

A quick introduction to GRanges and GRangesList objects

Hervé Pagès
hpages@fredhutch.org

—

Michael Lawrence
lawrence.michael@gene.com

July 2015

GRanges objects

- The `GRanges()` constructor

- `GRanges` accessors

- Vector operations on `GRanges` objects

- Range-based operations on `GRanges` objects

GRangesList objects

- The `GRangesList()` constructor

- `GRangesList` accessors

- Vector operations on `GRangesList` objects

- List operations on `GRangesList` objects

- Range-based operations on `GRangesList` objects

Other resources

The GRanges class is a container for...

... storing a set of *genomic ranges* (a.k.a. *genomic regions* or *genomic intervals*).

- ▶ Each genomic range is described by a chromosome name, a *start*, an *end*, and a strand.
- ▶ *start* and *end* are both **1-based** positions relative to the 5' end of the plus strand of the chromosome, even when the range is on the minus strand.
- ▶ *start* and *end* are both considered to be included in the interval (except when the range is empty).
- ▶ The *width* of the range is the number of genomic positions included in it. So $width = end - start + 1$.
- ▶ *end* is always $\geq start$, except for empty ranges (a.k.a. zero-width ranges) where $end = start - 1$.

Note that the *start* is always the leftmost position and the *end* the rightmost, even when the range is on the minus strand.

Gotcha: A TSS is at the *end* of the range associated with a transcript located on the minus strand.

The `GRanges()` constructor

```
> library(GenomicRanges)
> gr1 <- GRanges(seqnames=Rle(c("ch1", "chMT"), c(2, 4)),
+               ranges=IRanges(16:21, 20),
+               strand=rep(c("+", "-", "*"), 2))
> gr1
```

GRanges object with 6 ranges and 0 metadata columns:

| | seqnames | ranges | strand |
|-----|----------|-----------|--------|
| | <Rle> | <IRanges> | <Rle> |
| [1] | ch1 | [16, 20] | + |
| [2] | ch1 | [17, 20] | - |
| [3] | chMT | [18, 20] | * |
| [4] | chMT | [19, 20] | + |
| [5] | chMT | [20, 20] | - |
| [6] | chMT | [21, 20] | * |

seqinfo: 2 sequences from an unspecified genome; no seqlengths

GRanges accessors: `length()`, `seqnames()`, `ranges()`

```
> length(gr1)
[1] 6

> seqnames(gr1)
factor-Rle of length 6 with 2 runs
  Lengths:    2    4
  Values  :  ch1 chMT
Levels(2): ch1 chMT

> ranges(gr1)
IRanges object with 6 ranges and 0 metadata columns:
      start      end      width
<integer> <integer> <integer>
[1]      16      20         5
[2]      17      20         4
[3]      18      20         3
[4]      19      20         2
[5]      20      20         1
[6]      21      20         0
```

GRanges accessors: `start()`, `end()`, `width()`, `strand()`

```
> start(gr1)
[1] 16 17 18 19 20 21
> end(gr1)
[1] 20 20 20 20 20 20
> width(gr1)
[1] 5 4 3 2 1 0
> strand(gr1)
factor-Rle of length 6 with 6 runs
  Lengths: 1 1 1 1 1 1
  Values  : + - * + - *
Levels(3): + - *
> strand(gr1) <- c("-", "-", "+")
> strand(gr1)
factor-Rle of length 6 with 4 runs
  Lengths: 2 1 2 1
  Values  : - + - +
Levels(3): + - *
```

GRanges accessors: `names()`

```
> names(gr1) <- LETTERS[1:6]
> gr1
```

GRanges object with 6 ranges and 0 metadata columns:

| | seqnames | ranges | strand |
|---|----------|-----------|--------|
| | <Rle> | <IRanges> | <Rle> |
| A | ch1 | [16, 20] | - |
| B | ch1 | [17, 20] | - |
| C | chMT | [18, 20] | + |
| D | chMT | [19, 20] | - |
| E | chMT | [20, 20] | - |
| F | chMT | [21, 20] | + |

seqinfo: 2 sequences from an unspecified genome; no seqlengths

```
> names(gr1)
[1] "A" "B" "C" "D" "E" "F"
```

GRanges accessors: `mcols()`

Like with most *Bioconductor* vector-like objects, *metadata columns* can be added to a GRanges object:

```
> mcols(gr1) <- DataFrame(score=11:16, GC=seq(1, 0, length=6))
> gr1
```

GRanges object with 6 ranges and 2 metadata columns:

| | seqnames | ranges | strand | score | GC |
|---|----------|-----------|--------|-----------|-----------|
| | <Rle> | <IRanges> | <Rle> | <integer> | <numeric> |
| A | ch1 | [16, 20] | - | 11 | 1 |
| B | ch1 | [17, 20] | - | 12 | 0.8 |
| C | chMT | [18, 20] | + | 13 | 0.6 |
| D | chMT | [19, 20] | - | 14 | 0.4 |
| E | chMT | [20, 20] | - | 15 | 0.2 |
| F | chMT | [21, 20] | + | 16 | 0 |

seqinfo: 2 sequences from an unspecified genome; no seqlengths

```
> mcols(gr1)
```

