S4 System Development in Bioconductor

Patrick Aboyoun

Fred Hutchinson Cancer Research Center

20-21 May, 2010
Introduction
  Object-Oriented Programming in R
  Role of S4 in Bioconductor

S4 Basics
  S4 Classes
  S4 Generic Functions & Methods

S4 Exercises

S4 Constraints
  Copy on Slot Modification
  Object Overhead
  Method Dispatch

S4 Case Studies
  Slot-Oriented Virtual Class (eSet)
  Method-Oriented Virtual Class (Sequence)
  Multiple Inheritance & Vectorization (CompressedIRangesList)
  Build or Reuse? (CompressedIRangesList)
  Class Union & Group Generic (Rle)

Resources
BioC Motivation for OOP

OOP: Minimizing inaccurate inputs & modularizing inputs and outputs.
Outline

Introduction
Object-Oriented Programming in R
Role of S4 in Bioconductor

S4 Basics
S4 Classes
S4 Generic Functions & Methods

S4 Exercises

S4 Constraints
Copy on Slot Modification
Object Overhead
Method Dispatch

S4 Case Studies
Slot-Oriented Virtual Class (eSet)
Method-Oriented Virtual Class (Sequence)
Multiple Inheritance & Vectorization (CompressedIRangesList)
Build or Reuse? (CompressedIRangesList)
Class Union & Group Generic (Rle)

Resources
Two Programming Systems in R

- R is an implementation of the S3 language by Chambers and Hastie (1992).
- S3 uses a “class” attribute to mimic aspects of OOP.
- S4 is a formal OOP system.
- The methods package implements the S4 system as described in Chambers (1998) and Chambers (2008).
Main Features of Bioconductor

S4 superiority to S3 in realizing BioC features

- The R project for statistical computing
- Documentation and **reproducible research**
- Statistical and graphical **methods for genomic data**
- **Genomic annotations**
- Open source
- Open development

Reasons

- Explicit representation of classes
- Validity checking of instances
- Methods registered with generics
Use of S4 in Bioconductor

Statistics on packages in BioC 2.6

- 197 of 389 (51%) BioC packages define S4 classes
- 97 use inheritance
  - traditional parents: ExpressionSet and eSet from Biobase
  - newer prolific parent: Sequence from IRanges
  - common off-beat parent: list, an S3 type
- 30 define virtual classes
- 14 use multiple inheritance
Outline

Introduction
Object-Oriented Programming in R
Role of S4 in Bioconductor

S4 Basics
S4 Classes
S4 Generic Functions & Methods

S4 Exercises

S4 Constraints
Copy on Slot Modification
Object Overhead
Method Dispatch

S4 Case Studies
Slot-Oriented Virtual Class (eSet)
Method-Oriented Virtual Class (Sequence)
Multiple Inheritance & Vectorization (CompressedIRangesList)
Build or Reuse? (CompressedIRangesList)
Class Union & Group Generic (Rle)

Resources
S4 Framework

S4 Components

**Class**
- is virtual?
- slots (properties)
- contains (inherits)
- prototype
- validity function

**Generic Function**
- dispatch signature
- dispatch method

**Methods***
- signature mapping
- function definition

* Grouped in methods list

Important differences with Java and C++

1. Classes don’t own methods.
2. Design goal: Be “vectorized”.
   (a) Class slots ideally vectors and matrices.
   (b) Methods avoid unnecessary looping.
S4 Class Creation

**setClass function arguments**

- Class – the class name.
- representation – the slot definitions.
- contains – the parent classes.
- prototype – the slot values for a default instance.
- validity – a function that checks instance validity.

**new constructor arguments**

- Class – the class name.
- ... – typically, slot values.
Canonical Example in S4 (1/2)

Class definition

```r
> setClass("Greeting",
+ representation =
+ representation(phrase = "character"),
+ prototype = prototype(phrase = "Hello world!"))
```

Default instantiation

```r
> new("Greeting")
```

An object of class "Greeting"
Slot "phrase":
[1] "Hello world!"

