

S4 Case Studies

with an eye towards the GRanges and GRangesList classes

Patrick Aboyoun

Fred Hutchinson Cancer Research Center

28 July, 2010

Use of S4 in Bioconductor

S4 Case Studies

- Data Representations (Annotated Genomic Intervals)

- Slot-Oriented Virtual Class (eSet)

- Method-Oriented Virtual Class (Sequence)

- Multiple Inheritance & Vectorization (CompressedIRangesList)

- Build or Reuse? (CompressedIRangesList)

- Class Union & Group Generic (Rle)

Resources

Outline

Use of S4 in Bioconductor

S4 Case Studies

Data Representations (Annotated Genomic Intervals)

Slot-Oriented Virtual Class (eSet)

Method-Oriented Virtual Class (Sequence)

Multiple Inheritance & Vectorization (CompressedIRangesList)

Build or Reuse? (CompressedIRangesList)

Class Union & Group Generic (Rle)

Resources

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

Use of S4 in Bioconductor

S4 Case Studies

Data Representations (Annotated Genomic Intervals)

Slot-Oriented Virtual Class (eSet)

Method-Oriented Virtual Class (Sequence)

Multiple Inheritance & Vectorization (CompressedIRangesList)

Build or Reuse? (CompressedIRangesList)

Class Union & Group Generic (Rle)

Resources

General Feature Format

1. **seqname** - The name of the sequence. Must be a chromosome or scaffold.
2. **source** - The program that generated this feature.
3. **feature** - The name of this type of feature. Some examples of standard feature types are "CDS", "start_codon", "stop_codon", and "exon".
4. **start** - The starting position of the feature in the sequence. The first base is numbered 1.
5. **end** - The ending position of the feature (inclusive).
6. **score** - A score between 0 and 1000.
7. **strand** - Valid entries include '+', '-', or '.' (for don't know/don't care).
8. **frame** - If the feature is a coding exon, frame should be a number between 0-2 that represents the reading frame of the first base. If the feature is not a coding exon, the value should be '.'.
9. **group** - All lines with the same group are linked together into a single item.

Data Representations (Annotated Genomic Intervals)

GenomicRanges's *GRanges*

```
> library(GenomicRanges)
```

```
> getSlots("GRanges")
```

seqnames	ranges
"Rle"	"IRanges"
strand	seqlengths
"Rle"	"integer"
elementMetadata	elementType
"ANY"	"character"
metadata	
"list"	

Important considerations

1. Interval overlap timings
2. Coverage timings
3. Subset timings
4. Object size
5. Grouping (exons in tx)

genomeIntervals's *Genome_intervals_stranded*

```
> library(genomeIntervals)
```

```
> getSlots("Genome_intervals_stranded")
```

.Data	annotation	closed
"matrix"	"data.frame"	"matrix"
type		
"character"		

IRanges's *RangedData*

```
> library(IRanges)
```

```
> getSlots("RangedData")
```

ranges	values
"RangesList"	"SplitDataFrameList"
elementMetadata	elementType
"ANY"	"character"
metadata	
"list"	

Data Representations (Grouped Genomic Intervals)

GenomicRanges's *GRangesList*

```
> library(GenomicRanges)
> getSlots("GRangesList")
```

```
partitioning
"PartitioningByEnd"
unlistData
"ANY"
elementMetadata
"ANY"
elementType
"character"
metadata
"list"
```

```
> is(new("GRangesList"),
+    "CompressedList")
```

```
[1] TRUE
```

GenomicRanges's *GappedAlignments*

```
> getSlots("GappedAlignments")
```

seqnames	start
"Rle"	"integer"
cigar	strand
"character"	"raw"
seqlengths	elementMetadata
"integer"	"ANY"
elementType	metadata
"character"	"list"

Slot-Oriented Virtual Class (eSet)

Biobase's *eSet*

```
> 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*

```
> library(limma)
> isVirtualClass("MAList")
[1] FALSE
> getSlots("MAList")
      .Data
"list"
```

Finding Methods for an S4 Class

`showMethods` function wrapper

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

Method-Oriented Virtual Class (Sequence)

IRanges's *Sequence*

```
> 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] 45
```

```
> head(s4Methods("Sequence"), 8)
```

[1] "!="	"\$"	"Filter"	"Find"	"Map"
[6] "NROW"	"Position"	"Reduce"		

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*

```
> library(IRanges)

> is(new("CompressedIRangesList"))

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

> length(s4Methods("RangesList"))

[1] 39

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

[1] "%in%"           "Ops"
[3] "ScanBamParam"   "as.data.frame"
[5] "chrom"          "chrom<-"
[7] "coerce"         "countOverlaps"
```

Biostrings's *MIndex*

```
> library(Biostrings)

> isVirtualClass("MIndex")

[1] TRUE

> s4Methods("MIndex")

[1] "coerce"
[2] "countIndex"
[3] "coverage"
[4] "elementLengths"
[5] "length"
[6] "names"
[7] "names<-"
[8] "unlist"
[9] "width0"

> s4Methods("ByPos_MIndex")

[1] "[["           "endIndex"
[3] "show"         "startIndex"
```

Class Union & Group Generic (Rle)

IRanges's *Rle*

```
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

Use of S4 in Bioconductor

S4 Case Studies

- Data Representations (Annotated Genomic Intervals)
- 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

- ▶ John M. Chambers. *Software for Data Analysis: Programming with R*. Springer, New York, 2008. ISBN-13 978-0387759357.
- ▶ Robert Gentleman. *R Programming for Bioinformatics*. Chapman & Hall/CRC, New York, 2008. ISBN-13 978-1420063677.

Documents on the Web

- ▶ Patrick Aboyoun, “S4 System Development in Bioconductor”,
<http://www.bioconductor.org/help/course-materials/2010/AdvancedR/S4InBioconductor.pdf>
- ▶ John M. Chambers, “How S4 Methods Work”,
<http://developer.r-project.org/howMethodsWork.pdf>
- ▶ Friedrich Leisch, “S4 Classes and Methods”,
<http://www.ci.tuwien.ac.at/Conferences/user-2004/Keynotes/Leisch.pdf>
- ▶ “S4 Classes in 15 pages, more or less”,
<http://www.stat.auckland.ac.nz/S-Workshop/Gentleman/S4objects.pdf>