Ranges and Features in Bioconductor

The foundation layers of genomic arithmetic

Greg Tucker-Kellogg

September 30, 2022

Outline

Prequel: The pieces and layers of Bioconductor

Some concepts

Integer ranges of information (IRanges)

Genomic Ranges (GRanges)

GenomicFeatures

Prequel: The pieces and layers of Bioconductor

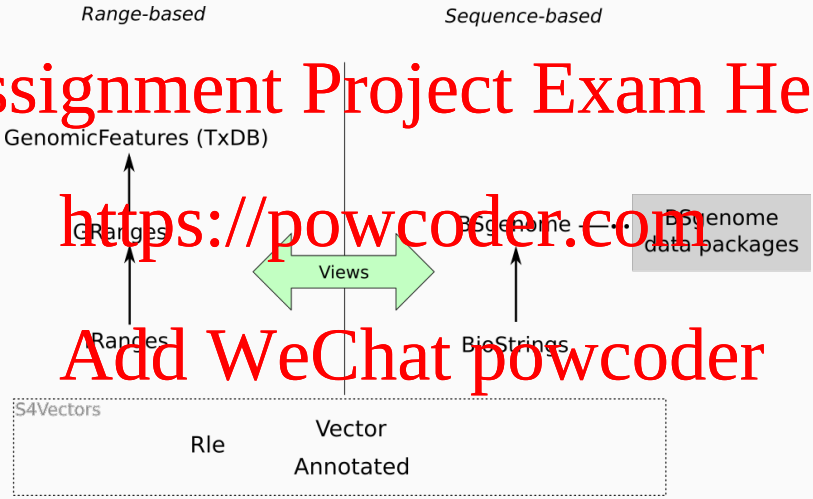
The elephant in the room

Why does Bioconductor reinvent so many wheels using S4?

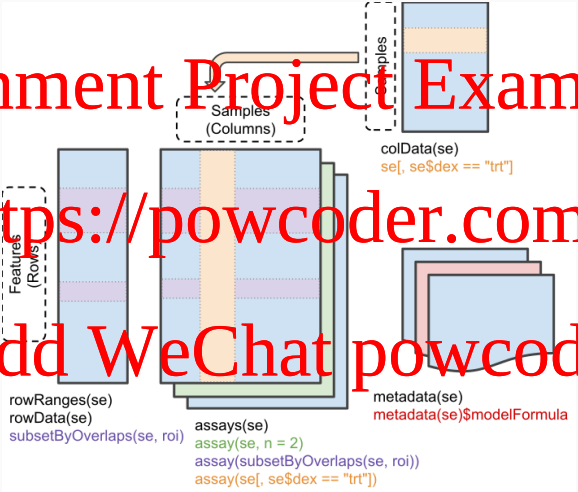
- Vectors
- Ranges
- and so on

Is this just so much yak shaving?

The core components of Bioconductor classes work together



Orchestration provides power (SummarizedExperiment)



Some concepts

Run Length Encoding (Rle)

Why

- Chromosomes can be large ($> 10^8$ base pairs), which make for big vectors
- Information on chromosomes tends to come in chunks, and many repeats
- We can use compression to make it easier to work with in memory.

How

- Run-length encoding (RLE) is a common compression technique for storing long sequences with lengthy repeats.

- Rle vectors of a sequence will represent repeats by their length, e.g., A8
- DNA only has four letters, so this is pretty efficient even for sequences

Annotation

Why

- Information on genomes includes lots of metadata, like gene names, peak heights, and so on
- We need to use this information efficiently

How

- Genomic information in Bioconductor can include *annotation*, which is a kind of data frame attached to the vector of ranges.

Views

Why

- Chromosomes can be large (> 10^8 base pairs), which means creating lots of subsets of sequences is *very* inefficient
- We need a solution

How

- Keep the original sequence, but only pass a reference to a location, not the value.
- Associate ranges with the sequence to select subsequences

Integer ranges of information (IRanges)

All genome browswer information sits on ranges

The IRanges class (from Bioconductor package IRanges)

```
ir <- IRanges(c(1, 8, 14, 15, 19, 34, 40),
              width=c(12, 6, 6, 15, 6, 2, 7))
ir
```



```
IRanges object with 7 ranges and 0 metadata columns:
```


	start	end	width
	<integer>	<integer>	<integer>
[1]	1	12	12
[2]	8	13	6
[3]	14	19	6
[4]	15	29	15
[5]	19	24	6
[6]	34	35	2
[7]	40	46	7

What is an IRanges object?

