

10 things (maybe) you didn't know about GenomicRanges, Biostrings, and Rsamtools

Hervé Pagès
hpages@fredhutch.org

June 2016

1. Inner vs outer metadata columns

```
> mcols(grl)$id <- paste0("ID", seq_along(grl))
```

```
> grl
```

GRangesList object of length 3:

\$gr1

GRanges object with 1 range and 2 metadata columns:

	seqnames	ranges	strand	score	GC
	<Rle>	<IRanges>	<Rle>	<integer>	<numeric>
[1]	Chrom2	3-6	+	5	0.45

seqinfo: 2 sequences from an unspecified genome; no seqlengths

\$gr2

GRanges object with 2 ranges and 2 metadata columns:

	seqnames	ranges	strand	score	GC
	<Rle>	<IRanges>	<Rle>	<integer>	<numeric>
[1]	Chrom1	7-9	+	3	0.3
[2]	Chrom1	13-15	-	4	0.5

seqinfo: 2 sequences from an unspecified genome; no seqlengths

\$gr3

GRanges object with 2 ranges and 2 metadata columns:

	seqnames	ranges	strand	score	GC
	<Rle>	<IRanges>	<Rle>	<integer>	<numeric>
[1]	Chrom1	1-3	-	6	0.4
[2]	Chrom2	4-9	-	2	0.1

1. Inner vs outer metadata columns

```
> mcols(gr1) # outer mcols
DataFrame with 3 rows and 1 column
      id
  <character>
gr1      ID1
gr2      ID2
gr3      ID3

> mcols(unlist(gr1, use.names=FALSE)) # inner mcols
DataFrame with 5 rows and 2 columns
      score      GC
  <integer> <numeric>
1         5      0.45
2         3      0.30
3         4      0.50
4         6      0.40
5         2      0.10
```

2. invertStrand()

Works out-of-the-box on any object that has a strand() getter and setter ==> no need to implement specific methods.

```
> gr
```

```
GRanges object with 10 ranges and 2 metadata columns:
```

	seqnames	ranges	strand	score	GC
	<Rle>	<IRanges>	<Rle>	<integer>	<numeric>
a	chr2	1-10	-	1	1.000000
b	chr2	2-10	+	2	0.888889
c	chr2	3-10	+	3	0.777778
.
h	chr3	8-10	+	8	0.222222
i	chr3	9-10	-	9	0.111111
j	chr3	10	-	10	0.000000

```
-----  
seqinfo: 3 sequences from an unspecified genome; no seqlengths
```

2. invertStrand()

```
> invertStrand(gr)
```

```
GRanges object with 10 ranges and 2 metadata columns:
```

	seqnames	ranges	strand	score	GC
	<Rle>	<IRanges>	<Rle>	<integer>	<numeric>
a	chr2	1-10	+	1	1.000000
b	chr2	2-10	-	2	0.888889
c	chr2	3-10	-	3	0.777778
.
h	chr3	8-10	-	8	0.222222
i	chr3	9-10	+	9	0.111111
j	chr3	10	+	10	0.000000

```
seqinfo: 3 sequences from an unspecified genome; no seqlengths
```

2. invertStrand()

```
> gr1
```

```
GRangesList object of length 3:
```

```
$gr1
```

```
GRanges object with 1 range and 2 metadata columns:
```

	seqnames	ranges	strand	score	GC
	<Rle>	<IRanges>	<Rle>	<integer>	<numeric>
[1]	Chrom2	3-6	+	5	0.45

```
-----  
seqinfo: 2 sequences from an unspecified genome; no seqlengths
```

```
$gr2
```

```
GRanges object with 2 ranges and 2 metadata columns:
```

	seqnames	ranges	strand	score	GC
	<Rle>	<IRanges>	<Rle>	<integer>	<numeric>
[1]	Chrom1	7-9	+	3	0.3
[2]	Chrom1	13-15	-	4	0.5

```
-----  
seqinfo: 2 sequences from an unspecified genome; no seqlengths
```

```
$gr3
```

```
GRanges object with 2 ranges and 2 metadata columns:
```

	seqnames	ranges	strand	score	GC
	<Rle>	<IRanges>	<Rle>	<integer>	<numeric>
[1]	Chrom1	1-3	-	6	0.4
[2]	Chrom2	4-9	-	2	0.1

