_property(
    name = "length",
    class = "numeric",
    getter = function(x)
      x@end - x@start,
    setter = function(x, value) {
      x@end <- x@start + value
      x
    }
  )
)
```
Presentation of prototype
Jim Hester
How might Bioconductor adopt R7?

- Top-down, bottom-up, or some combination?
- **R7 is just an extension of S3**, which is compatible with S4, so:
  - R7 generics and methods should just work on S4 objects
  - S4 classes should be able to extend R7 classes via `setOldClass()`
    - Would enable top-down approach
- Extending S4 classes with R7 would take more work
  - Since S4 generics require S4 classes (no equivalent of `setOldClass()` for S4 extensions)
  - But would be ideal because it would enable bottom-up experimentation in “leaf” packages
- Serialized objects might also preclude removing S4 classes
  - Could explore enabling auto-updating for a converted class
- Luckily, we control everything and might be able to solve these problems