Customized instantiation

```r
> new("Greeting", phrase = "ciao")
```

An object of class "Greeting"
Slot "phrase":
[1] "ciao"
Validity method

```r
> setValidity("Greeting",
+   function(object) {
+     if (length(object@phrase) != 1)
+       "'phrase' not a single string"
+     else if (!nzchar(object@phrase))
+       "'phrase' is empty"
+     else
+       TRUE
+   })
```

Invalid object instantiation

```r
> new("Greeting", phrase = "")
Error in validObject(.Object) :
  invalid class "Greeting" object: 'phrase' is empty
```
Creating New S4 Objects from Old

initialize function arguments

► .Object – an S4 object.
► … – typically, new slot values.

Creation via initialize

```r
> nametag <- new("Greeting")
> nametag <- initialize(nametag, phrase = "Welcome")
> nametag

An object of class "Greeting"
Slot "phrase":
[1] "Welcome"
```
S4 Generic Function & Method Creation

**setGeneric function arguments**

- **name** – the generic function name.
- **def** – the generic definition, typically of the form
  
  ```
  function(<<ARG_LIST>>) standardGeneric("<<GEN_NAME>>")
  ```

- **signature** – the method dispatch signature.

**setMethod function arguments**

- **f** – the generic function name.
- **signature** – the mapping of arguments in dispatch signature to classes.
- **definition** – the method definition.
Generic function definition

```r
> setGeneric("words", signature = "x",
+       function(x) standardGeneric("words"))
```

Method definition

```r
> setMethod("words", c("x" = "Greeting"),
+       function(x)
+       {
+           ans <- strsplit(x@phrase, "\W")[[1L]]
+           sort(unique(ans[nzchar(ans)]))
+       })
```

Method invocation

```r
> words(new("Greeting"))
```

[1] "Hello" "world"
Method definition

```r
> setMethod("words", c("x" = "character"),
+ function(x)
+ {
+   ans <- unlist(strsplit(x, "\W"),
+   use.names = FALSE)
+   sort(unique(ans[nzchar(ans)]))
+ })
> setMethod("words", c("x" = "Greeting"),
+ function(x) callGeneric(x@phrase))
```

Method invocation

```r
> words(new("Greeting"))

[1] "Hello" "world"
```
Implicit Generic Method for Canonical Example

Non-S4 generic function

> showMethods("nchar")

Function: nchar (package base)
  x="AlignedXStringSet0"
  x="ANY"
  x="character"
    (inherited from: x="ANY")
  x="CompressedCharacterList"
  x="CompressedRleList"
  x="MaskedXString"
  x="PairwiseAlignedFixedSubjectSummary"
  x="PairwiseAlignedXStringSet"
  x="Rle"
  x="SimpleCharacterList"
  x="SimpleRleList"
  x="XString"
  x="XStringSet"
  x="XStringViews"
Implicit Generic Not Always Available

> showMethods("nzchar")

Error in genericForPrimitive(f) :
  methods may not be defined for primitive function "nzchar" in this version of R

> setMethod("nzchar", c("x" = "Greeting"),
+   function(x) nzchar(x@phrase))

Error in genericForPrimitive(f) :
  methods may not be defined for primitive function "nzchar" in this version of R
S4 Coerce and Replace Method Creation

setAs function arguments; produces coerce methods

▶ from – the old object’s class.
▶ to – the new object’s class.
▶ def – the method definition.

setReplaceMethod function; produces "<-" methods

▶ A wrapper for setMethod that adds suffix "<-" to argument f.
▶ e.g. "names<-", usage: names(x) <- letters.
Coercion Method for Canonical Example

Method definition

> setAs("Greeting", "character", + function(from) from@phrase)

Display coerce methods

> showMethods("coerce", classes = "Greeting")