2. invertStrand()

```
> invertStrand(gr1)
```

```
GRangesList object of length 3:
```

```
$gr1
```

```
GRanges object with 1 range and 2 metadata columns:
```

	seqnames	ranges	strand	score	GC
	<Rle>	<IRanges>	<Rle>	<integer>	<numeric>
[1]	Chrom2	3-6	-	5	0.45

```
-----  
seqinfo: 2 sequences from an unspecified genome; no seqlengths
```

```
$gr2
```

```
GRanges object with 2 ranges and 2 metadata columns:
```

	seqnames	ranges	strand	score	GC
	<Rle>	<IRanges>	<Rle>	<integer>	<numeric>
[1]	Chrom1	7-9	-	3	0.3
[2]	Chrom1	13-15	+	4	0.5

```
-----  
seqinfo: 2 sequences from an unspecified genome; no seqlengths
```

```
$gr3
```

```
GRanges object with 2 ranges and 2 metadata columns:
```

	seqnames	ranges	strand	score	GC
	<Rle>	<IRanges>	<Rle>	<integer>	<numeric>
[1]	Chrom1	1-3	+	6	0.4
[2]	Chrom2	4-9	+	2	0.1

3. extractList()

Extract groups of elements from a vector-like object and return them in a list-like object.

```
> cvg <- Rle(c(0L, 2L, 5L, 1L, 0L), c(10, 6, 3, 4, 15))
> cvg
integer-Rle of length 38 with 5 runs
  Lengths: 10  6  3  4 15
  Values  :  0  2  5  1  0
> i <- IRanges(c(16, 19, 9), width=5, names=letters[1:3])
> i
IRanges object with 3 ranges and 0 metadata columns:
      start      end      width
  <integer> <integer> <integer>
a         16         20         5
b         19         23         5
c          9         13         5
```


3. `extractList()`

```
> extractList(cvg, i)
RleList of length 3
$a
integer-Rle of length 5 with 3 runs
  Lengths: 1 3 1
  Values  : 2 5 1

$b
integer-Rle of length 5 with 2 runs
  Lengths: 1 4
  Values  : 5 1

$c
integer-Rle of length 5 with 2 runs
  Lengths: 2 3
  Values  : 0 2
```

3. extractList()

`i` can be an `IntegerList` object:

```
> i <- IntegerList(c(25:20), NULL, seq(from=2, to=length(cvg), by=2))  
> i
```

```
IntegerList of length 3
```

```
[[1]] 25 24 23 22 21 20
```

```
[[2]] integer(0)
```

```
[[3]] 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38
```

```
> extractList(cvg, i)
```

```
RleList of length 3
```

```
[[1]]
```

```
integer-Rle of length 6 with 2 runs
```

```
Lengths: 2 4
```

```
Values : 0 1
```

```
[[2]]
```

```
integer-Rle of length 0 with 0 runs
```

```
Lengths:
```

```
Values :
```

```
[[3]]
```

```
integer-Rle of length 19 with 5 runs
```

```
Lengths: 5 3 1 2 8
```

```
Values : 0 2 5 1 0
```

4. 'with.revmap' arg for reduce() and (now) disjoint()

```
> ir
```

IRanges object with 6 ranges and 2 metadata columns:

	start	end	width		id	score
	<integer>	<integer>	<integer>		<character>	<integer>
[1]	11	13	3		a	3
[2]	12	14	3		b	2
[3]	13	15	3		c	1
[4]	2	4	3		d	0
[5]	7	9	3		e	-1
[6]	6	8	3		f	-2

```
> ir2 <- reduce(ir, with.revmap=TRUE)
```

```
> ir2
```

IRanges object with 3 ranges and 1 metadata column:

	start	end	width		revmap
	<integer>	<integer>	<integer>		<IntegerList>
[1]	2	4	3		4
[2]	6	9	4		6,5
[3]	11	15	5		1,2,3