DataFrame with 6 rows and 2 columns

| | score | GC |
|---|-----------|-----------|
| | <integer> | <numeric> |
| 1 | 11 | 1.0 |
| 2 | 12 | 0.8 |
| 3 | 13 | 0.6 |
| 4 | 14 | 0.4 |
| 5 | 15 | 0.2 |
| 6 | 16 | 0.0 |

GRanges accessors: `seqinfo()`, `seqlevels()`, `seqlengths()`

```
> seqinfo(gr1)
```

Seqinfo object with 2 sequences from an unspecified genome; no seqlengths:

| seqnames | seqlengths | isCircular | genome |
|----------|------------|------------|--------|
| ch1 | NA | NA | <NA> |
| chMT | NA | NA | <NA> |

```
> seqlevels(gr1)
```

```
[1] "ch1" "chMT"
```

```
> seqlengths(gr1)
```

| ch1 | chMT |
|-----|------|
| NA | NA |

```
> seqlengths(gr1) <- c(50000, 800)
```

```
> seqlengths(gr1)
```

| ch1 | chMT |
|-------|------|
| 50000 | 800 |

Vector operations on GRanges objects

What we call *vector operations* are operations that work on any ordinary vector:

- ▶ `length()`, `names()`
- ▶ Single-bracket subsetting: `[`
- ▶ Combining: `c()`
- ▶ `split()`, `relist()`
- ▶ Comparing: `==`, `!=`, `match()`, `%in%`, `duplicated()`, `unique()`
- ▶ Ordering: `<=`, `>=`, `<`, `>`, `order()`, `sort()`, `rank()`

GRanges objects support all these *vector operations* ==> They're considered *vector-like* objects.

Vector operations on GRanges objects: Single-bracket subsetting

```
> gr1[c("F", "A")]
```

GRanges object with 2 ranges and 2 metadata columns:

| seqnames | ranges | strand | score | GC |
|----------|---------------|--------|-----------|-----------|
| <Rle> | <IRanges> | <Rle> | <integer> | <numeric> |
| F | chMT [21, 20] | + | 16 | 0 |
| A | ch1 [16, 20] | - | 11 | 1 |

seqinfo: 2 sequences from an unspecified genome

```
> gr1[strand(gr1) == "+"]
```

GRanges object with 2 ranges and 2 metadata columns:

| seqnames | ranges | strand | score | GC |
|----------|---------------|--------|-----------|-----------|
| <Rle> | <IRanges> | <Rle> | <integer> | <numeric> |
| C | chMT [18, 20] | + | 13 | 0.6 |
| F | chMT [21, 20] | + | 16 | 0 |

seqinfo: 2 sequences from an unspecified genome

Vector operations on GRanges objects: Single-bracket subsetting

```
> gr1 <- gr1[-5]
```

```
> gr1
```

GRanges object with 5 ranges and 2 metadata columns:

| | seqnames | ranges | strand | score | GC |
|---|----------|-----------|--------|-----------|-----------|
| | <Rle> | <IRanges> | <Rle> | <integer> | <numeric> |
| A | ch1 | [16, 20] | - | 11 | 1 |
| B | ch1 | [17, 20] | - | 12 | 0.8 |
| C | chMT | [18, 20] | + | 13 | 0.6 |
| D | chMT | [19, 20] | - | 14 | 0.4 |
| F | chMT | [21, 20] | + | 16 | 0 |

seqinfo: 2 sequences from an unspecified genome

Vector operations on GRanges objects: Combining

```
> gr2 <- GRanges(seqnames="ch2",
+               ranges=IRanges(start=c(2:1,2), width=6),
+               score=15:13,
+               GC=seq(0, 0.4, length=3))
> gr12 <- c(gr1, gr2)
> gr12
```

GRanges object with 8 ranges and 2 metadata columns:

| | seqnames | ranges | strand | score | GC |
|---|----------|-----------|--------|-----------|-----------|
| | <Rle> | <IRanges> | <Rle> | <integer> | <numeric> |
| A | ch1 | [16, 20] | - | 11 | 1 |
| B | ch1 | [17, 20] | - | 12 | 0.8 |
| C | chMT | [18, 20] | + | 13 | 0.6 |
| . | ... | ... | ... | ... | ... |
| | ch2 | [2, 7] | * | 15 | 0 |
| | ch2 | [1, 6] | * | 14 | 0.2 |
| | ch2 | [2, 7] | * | 13 | 0.4 |

seqinfo: 3 sequences from an unspecified genome

Vector operations on GRanges objects: Comparing

```
> gr12[length(gr12)] == gr12
[1] FALSE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE
> duplicated(gr12)
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
> unique(gr12)
```