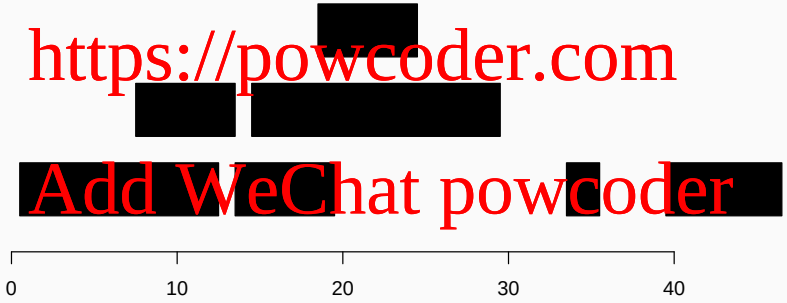
- A bit like a vector (A vector of integer ranges, built from the S4Vectors framework)
- A bit like a data.frame (you can add arbitrary metadata to it)
- Very memory-efficient
- Lots of associated methods that operate on IRanges

A convenience function (from the IRanges vignette)

```
plotRanges <- function(x, xlim=x, main=deparse(substitute(x)),
                       col="black", sep=0.5, ...)
{
  height <- 1
  if (is(xlim, "IntegerRanges"))
    xlim <- c(min(start(xlim)), max(end(xlim)))
  bins <- disjointBins(IRanges(start(x), end(x) + 1))
  plot.new()
  plot.window(xlim, c(0, max(bins)*(height + sep)))
  ybottom <- bins * (sep + height) - height
  rect(start(x)-0.5, ybottom, end(x)+0.5, ybottom + height, col=col, ...)
  title(main)
  axis(1)
}
```

Plotting some ranges

```
plotRanges(ir)
```

Normal ranges

- A *normal* IRange has no overlaps
- Any `IRange` can be normalised using `reduce(ir)`.
- A normal `IRange` is also a set of integers

- In addition to `reduce(ir)`, you may find `start(ir)`, `end(ir)` and `width(ir)` useful.
- These produce the corresponding integer vectors that define the IRange.
  Many other operations to modify ranges, such as `narrow`, `resize`, `flank`, `reflect`, `restrict`, and `threebands`

Finding overlaps between ranges

```
findOverlaps(ir,reduce(ir))

Hits object with 7 hits and 0 metadata columns:
      queryHits subjectHits
      <integer>   <integer>
  [1]         1           1
  [2]         2           1
  [3]         3           1
  [4]         4           1
  [5]         5           1
  [6]         6           2
  [7]         7           3
  -------
  queryLength: 7 / subjectLength: 3
```

Calculating “coverage”


```
coverage(ir)
coverage(reduce(ir))

integer-Rle of length 46 with 11 runs
  Lengths: 7 5 2 4 1 5 5 4 2 4 7
  Values : 1 2 1 2 3 2 1 0 1 0 1
integer-Rle of length 46 with 5 runs
  Lengths: 29  4  2  4  7
  Values :  1  0  1  0  1
```

Genomic Ranges (GRanges)

What are GRanges?

- Bioconductor package `GenomicRanges`.
- Builds upon `IRanges`.
- Includes `seqnames` (typically chromosomes), and `strand` information.
- Like `IRanges`, metadata can be added to GRanges objects

An example

```
  gr <- GRanges(
    seqnames = Rle(c("chr1", "chr2", "chr1", "chr3"), c(1, 3, 2, 4)),
    ranges = IRanges(101:110, end = 111:120, names = head(letters, 10)),
    strand = Rle(strand(c("-", "+", "*", "+", "-")), c(1, 2, 2, 3, 2)),
    score = 1:10,
    GC = seq(1, 0, length=10))
gr

GRanges object with 10 ranges and 2 metadata columns:
    seqnames    ranges strand |     score        GC
       <Rle> <IRanges>  <Rle> | <integer> <numeric>
  a     chr1   101-111      - |         1  1.000000
  b     chr2   102-112      + |         2  0.888889
  c     chr2   103-113      + |         3  0.777778
  d     chr2   104-114      * |         4  0.666667
  e     chr1   105-115      * |         5  0.555556
```