4. 'with.revmap' arg for reduce() and disjoin()

```
> revmap <- mcols(ir2)$revmap
> extractList(mcols(ir)$id, revmap)
CharacterList of length 3
[[1]] d
[[2]] f e
[[3]] a b c
> extractList(mcols(ir)$score, revmap)
IntegerList of length 3
[[1]] 0
[[2]] -2 -1
[[3]] 3 2 1
> mcols(ir2) <- DataFrame(id=extractList(mcols(ir)$id, revmap),
+                          score=extractList(mcols(ir)$score, revmap))
> ir2
```

IRanges object with 3 ranges and 2 metadata columns:

	start	end	width		id	score
	<integer>	<integer>	<integer>		<CharacterList>	<IntegerList>
[1]	2	4	3		d	0
[2]	6	9	4		f,e	-2,-1
[3]	11	15	5		a,b,c	3,2,1

5. Zero-width ranges

`findOverlaps/countOverlaps` support zero-width ranges.

```
> sliding_query <- IRanges(1:6, width=0)
> sliding_query
```

IRanges object with 6 ranges and 0 metadata columns:

	start	end	width
	<integer>	<integer>	<integer>
[1]	1	0	0
[2]	2	1	0
[3]	3	2	0
[4]	4	3	0
[5]	5	4	0
[6]	6	5	0

```
> countOverlaps(sliding_query, IRanges(3, 4))
[1] 0 0 0 1 0 0
```

But you have to specify `minoverlap=0` for this to work (default is 1).

```
> countOverlaps(sliding_query, IRanges(3, 4), minoverlap=0)
[1] 0 0 0 1 0 0
```

6. Biostrings::replaceAt()

Perform multiple substitutions at arbitrary positions in a set of sequences.

```
> library(Biostrings)
> library(hgu95av2probe)
> probes <- DNASTringSet(hgu95av2probe)
> probes
```

DNASTringSet object of length 201800:

	width	seq
[1]	25	TGGCTCCTGCTGAGGTCCCCTTTCC
[2]	25	GGCTGTGAATTCCTGTACATATTC
[3]	25	GCTTCAATTCCATTATGTTTTAATG
...
[201798]	25	TTCTGTCAAAGCATCATCTCAACAA
[201799]	25	CAAAGCATCATCTCAACAAGCCCTC
[201800]	25	GTGCTCCTTGTC AACAGCGCACCCA

6. Biostrings::replaceAt()

Replace 3rd and 4th nucleotides by pattern -++-.

```
> replaceAt(probes, at=IRanges(3, 4), value="-++-")
```

DNASTringSet object of length 201800:

```
      width seq
[1]      27 TG-++-TCCTGCTGAGGTCCCCTTTCC
[2]      27 GG-++-GTGAATTCCTGTACATATTTC
[3]      27 GC-++-CAATTCATTATGTTTTAATG
...      ...
[201798] 27 TT-++-GTCAAAGCATCATCTCAACAA
[201799] 27 CA-++-GCATCATCTCAACAAGCCCTC
[201800] 27 GT-++-TCCTTGTC AACAGCGCACCCA
```

6. Biostrings::replaceAt()

If supplied pattern is empty, then performs deletions.

```
> replaceAt(probes, at=IRanges(3, 4), value="")
```

DNASTringSet object of length 201800:

	width	seq
[1]	23	TGTCCTGCTGAGGTCCCCTTTCC
[2]	23	GGGTGAATTCCTGTACATATTTTC
[3]	23	GCCAATTCCATTATGTTTAAATG
...
[201798]	23	TTGTCAAAGCATCATCTCAACAA
[201799]	23	CAGCATCATCTCAACAAGCCCTC
[201800]	23	GTCCTTGTCAACAGCGCACCCA

6. Biostrings::replaceAt()

If `at` is a zero-width range, then performs insertions.

```
> replaceAt(probes, at=IRanges(4, 3), value="--+-")
```

DNASTringSet object of length 201800:

```
      width seq
[1]      29 TGG--+-CTCCTGCTGAGGTCCCCTTTCC
[2]      29 GGC--+-TGTGAATTCCTGTACATATTTC
[3]      29 GCT--+-TCAATTCATTATGTTTTAATG
...      ...
[201798]  29 TTC--+-TGTCAAAGCATCATCTCAACAA
[201799]  29 CAA--+-AGCATCATCTCAACAAGCCCTC
[201800]  29 GTG--+-CTCCTTGTC AACAGCGCACCCA
```