GRanges object with 7 ranges and 2 metadata columns:

| | seqnames | ranges | strand | score | GC |
|---|----------|-----------|--------|-----------|-----------|
| | <Rle> | <IRanges> | <Rle> | <integer> | <numeric> |
| A | ch1 | [16, 20] | - | 11 | 1 |
| B | ch1 | [17, 20] | - | 12 | 0.8 |
| C | chMT | [18, 20] | + | 13 | 0.6 |
| D | chMT | [19, 20] | - | 14 | 0.4 |
| F | chMT | [21, 20] | + | 16 | 0 |
| | ch2 | [2, 7] | * | 15 | 0 |
| | ch2 | [1, 6] | * | 14 | 0.2 |

seqinfo: 3 sequences from an unspecified genome

Vector operations on GRanges objects: Ordering

```
> sort(gr12)
```

GRanges object with 8 ranges and 2 metadata columns:

| | seqnames | ranges | strand | score | GC |
|---|----------|-----------|--------|-----------|-----------|
| | <Rle> | <IRanges> | <Rle> | <integer> | <numeric> |
| A | ch1 | [16, 20] | - | 11 | 1 |
| B | ch1 | [17, 20] | - | 12 | 0.8 |
| C | chMT | [18, 20] | + | 13 | 0.6 |
| . | ... | ... | ... | ... | ... |
| | ch2 | [1, 6] | * | 14 | 0.2 |
| | ch2 | [2, 7] | * | 15 | 0 |
| | ch2 | [2, 7] | * | 13 | 0.4 |

seqinfo: 3 sequences from an unspecified genome

Splitting a GRanges object

```
> split(gr12, seqnames(gr12))
```

GRangesList object of length 3:

\$ch1

GRanges object with 2 ranges and 2 metadata columns:

| | seqnames | ranges | strand | score | GC |
|---|----------|-----------|--------|-----------|-----------|
| | <Rle> | <IRanges> | <Rle> | <integer> | <numeric> |
| A | ch1 | [16, 20] | - | 11 | 1 |
| B | ch1 | [17, 20] | - | 12 | 0.8 |

\$chMT

GRanges object with 3 ranges and 2 metadata columns:

| | seqnames | ranges | strand | score | GC |
|---|----------|----------|--------|-------|-----|
| C | chMT | [18, 20] | + | 13 | 0.6 |
| D | chMT | [19, 20] | - | 14 | 0.4 |
| F | chMT | [21, 20] | + | 16 | 0 |

\$ch2

GRanges object with 3 ranges and 2 metadata columns:

| | seqnames | ranges | strand | score | GC |
|--|----------|--------|--------|-------|-----|
| | ch2 | [2, 7] | * | 15 | 0 |
| | ch2 | [1, 6] | * | 14 | 0.2 |
| | ch2 | [2, 7] | * | 13 | 0.4 |

seqinfo: 3 sequences from an unspecified genome

Exercise 1

- a. Load the *GenomicRanges* package.
- b. Open the man page for the `GRanges` class and run the examples in it.
- c. Extract from `GRanges` object `gr` the elements (i.e. ranges) with a score between 4 and 8.
- d. Split `gr` by strand.

An overview of *range-based* operations

Intra range transformations

`shift()`, `narrow()`, `resize()`, `flank()`

Inter range transformations

`range()`, `reduce()`, `gaps()`, `disjoin()`

Range-based set operations

`union()`, `intersect()`, `setdiff()`,
`punion()`, `pintersect()`, `psetdiff()`,
`pgap()`

Coverage and slicing

`coverage()`, `slice()`

Finding/counting overlapping ranges

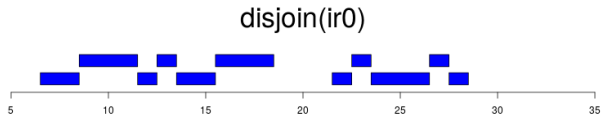
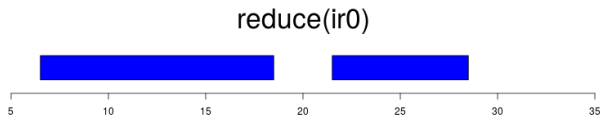
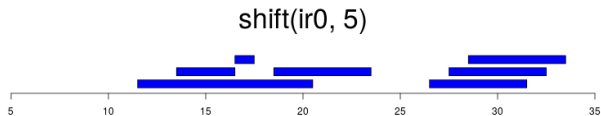
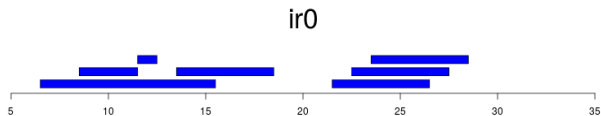
`findOverlaps()`, `countOverlaps()`

Finding the nearest range neighbor

`nearest()`, `precede()`, `follow()`

and more...