Function: coerce (package methods)
from="Greeting", to="character"

Perform coercion

> as(new("Greeting"), "character")

[1] "Hello world!"
Slot Accessor and Replacer in Canonical Example (1/2)

Accessor generic and method

```r
> setGeneric("phrase", signature = "x",
+     function(x) standardGeneric("phrase"))
> setMethod("phrase", "Greeting",
+     function(x) x@phrase)
```

Replacer generic and method

```r
> setGeneric("phrase<-", signature = "x",
+     function(x, value) standardGeneric("phrase<-"))
> setReplaceMethod("phrase", "Greeting",
+     function(x, value) initialize(x, phrase = value))
```
Slot Accessor and Replacer in Canonical Example (2/2)

Accessor invocation

```r
> silentBob <- new("Greeting")
> phrase(silentBob)

[1] "Hello world!"
```

Replacer invocation

```r
> phrase(silentBob) <- ""
```

Error in validObject(.Object) :
  invalid class "Greeting" object: 'phrase' is empty
Motivation for Replacement Methods

Slot replacement methods don't validity check

```r
> silentBob <- new("Greeting")
> silentBob@phrase <- ""

> validObject(silentBob)
Error in validObject(silentBob) :
  invalid class "Greeting" object: 'phrase' is empty
```

initialize function does

```r
> initialize(silentBob, phrase = "")
Error in validObject(.Object) :
  invalid class "Greeting" object: 'phrase' is empty
```
Object Display in Canonical Example

**show method definition**

```r
> setMethod("show", "Greeting", 
+   function(object) 
+   { 
+     cat("A ", class(object), ": ", phrase(object), 
+         
+     } 
+   })
```

**show method invocation**

```r
> new("Greeting")
```

A Greeting: "Hello world!"
Finding Methods for an S4 Class

showMethods function wrapper

```r
> s4Methods <- function(class) 
+ { 
+   methods <- 
+     showMethods(classes = class, printTo = FALSE) 
+     methods <- methods[grep("^Function:", methods)] 
+     sapply(strsplit(methods, " "), ":[", 2) 
+ } 
```

Using custom function

```r
> s4Methods("Greeting")
```

```
[1] "coerce" "initialize" "nchar" "phrase"
[5] "phrase<-" "show" "words"
```
Outline

Introduction
  Object-Oriented Programming in R
  Role of S4 in Bioconductor

S4 Basics
  S4 Classes
  S4 Generic Functions & Methods

S4 Exercises

S4 Constraints
  Copy on Slot Modification
  Object Overhead
  Method Dispatch

S4 Case Studies
  Slot-Oriented Virtual Class (eSet)
  Method-Oriented Virtual Class (Sequence)
  Multiple Inheritance & Vectorization (CompressedIRangesList)
  Build or Reuse? (CompressedIRangesList)
  Class Union & Group Generic (Rle)

Resources
S4 Classes for Exercises

Virtual Class

> setClass("Sample",
+     representation("VIRTUAL",
+                     description = "character"))

Subclasses

> setClass("OneSample", contains = "Sample",
+     representation(one = "numeric"))

> setClass("TwoSample", contains = "Sample",
+     representation(one = "numeric",
+                     two = "numeric"))
S4 Exercises

1. Create a `PairedSample` class that inherits from `TwoSample`.
   (a) Same slots as `TwoSample`.
   (b) Validity method checks slots one and two to be of equal length.

2. Create a `tTest` generic and methods using `t.test` function from the stats package.
   (a) Generic signature
       
       (x, alternative = c("two.sided", "less", "greater"), mu = 0, conf.level = 0.95, ...)
   