6. Biostrings::replaceAt()

Use it in combination with `vmatchPattern` to replace all the occurrences of a given pattern with another pattern:

```
> midx <- vmatchPattern("VCGTT", probes, fixed=FALSE)
> replaceAt(probes, at=midx, value="-++-")
```

DNASTringSet object of length 201800:

	width	seq
[1]	25	TGGCTCCTGCTGAGGTCCCCTTTCC
[2]	25	GGCTGTGAATTCCTGTACATATTTC
[3]	25	GCTTCAATTCCATTATGTTTTAATG
...
[201798]	25	TTCTGTCAAAGCATCATCTCAACAA
[201799]	25	CAAAGCATCATCTCAACAAGCCCTC
[201800]	25	GTGCTCCTTGTC AACAGCGCACCCA

7. GRanges as a subscript

```
> cvg <- RleList(chr1=101:120, chr2=2:-8, chr3=31:40)
> gr
```

GRanges object with 10 ranges and 2 metadata columns:

	seqnames	ranges	strand	score	GC
	<Rle>	<IRanges>	<Rle>	<integer>	<numeric>
a	chr2	1-10	-	1	1.000000
b	chr2	2-10	+	2	0.888889
c	chr2	3-10	+	3	0.777778
.
h	chr3	8-10	+	8	0.222222
i	chr3	9-10	-	9	0.111111
j	chr3	10	-	10	0.000000

seqinfo: 3 sequences from an unspecified genome; no seqlengths

7. GRanges as a subscript

```
> cvg[gr]
RleList of length 10
$chr2
integer-Rle of length 10 with 10 runs
  Lengths:  1  1  1  1  1  1  1  1  1  1
  Values :  2  1  0 -1 -2 -3 -4 -5 -6 -7

$chr2
integer-Rle of length 9 with 9 runs
  Lengths:  1  1  1  1  1  1  1  1  1
  Values :  1  0 -1 -2 -3 -4 -5 -6 -7

$chr2
integer-Rle of length 8 with 8 runs
  Lengths:  1  1  1  1  1  1  1  1
  Values :  0 -1 -2 -3 -4 -5 -6 -7

$chr2
integer-Rle of length 7 with 7 runs
  Lengths:  1  1  1  1  1  1  1
  Values : -1 -2 -3 -4 -5 -6 -7

$chr1
integer-Rle of length 6 with 6 runs
  Lengths:  1  1  1  1  1  1
  Values : 105 106 107 108 109 110

...
<5 more elements>
```

8. BSgenomeViews objects

```
> library(BSgenome.Mmusculus.UCSC.mm10)
> genome <- BSgenome.Mmusculus.UCSC.mm10
> library(TxDb.Mmusculus.UCSC.mm10.knownGene)
> txdb <- TxDb.Mmusculus.UCSC.mm10.knownGene
> ex <- exons(txdb, columns=c("exon_id", "tx_name", "gene_id"))
> v <- Views(genome, ex)
```

8. BSgenomeViews objects

```
> v
```

```
BSgenomeViews object with 447558 views and 3 metadata columns:
```

	seqnames	ranges	strand	dna
	<Rle>	<IRanges>	<Rle>	<DNAStrngSet>
[1]	chr1	3073253-3074322	+	[AAGGAAAGAG...TAGAGAAATG]
[2]	chr1	3102016-3102125	+	[GTGCTTGCTT...ACAAAAATAT]
[3]	chr1	3252757-3253236	+	[TTCTTCTGTG...TACCTTCAAT]
...
[447556]	chrUn_JH584304	58564-58835	-	[CTGTGGTCCT...CAGAGAAATG]
[447557]	chrUn_JH584304	58564-59690	-	[CTCTCTGCTG...CAGAGAAATG]
[447558]	chrUn_JH584304	59592-59667	-	[AGCTGTCCCG...GCCTTCTCAG]

	exon_id	tx_name	gene_id
	<integer>	<CharacterList>	<CharacterList>
[1]	1	ENSMUST00000193812.1	
[2]	2	ENSMUST00000082908.1	
[3]	3	ENSMUST00000192857.1	
...
[447556]	447556	ENSMUST00000179505.7	66776
[447557]	447557	ENSMUST00000178343.1	66776
[447558]	447558	ENSMUST00000179505.7	66776

```
-----  
seqinfo: 66 sequences (1 circular) from mm10 genome
```