Examples of some common *range-based* operations



Range-based operations on GRanges objects

```
> gr2
```

GRanges object with 3 ranges and 2 metadata columns:

| | seqnames | ranges | strand | score | GC |
|-----|----------|-----------|--------|-----------|-----------|
| | <Rle> | <IRanges> | <Rle> | <integer> | <numeric> |
| [1] | ch2 | [2, 7] | * | 15 | 0 |
| [2] | ch2 | [1, 6] | * | 14 | 0.2 |
| [3] | ch2 | [2, 7] | * | 13 | 0.4 |

seqinfo: 1 sequence from an unspecified genome; no seqlengths

```
> shift(gr2, 50)
```

GRanges object with 3 ranges and 2 metadata columns:

| | seqnames | ranges | strand | score | GC |
|-----|----------|-----------|--------|-----------|-----------|
| | <Rle> | <IRanges> | <Rle> | <integer> | <numeric> |
| [1] | ch2 | [52, 57] | * | 15 | 0 |
| [2] | ch2 | [51, 56] | * | 14 | 0.2 |
| [3] | ch2 | [52, 57] | * | 13 | 0.4 |

seqinfo: 1 sequence from an unspecified genome; no seqlengths

Range-based operations on GRanges objects (continued)

```
> gr1
```

GRanges object with 5 ranges and 2 metadata columns:

| | seqnames | ranges | strand | score | GC |
|---|----------|-----------|--------|-----------|-----------|
| | <Rle> | <IRanges> | <Rle> | <integer> | <numeric> |
| A | ch1 | [16, 20] | - | 11 | 1 |
| B | ch1 | [17, 20] | - | 12 | 0.8 |
| C | chMT | [18, 20] | + | 13 | 0.6 |
| D | chMT | [19, 20] | - | 14 | 0.4 |
| F | chMT | [21, 20] | + | 16 | 0 |

seqinfo: 2 sequences from an unspecified genome

```
> resize(gr1, 12)
```

GRanges object with 5 ranges and 2 metadata columns:

| | seqnames | ranges | strand | score | GC |
|---|----------|-----------|--------|-----------|-----------|
| | <Rle> | <IRanges> | <Rle> | <integer> | <numeric> |
| A | ch1 | [9, 20] | - | 11 | 1 |
| B | ch1 | [9, 20] | - | 12 | 0.8 |
| C | chMT | [18, 29] | + | 13 | 0.6 |
| D | chMT | [9, 20] | - | 14 | 0.4 |
| F | chMT | [21, 32] | + | 16 | 0 |

seqinfo: 2 sequences from an unspecified genome

Range-based operations on GRanges objects (continued)

```
> gr1
```

GRanges object with 5 ranges and 2 metadata columns:

| | seqnames | ranges | strand | score | GC |
|---|----------|-----------|--------|-----------|-----------|
| | <Rle> | <IRanges> | <Rle> | <integer> | <numeric> |
| A | ch1 | [16, 20] | - | 11 | 1 |
| B | ch1 | [17, 20] | - | 12 | 0.8 |
| C | chMT | [18, 20] | + | 13 | 0.6 |
| D | chMT | [19, 20] | - | 14 | 0.4 |
| F | chMT | [21, 20] | + | 16 | 0 |

seqinfo: 2 sequences from an unspecified genome

```
> flank(gr1, 3)
```

GRanges object with 5 ranges and 2 metadata columns:

| | seqnames | ranges | strand | score | GC |
|---|----------|-----------|--------|-----------|-----------|
| | <Rle> | <IRanges> | <Rle> | <integer> | <numeric> |
| A | ch1 | [21, 23] | - | 11 | 1 |
| B | ch1 | [21, 23] | - | 12 | 0.8 |
| C | chMT | [15, 17] | + | 13 | 0.6 |
| D | chMT | [21, 23] | - | 14 | 0.4 |
| F | chMT | [18, 20] | + | 16 | 0 |

seqinfo: 2 sequences from an unspecified genome

Range-based operations on GRanges objects (continued)

```
> gr3 <- shift(gr1, c(35000, rep(0, 3), 100))
> width(gr3)[c(3,5)] <- 117
> gr3
```

GRanges object with 5 ranges and 2 metadata columns:

| | seqnames | ranges | strand | score | GC |
|---|----------|----------------|--------|-----------|-----------|
| | <Rle> | <IRanges> | <Rle> | <integer> | <numeric> |
| A | ch1 | [35016, 35020] | - | 11 | 1 |
| B | ch1 | [17, 20] | - | 12 | 0.8 |
| C | chMT | [18, 134] | + | 13 | 0.6 |
| D | chMT | [19, 20] | - | 14 | 0.4 |
| F | chMT | [121, 237] | + | 16 | 0 |

seqinfo: 2 sequences from an unspecified genome

```
> range(gr3)
```

GRanges object with 3 ranges and 0 metadata columns:

| | seqnames | ranges | strand |
|-----|----------|-------------|--------|
| | <Rle> | <IRanges> | <Rle> |
| [1] | ch1 | [17, 35020] | - |
| [2] | chMT | [18, 237] | + |
| [3] | chMT | [19, 20] | - |

seqinfo: 2 sequences from an unspecified genome

Range-based operations on GRanges objects (continued)

```
> gr3
```

```
GRanges object with 5 ranges and 2 metadata columns:
```