   (b) `OneSample` and `PairedSample` method signature
       
       (x, alternative = c("two.sided", "less", "greater"), mu = 0, conf.level = 0.95)
   
   (c) `TwoSample` method signature
       
       (x, alternative = c("two.sided", "less", "greater"), mu = 0, conf.level = 0.95, var.equal = FALSE)

3. Create `show` methods and any other features you find interesting.
Outline

Introduction
Object-Oriented Programming in R
Role of S4 in Bioconductor

S4 Basics
S4 Classes
S4 Generic Functions & Methods

S4 Exercises

S4 Constraints
Copy on Slot Modification
Object Overhead
Method Dispatch

S4 Case Studies
Slot-Oriented Virtual Class (eSet)
Method-Oriented Virtual Class (Sequence)
Multiple Inheritance & Vectorization (CompressedIRangesList)
Build or Reuse? (CompressedIRangesList)
Class Union & Group Generic (Rle)

Resources
S4 Objects Always Copied on Modification

S4 object copy-on-change

```r
> setClass("DotDash", representation = 
+       representation(dot = "raw", dash = "integer"))
> aDotDash <- new("DotDash", dot = as.raw(1), dash = 1:1e8)
> system.time(aDotDash@dot <- as.raw(2))
    user  system elapsed
   0.254   0.333   0.603
```

Best practices

- Use one `initialize` call instead of multiple `"@<-"/"slot<-"` calls.
- Vectorize (e.g. inherit from IRanges’s `CompressedList` class).
S4 Object Overhead

S4 object instantiation timing & size

> setClass("DotDash", representation =
+ representation(dot = "raw", dash = "integer"))

> system.time(dd <- lapply(1:1e4, function(i) new("DotDash")))

user  system elapsed
0.740 0.003 0.745

> print(object.size(dd), units = "Mb")
4.2 Mb

Best practice

▶ Vectorize (e.g. inherit from IRanges’s CompressedList class).
S4 Method Dispatch

S4 method dispatch

```r
> setClass("DotDash", representation =
+     representation(dot = "raw", dash = "integer"))
> aDotDash <- new("DotDash")
> setMethod("show", "DotDash", function(object) cat())
> system.time(lapply(1:1e4, function(i) show(aDotDash)))
  user  system elapsed
  0.424  0.004  0.441
```

Best practice

```r
> chosenFun <- selectMethod("show", "DotDash")
> system.time(lapply(1:1e4, function(i) chosenFun(aDotDash)))
  user  system elapsed
  0.084  0.001  0.094
```
Outline

Introduction
  Object-Oriented Programming in R
  Role of S4 in Bioconductor

S4 Basics
  S4 Classes
  S4 Generic Functions & Methods

S4 Exercises

S4 Constraints
  Copy on Slot Modification
  Object Overhead
  Method Dispatch

S4 Case Studies
  Slot-Oriented Virtual Class (eSet)
  Method-Oriented Virtual Class (Sequence)
  Multiple Inheritance & Vectorization (CompressedIRangesList)
  Build or Reuse? (CompressedIRangesList)
  Class Union & Group Generic (Rle)

Resources
Slot-Oriented Virtual Class (eSet)

Biobase’s eSet

```r
> library(Biobase)
> isVirtualClass("eSet")
[1] TRUE
> getSlots("eSet")
  assayData  "AssayData"
  phenoData  "AnnotatedDataFrame"
  featureData  "AnnotatedDataFrame"
  experimentData  "MIAME"
  annotation  "character"
  protocolData  "AnnotatedDataFrame"
  .__classVersion__  "Versions"
```

limma’s MAList

```r
> library(limma)
> isVirtualClass("MAList")
[1] FALSE
> getSlots("MAList")
  .Data  "list"
```
Method-Oriented Virtual Class (Sequence)

IRanges's Sequence

```r
> library(IRanges)
> isVirtualClass("Sequence")
[1] TRUE
> getSlots("Sequence")

  elementMetadata  elementType  metadata
               "ANY"       "character"   "list"

> getSlots("Rle")

  values  lengths  elementMetadata
  "vectorORfactor"  "integer"   "ANY"

  elementType  metadata
  "character"   "list"

> length(s4Methods("Sequence"))
[1] 42
> head(s4Methods("Sequence"), 8)

[1] "!='"  "["  "[<-"]"  "["  "$"  "aggregate"  "append"  "as.env"
```
Method-Oriented Virtual Class (Sequence)