- The seqnames means that IRanges on different chromosomes never overlap
- The IRanges and rle encoding ensure memory efficiency and speed
- When seqnames correspond to BSgenome chromosome names, GRanges can be used to retrieve sequences efficiently
- Metadata allows building all the types of data that appear in Genome browsers

GenomicFeatures

The GenomicFeatures package

The GenomicFeatures package defines TxDb objects.

```
library(GenomicFeatures)
library(TxDb.Hsapiens.UCSC.hg38.knownGene)
txdb <- TxDb.Hsapiens.UCSC.hg38.knownGene
head(seqlevels(txdb),25)
```

```
 [1] "chr1"  "chr2"  "chr3"  "chr4"  "chr5"  "chr6"  "chr7"  "chr8"  "chr9"
[10] "chr10" "chr11" "chr12" "chr13" "chr14" "chr15" "chr16" "chr17" "chr18"
[19] "chr19" "chr20" "chr21" "chr22" "chrX"  "chrY"  "chrM"
```

What's in a TxDb object?


```
columns(txdb)

 [1] "CDSCHROM"   "CDSEND"     "CDSID"      "CDSNAME"    "CDSPHASE"
 [6] "CDSSTART"   "CDSSTRAND"  "EXONCHROM"  "EXONEND"    "EXONID"
[11] "EXONNAME"   "EXONRANK"   "EXONSTART"  "EXONSTRAND" "GENEID"
[16] "TXCHROM"    "TXEND"      "TXID"       "TXNAME"     "TXSTART"
[21] "TXSTRAND"   "TXTYPE"
```

Getting some ranges

```
genes(txdb)

  1662 genes were dropped because they have exons located on both strands
  of the same reference sequence or on more than one reference sequence,
  so cannot be represented by a single genomic range.
  Use 'single.strand.genes.only=FALSE' to get all the genes in a
  GRangesList object, or use suppressMessages() to suppress this message.
GRanges object with 29721 ranges and 1 metadata column:
           seqnames              ranges strand |     gene_id
              <Rle>           <IRanges>  <Rle> | <character>
         1    chr19   58345178-58362751      - |           1
        10     chr8   18386311-18401218      + |          10
       100    chr20   44619522-44652233      - |         100
      1000    chr18   27932879-28177946      - |        1000
 100008586     chrX   49551278-49568218      + |   100008586
       ...      ...                 ...    ... .         ...
```

Getting some ranges

```
transcripts(txdb)
```

```
GRanges object with 266064 ranges and 2 metadata columns:
                seqnames              ranges strand |        tx_id            tx_name
                   <Rle>           <IRanges>  <Rle> | <integer>        <character>
       [1]           chr1       11869-14409      + |         1 ENST00000456328.2
       [2]           chr1       11870-14660      - |         2 ENST00000450305.2
       [3]           chr1       29554-31097      + |         3 ENST00000473358.1
       [4]           chr1       30267-31109      + |         4 ENST00000469289.1
       [5]           chr1       30366-30503      + |         5 ENST00000607096.1
       ...            ...               ...    ... .       ...             ...
  [266060] chrUn_GL000220v1 155997-156149      + |    266060 ENST00000619779.1
  [266061] chrUn_KI270442v1 380608-380726      + |    266061 ENST00000620265.1
  [266062] chrUn_KI270442v1 217250-217401      - |    266062 ENST00000611690.1
  [266063] chrUn_KI270744v1   51009-51114      - |    266063 ENST00000616830.1
  [266064] chrUn_KI270750v1 148668-148843      + |    266064 ENST00000612925.1
```

Getting some ranges

```
exons(txdb)
```

```
GRanges object with 713360 ranges and 1 metadata column:
                seqnames          ranges strand |   exon_id
                   <Rle>       <IRanges>  <Rle> | <integer>
        [1]         chr1   11869-12227      + |         1
        [2]         chr1   12010-12057      - |         2
        [3]         chr1   12179-12227      + |         3
        [4]         chr1   12613-12697      + |         4
        [5]         chr1   12613-12721      + |         5
        ...          ...             ...    ...       ...
   [713356] chrUn_GL000220v1 155997-156149      + |    713356
   [713357] chrUn_KI270442v1 380608-380726      + |    713357
   [713358] chrUn_KI270442v1 217250-217401      - |    713358
   [713359] chrUn_KI270744v1   51009-51114      - |    713359
   [713360] chrUn_KI270750v1 148668-148843      + |    713360
```