| | seqnames | ranges | strand | score | GC |
|---|----------|----------------|--------|-----------|-----------|
| | <Rle> | <IRanges> | <Rle> | <integer> | <numeric> |
| A | ch1 | [35016, 35020] | - | 11 | 1 |
| B | ch1 | [17, 20] | - | 12 | 0.8 |
| C | chMT | [18, 134] | + | 13 | 0.6 |
| D | chMT | [19, 20] | - | 14 | 0.4 |
| F | chMT | [121, 237] | + | 16 | 0 |

```
-----  
seqinfo: 2 sequences from an unspecified genome
```

```
> reduce(gr3)
```

```
GRanges object with 4 ranges and 0 metadata columns:
```

| | seqnames | ranges | strand |
|-----|----------|----------------|--------|
| | <Rle> | <IRanges> | <Rle> |
| [1] | ch1 | [17, 20] | - |
| [2] | ch1 | [35016, 35020] | - |
| [3] | chMT | [18, 237] | + |
| [4] | chMT | [19, 20] | - |

```
-----  
seqinfo: 2 sequences from an unspecified genome
```


Range-based operations on GRanges objects (continued)

```
> gr3
```

```
GRanges object with 5 ranges and 2 metadata columns:
```

| | seqnames | ranges | strand | score | GC |
|---|----------|----------------|--------|-----------|-----------|
| | <Rle> | <IRanges> | <Rle> | <integer> | <numeric> |
| A | chr1 | [35016, 35020] | - | 11 | 1 |
| B | chr1 | [17, 20] | - | 12 | 0.8 |
| C | chrMT | [18, 134] | + | 13 | 0.6 |
| D | chrMT | [19, 20] | - | 14 | 0.4 |
| F | chrMT | [121, 237] | + | 16 | 0 |

```
-----
```

```
seqinfo: 2 sequences from an unspecified genome
```

```
> gaps(gr3)
```

```
GRanges object with 10 ranges and 0 metadata columns:
```

| | seqnames | ranges | strand |
|------|----------|-------------|--------|
| | <Rle> | <IRanges> | <Rle> |
| [1] | chr1 | [1, 50000] | + |
| [2] | chr1 | [1, 16] | - |
| [3] | chr1 | [21, 35015] | - |
| ... | ... | ... | ... |
| [8] | chrMT | [1, 18] | - |
| [9] | chrMT | [21, 800] | - |
| [10] | chrMT | [1, 800] | * |

```
-----
```

```
seqinfo: 2 sequences from an unspecified genome
```

Range-based operations on GRanges objects (continued)

```
> gr3
```

GRanges object with 5 ranges and 2 metadata columns:

| | seqnames | ranges | strand | score | GC |
|---|----------|----------------|--------|-----------|-----------|
| | <Rle> | <IRanges> | <Rle> | <integer> | <numeric> |
| A | chr1 | [35016, 35020] | - | 11 | 1 |
| B | chr1 | [17, 20] | - | 12 | 0.8 |
| C | chrMT | [18, 134] | + | 13 | 0.6 |
| D | chrMT | [19, 20] | - | 14 | 0.4 |
| F | chrMT | [121, 237] | + | 16 | 0 |

seqinfo: 2 sequences from an unspecified genome

```
> disjoint(gr3)
```

GRanges object with 6 ranges and 0 metadata columns:

| | seqnames | ranges | strand |
|-----|----------|----------------|--------|
| | <Rle> | <IRanges> | <Rle> |
| [1] | chr1 | [17, 20] | - |
| [2] | chr1 | [35016, 35020] | - |
| [3] | chrMT | [18, 120] | + |
| [4] | chrMT | [121, 134] | + |
| [5] | chrMT | [135, 237] | + |
| [6] | chrMT | [19, 20] | - |

seqinfo: 2 sequences from an unspecified genome

Exercise 2

Using `GRanges` object `gr` created at Exercise 1:

- a. Shift the ranges in `gr` by 1000 positions to the right.
- b. What method is called when doing `shift()` on a `GRanges` object? Find the man page for this method.

Coverage

```
> cvg12 <- coverage(gr12)
> cvg12

RleList of length 3
$ch1
integer-Rle of length 50000 with 4 runs
  Lengths:    15     1     4 49980
  Values  :     0     1     2     0

$chMT
integer-Rle of length 800 with 4 runs
  Lengths:   17     1     2  780
  Values  :    0     1     2     0

$ch2
integer-Rle of length 7 with 3 runs
  Lengths:  1 5 1
  Values  :  1 3 2
```

Coverage (continued)

```
> mean(cvg12)
      ch1      chMT      ch2
0.000180 0.006250 2.571429
```

```
> max(cvg12)
ch1 chMT ch2
  2   2   3
```

Slicing the coverage

```
> sl12 <- slice(cvg12, lower=1)
> sl12

RleViewsList of length 3
names(3): ch1 chMT ch2

> elementNROWS(sl12)

  ch1 chMT  ch2
    1    1    1

> sl12$chMT

Views on a 800-length Rle subject

views:
      start end width
[1]    18  20     3 [1 2 2]

> mean(sl12$chMT)

[1] 1.666667

> max(sl12$chMT)

[1] 2
```

findOverlaps()