**Sequence subclass defines**

```
"[", c, length, window, seqselect, "seqselect<-"
```

**Sequence subclass inherits**

head, tail, append, subset, rep, rev (optimize?), "window<-", "[<-"

```
setMethod("head", "Sequence",
    function(x, n = 6L, ...)
    {
        stopifnot(length(n) == 1L)
        if (n < 0L)
        n <- max(length(x) + n, 0L)
        else
        n <- min(n, length(x))
        if (n == 0L)
        x[integer(0)]
        else
        window(x, 1L, n)
    })
```
Multiple Inheritance & Vectorization (CompressedIRangesList)

**CompressedIRangesList**

```
setClass("CompressedIRangesList",
    prototype = prototype(elementType = "IRanges",
                           unlistData = new("IRanges")),
    contains = c("IRangesList", "CompressedList"))
```

**IRanges’s List paradigm**

- *IRangesList* – a method-oriented virtual class
- *CompressedList* – a slot-oriented virtual class
- *SimpleIRangesList* – a sibling class to *CompressedIRangesList*
Build or Reuse? (CompressedIRangesList)

IRanges’s *CompressedIRangesList*

```r
> library(IRanges)
> is(new("CompressedIRangesList"))

[1] "CompressedIRangesList"
[2] "IRangesList"
[3] "CompressedList"
[4] "RangesList"
[5] "Sequence"
[6] "Annotated"

> length(s4Methods("RangesList"))
[1] 34

> head(s4Methods("RangesList"), 8)

[1] "as.data.frame"  "coerce"
[3] "countOverlaps"  "coverage"
[5] "disjoin"        "end"
[7] "end<-"         "findOverlaps"
```

Biostrings’s *MIndex*

```r
> library(Biostrings)
> isVirtualClass("MIndex")

[1] TRUE

> s4Methods("MIndex")

[1] "coerce"  "countIndex"
[3] "coverage" "length"
[5] "names"   "names<-"
[7] "unlist"  "width0"

> s4Methods("ByPos_MIndex")

[1] "[["  "endIndex"
[3] "show"  "startIndex"
```
Class Union & Group Generic (Rle)

IRanges’s *Rle*

```r
setClassUnion("vectorORfactor", c("vector", "factor"))

setClass("Rle", contains = "Sequence")
  representation(values = "vectorORfactor",
                lengths = "integer"),
  <<REMAINING DEFINITION>>)

setMethod("Summary", "Rle",
          function(x, ..., na.rm = FALSE)
          switch(.Generic,
                  all =, any =, min =, max =, range =
                  callGeneric(runValue(x), ..., na.rm = na.rm),
                  sum = sum(runValue(x) * runLength(x), ...,
                     na.rm = na.rm),
                  prod = prod(runValue(x) ^ runLength(x), ...,
                  na.rm = na.rm)))
```
Outline

Introduction
  Object-Oriented Programming in R
  Role of S4 in Bioconductor

S4 Basics
  S4 Classes
  S4 Generic Functions & Methods

S4 Exercises

S4 Constraints
  Copy on Slot Modification
  Object Overhead
  Method Dispatch

S4 Case Studies
  Slot-Oriented Virtual Class (eSet)
  Method-Oriented Virtual Class (Sequence)
  Multiple Inheritance & Vectorization (CompressedIRangesList)
  Build or Reuse? (CompressedIRangesList)
  Class Union & Group Generic (Rle)

Resources
Resources

Books


Documents on the Web


▶ “S4 Classes in 15 pages, more or less”, http://www.stat.auckland.ac.nz/S-Workshop/Gentleman/S4Objects.pdf