8. BSgenomeViews objects

```
> af <- alphabetFrequency(v, baseOnly=TRUE)
> head(af)
```

	A	C	G	T	other
[1,]	376	160	206	328	0
[2,]	45	20	20	25	0
[3,]	138	105	86	151	0
[4,]	28	14	30	29	0
[5,]	57	39	20	33	0
[6,]	208	258	204	256	0

9. Pile-up statistics on a BAM file with Rsamtools::pileup()

```
> library(Rsamtools)
> library(RNAseqData.HNRNPC.bam.chr14)
> fl <- RNAseqData.HNRNPC.bam.chr14_BAMFILES[1]
> sbp <- ScanBamParam(which=GRanges("chr14", IRanges(1, 53674770)))
> pp <- PileupParam(distinguish_nucleotides=FALSE,
+                  distinguish_strands=FALSE,
+                  min_mapq=13,
+                  min_base_quality=10,
+                  min_nucleotide_depth=4)
> res <- pileup(fl, scanBamParam=sbp, pileupParam=pp)
```


9. Pile-up statistics on a BAM file with Rsamtools::pileup()

```
> dim(res)
```

```
[1] 248409      4
```

```
> head(res)
```

	seqnames	pos	count	which_label
1	chr14	19681651	4	chr14:1-53674770
2	chr14	19681655	4	chr14:1-53674770
3	chr14	19681657	4	chr14:1-53674770
4	chr14	19681658	4	chr14:1-53674770
5	chr14	19681661	4	chr14:1-53674770
6	chr14	19681662	4	chr14:1-53674770

10. Merging 2 GRanges objects (added this week)

```
> x
```

```
GRanges object with 2 ranges and 3 metadata columns:
```

	seqnames	ranges	strand	score	a1	a2
	<Rle>	<IRanges>	<Rle>	<numeric>	<integer>	<numeric>
[1]	chr1	1-1000	*	0.45	5	6
[2]	chr2	2000-3000	*	NA	7	8

```
-----  
seqinfo: 2 sequences from an unspecified genome; no seqlengths
```

```
> y
```

```
GRanges object with 3 ranges and 3 metadata columns:
```

	seqnames	ranges	strand	score	b1	b2
	<Rle>	<IRanges>	<Rle>	<numeric>	<integer>	<numeric>
[1]	chr2	150-151	*	0.70	0	1
[2]	chr1	1-10	*	0.82	5	-2
[3]	chr2	2000-3000	*	0.10	1	1

```
-----  
seqinfo: 2 sequences from an unspecified genome; no seqlengths
```

10. Merging 2 GRanges objects

```
> merge(x, y)
```

GRanges object with 1 range and 5 metadata columns:

	seqnames	ranges	strand	score	a1	a2	b1
	<Rle>	<IRanges>	<Rle>	<numeric>	<integer>	<numeric>	<integer>
[1]	chr2	2000-3000	*	0.1	7	8	1
		b2					
		<numeric>					
[1]		1					

seqinfo: 2 sequences from an unspecified genome; no seqlengths

10. Merging 2 GRanges objects

```
> merge(x, y, all=TRUE)
```

GRanges object with 4 ranges and 5 metadata columns:

	seqnames	ranges	strand	score	a1	a2	b1
	<Rle>	<IRanges>	<Rle>	<numeric>	<integer>	<numeric>	<integer>
[1]	chr1	1-10	*	0.82	<NA>	NA	5
[2]	chr1	1-1000	*	0.45	5	6	<NA>
[3]	chr2	150-151	*	0.70	<NA>	NA	0
[4]	chr2	2000-3000	*	0.10	7	8	1

b2

<numeric>

[1]	-2
[2]	NA
[3]	1
[4]	1

seqinfo: 2 sequences from an unspecified genome; no seqlengths