Load aligned reads from a BAM file:

```
> library(pasillaBamSubset)
> untreated1_chr4()

[1] "/home/biocbuild/bbs-3.3-bioc/R/library/pasillaBamSubset/extdata/untreated1_chr4.bam"

> library(GenomicAlignments)
> reads <- readGAlignments(untreated1_chr4())
```

and store them in a GRanges object:

```
> reads <- as(reads, "GRanges")
> reads[1:4]
```

GRanges object with 4 ranges and 0 metadata columns:

| | seqnames | ranges | strand |
|-----|----------|-------------|--------|
| | <Rle> | <IRanges> | <Rle> |
| [1] | chr4 | [892, 966] | - |
| [2] | chr4 | [919, 993] | - |
| [3] | chr4 | [924, 998] | + |
| [4] | chr4 | [936, 1010] | + |

seqinfo: 8 sequences from an unspecified genome

findOverlaps() (continued)

Load the gene ranges from a *TxDb* package:

```
> library(TxDb.Dmelanogaster.UCSC.dm3.ensGene)
> txdb <- TxDb.Dmelanogaster.UCSC.dm3.ensGene
> dm3_genes <- genes(txdb)
```

and find the overlaps between the reads and the genes:

```
> hits <- findOverlaps(reads, dm3_genes)
> head(hits)
```

Hits object with 6 hits and 0 metadata columns:

| | queryHits | subjectHits |
|-----|-----------|-------------|
| | <integer> | <integer> |
| [1] | 6296 | 11499 |
| [2] | 6304 | 11499 |
| [3] | 6305 | 11499 |
| [4] | 6310 | 11499 |
| [5] | 6311 | 11499 |
| [6] | 6312 | 11499 |

queryLength: 204355 / subjectLength: 15682

Exercise 3

- a. Recreate `GRanges` objects `reads` and `dm3_genes` from previous slides.
- b. What method is called when calling `findOverlaps()` on them? Open the man page for this method.
- c. Find the overlaps between the 2 objects but this time the strand should be ignored.

Exercise 4

In this exercise we want to get the exon sequences for the dm3 genome.

- a. Extract the exon ranges from `txdb`.
- b. Load the *BSgenome.Dmelanogaster.UCSC.dm3* package.
- c. Use `getSeq()` to extract the exon sequences from the `BSgenome` object in *BSgenome.Dmelanogaster.UCSC.dm3*.

The GRangesList class is a container for...

storing a list of *compatible* GRanges objects.

compatible means:

- ▶ they are relative to the same genome,
- ▶ AND they have the same metadata columns (accessible with the `mcols()` accessor).

The GRangesList() constructor

```
> grl <- GRangesList(gr3, gr2)
> grl
```

GRangesList object of length 2:

[[1]]

GRanges object with 5 ranges and 2 metadata columns:

| | seqnames | ranges | strand | score | GC |
|---|----------|----------------|--------|-----------|-----------|
| | <Rle> | <IRanges> | <Rle> | <integer> | <numeric> |
| A | ch1 | [35016, 35020] | - | 11 | 1 |
| B | ch1 | [17, 20] | - | 12 | 0.8 |
| C | chMT | [18, 134] | + | 13 | 0.6 |
| D | chMT | [19, 20] | - | 14 | 0.4 |
| F | chMT | [121, 237] | + | 16 | 0 |

[[2]]

GRanges object with 3 ranges and 2 metadata columns:

| | seqnames | ranges | strand | score | GC |
|--|----------|--------|--------|-------|-----|
| | ch2 | [2, 7] | * | 15 | 0 |
| | ch2 | [1, 6] | * | 14 | 0.2 |
| | ch2 | [2, 7] | * | 13 | 0.4 |

seqinfo: 3 sequences from an unspecified genome

GRangesList accessors

```
> length(grl)
```

```
[1] 2
```

```
> seqnames(grl)
```

```
RleList of length 2
```

```
[[1]]
```

```
factor-Rle of length 5 with 2 runs
```

```
  Lengths:    2    3
```

```
  Values :  ch1 chMT
```

```
Levels(3): ch1 chMT ch2
```

```
[[2]]
```

```
factor-Rle of length 3 with 1 run
```

```
  Lengths:    3
```

```
  Values :  ch2
```

```
Levels(3): ch1 chMT ch2
```

```
> strand(grl)
```

```
RleList of length 2
```

```
[[1]]
```

```
factor-Rle of length 5 with 4 runs
```

```
  Lengths: 2 1 1 1
```

```
  Values :  - + - +
```

```
Levels(3): + - *
```

```
[[2]]
```

```
factor-Rle of length 3 with 1 run
```

```
  Lengths:    3
```

```
  Values :  *
```

```
Levels(3): + - *
```

GRangesList accessors (continued)

```
> ranges(grl)
IRangesList of length 2
[[1]]
IRanges object with 5 ranges and 0 metadata columns
      start      end      width
  <integer> <integer> <integer>
A      35016      35020         5
B         17         20         4
C         18        134       117
D         19         20         2
F        121        237       117

[[2]]
IRanges object with 3 ranges and 0 metadata columns
      start      end      width
  <integer> <integer> <integer>
          2         7         6
          1         6         6
          2         7         6
```

```
> start(grl)
IntegerList of length 2
[[1]] 35016 17 18 19 121
[[2]] 2 1 2

> end(grl)
IntegerList of length 2
[[1]] 35020 20 134 20 237
[[2]] 7 6 7

> width(grl)
IntegerList of length 2
[[1]] 5 4 117 2 117
[[2]] 6 6 6
```

GRangesList accessors (continued)

```
> names(grl) <- c("TX1", "TX2")
> grl
```

GRangesList object of length 2:

\$TX1

GRanges object with 5 ranges and 2 metadata columns:

| | seqnames | ranges | strand | score | GC |
|---|----------|----------------|--------|-----------|-----------|
| | <Rle> | <IRanges> | <Rle> | <integer> | <numeric> |
| A | ch1 | [35016, 35020] | - | 11 | 1 |
| B | ch1 | [17, 20] | - | 12 | 0.8 |
| C | chMT | [18, 134] | + | 13 | 0.6 |
| D | chMT | [19, 20] | - | 14 | 0.4 |
| F | chMT | [121, 237] | + | 16 | 0 |

\$TX2

GRanges object with 3 ranges and 2 metadata columns:

| | seqnames | ranges | strand | score | GC |
|--|----------|--------|--------|-------|-----|
| | ch2 | [2, 7] | * | 15 | 0 |
| | ch2 | [1, 6] | * | 14 | 0.2 |
| | ch2 | [2, 7] | * | 13 | 0.4 |

seqinfo: 3 sequences from an unspecified genome

GRangesList accessors (continued)

```
> mcols(grl)$geneid <- c("GENE1", "GENE2")
```

```
> mcols(grl)
```

DataFrame with 2 rows and 1 column

```
  geneid  
<character>  
1    GENE1  
2    GENE2
```

```
> grl
```

GRangesList object of length 2:

\$TX1

GRanges object with 5 ranges and 2 metadata columns:

| | seqnames | ranges | strand | score | GC |
|---|----------|----------------|--------|-----------|-----------|
| | <Rle> | <IRanges> | <Rle> | <integer> | <numeric> |
| A | ch1 | [35016, 35020] | - | 11 | 1 |
| B | ch1 | [17, 20] | - | 12 | 0.8 |
| C | chMT | [18, 134] | + | 13 | 0.6 |
| D | chMT | [19, 20] | - | 14 | 0.4 |
| F | chMT | [121, 237] | + | 16 | 0 |

\$TX2

GRanges object with 3 ranges and 2 metadata columns:

| | seqnames | ranges | strand | score | GC |
|--|----------|--------|--------|-------|-----|
| | ch2 | [2, 7] | * | 15 | 0 |
| | ch2 | [1, 6] | * | 14 | 0.2 |
| | ch2 | [2, 7] | * | 13 | 0.4 |

seqinfo: 3 sequences from an unspecified genome

GRangesList accessors (continued)

```
> seqinfo(grl)
```

Seqinfo object with 3 sequences from an unspecified genome:

| seqnames | seqlengths | isCircular | genome |
|----------|------------|------------|--------|
| ch1 | 50000 | NA | <NA> |
| chMT | 800 | NA | <NA> |
| ch2 | NA | NA | <NA> |

Vector operations on GRangesList objects

Only the following *vector operations* are supported on GRangesList objects:

- ▶ `length()`, `names()`
- ▶ Single-bracket subsetting: `[`
- ▶ Combining: `c()`

Vector operations on GRangesList objects

```
> grl[c("TX2", "TX1")]
```

GRangesList object of length 2:

\$TX2

GRanges object with 3 ranges and 2 metadata columns:

| seqnames | ranges | strand | score | GC |
|----------|-----------|--------|-----------|-----------|
| <Rle> | <IRanges> | <Rle> | <integer> | <numeric> |
| ch2 | [2, 7] | * | 15 | 0 |
| ch2 | [1, 6] | * | 14 | 0.2 |
| ch2 | [2, 7] | * | 13 | 0.4 |

\$TX1

GRanges object with 5 ranges and 2 metadata columns:

| | seqnames | ranges | strand | score | GC |
|---|----------|----------------|--------|-------|-----|
| A | ch1 | [35016, 35020] | - | 11 | 1 |
| B | ch1 | [17, 20] | - | 12 | 0.8 |
| C | chMT | [18, 134] | + | 13 | 0.6 |
| D | chMT | [19, 20] | - | 14 | 0.4 |
| F | chMT | [121, 237] | + | 16 | 0 |

seqinfo: 3 sequences from an unspecified genome

Vector operations on GRangesList objects (continued)

```
> c(gr1, GRangesList(gr3))
```

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

```
$TX1
```

```
GRanges object with 5 ranges and 2 metadata columns:
```

| | seqnames | ranges | strand | score | GC |
|---|----------|----------------|--------|-----------|-----------|
| | <Rle> | <IRanges> | <Rle> | <integer> | <numeric> |
| A | ch1 | [35016, 35020] | - | 11 | 1 |
| B | ch1 | [17, 20] | - | 12 | 0.8 |
| C | chMT | [18, 134] | + | 13 | 0.6 |
| D | chMT | [19, 20] | - | 14 | 0.4 |
| F | chMT | [121, 237] | + | 16 | 0 |

```
$TX2
```

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

| | seqnames | ranges | strand | score | GC |
|--|----------|--------|--------|-------|-----|
| | ch2 | [2, 7] | * | 15 | 0 |
| | ch2 | [1, 6] | * | 14 | 0.2 |
| | ch2 | [2, 7] | * | 13 | 0.4 |

```
[[3]]
```

```
GRanges object with 5 ranges and 2 metadata columns:
```

| | seqnames | ranges | strand | score | GC |
|---|----------|----------------|--------|-------|-----|
| A | ch1 | [35016, 35020] | - | 11 | 1 |
| B | ch1 | [17, 20] | - | 12 | 0.8 |
| C | chMT | [18, 134] | + | 13 | 0.6 |
| D | chMT | [19, 20] | - | 14 | 0.4 |
| F | chMT | [121, 237] | + | 16 | 0 |

```
-----  
seqinfo: 3 sequences from an unspecified genome
```

List operations on GRangesList objects

What we call *list operations* are operations that work on an ordinary list:

- ▶ Double-bracket subsetting: `[[`
- ▶ `elementNROWS()`, `unlist()`
- ▶ `lapply()`, `sapply()`, `endoapply()`
- ▶ `mendoapply()` (not covered in this presentation)

GRangesList objects support all these *list operations* ==> They're considered *list-like* objects.

elementNROWS() and unlist()

```
> gr1[[2]]
```

GRanges object with 3 ranges and 2 metadata columns:

| seqnames | ranges | strand | score | GC |
|----------|-----------|--------|-----------|-----------|
| <Rle> | <IRanges> | <Rle> | <integer> | <numeric> |
| ch2 | [2, 7] | * | 15 | 0 |
| ch2 | [1, 6] | * | 14 | 0.2 |
| ch2 | [2, 7] | * | 13 | 0.4 |

seqinfo: 3 sequences from an unspecified genome

```
> elementNROWS(gr1)
```

TX1 TX2

5 3

```
> unlisted <- unlist(gr1, use.names=FALSE) # same as c(gr1[[1]], gr1[[2]])
```

```
> unlisted
```

GRanges object with 8 ranges and 2 metadata columns:

| | seqnames | ranges | strand | score | GC |
|---|----------|----------------|--------|-----------|-----------|
| | <Rle> | <IRanges> | <Rle> | <integer> | <numeric> |
| A | ch1 | [35016, 35020] | - | 11 | 1 |
| B | ch1 | [17, 20] | - | 12 | 0.8 |
| C | chMT | [18, 134] | + | 13 | 0.6 |
| . | ... | ... | ... | ... | ... |
| | ch2 | [2, 7] | * | 15 | 0 |
| | ch2 | [1, 6] | * | 14 | 0.2 |
| | ch2 | [2, 7] | * | 13 | 0.4 |

seqinfo: 3 sequences from an unspecified genome

relist()

```
> grl100 <- relist(shift(unlisted, 100), grl)
> grl100
```

GRangesList object of length 2:

\$TX1

GRanges object with 5 ranges and 2 metadata columns:

| | seqnames | ranges | strand | score | GC |
|---|----------|----------------|--------|-----------|-----------|
| | <Rle> | <IRanges> | <Rle> | <integer> | <numeric> |
| A | ch1 | [35116, 35120] | - | 11 | 1 |
| B | ch1 | [117, 120] | - | 12 | 0.8 |
| C | chMT | [118, 234] | + | 13 | 0.6 |
| D | chMT | [119, 120] | - | 14 | 0.4 |
| F | chMT | [221, 337] | + | 16 | 0 |

\$TX2

GRanges object with 3 ranges and 2 metadata columns:

| | seqnames | ranges | strand | score | GC |
|--|----------|------------|--------|-------|-----|
| | ch2 | [102, 107] | * | 15 | 0 |
| | ch2 | [101, 106] | * | 14 | 0.2 |
| | ch2 | [102, 107] | * | 13 | 0.4 |

seqinfo: 3 sequences from an unspecified genome

endoapply()

```
> grl100b <- endoapply(grl, shift, 100)
> grl100b

GRangesList object of length 2:
$TX1
GRanges object with 5 ranges and 2 metadata columns:
      seqnames      ranges strand |      score      GC
      <Rle>      <IRanges> <Rle> | <integer> <numeric>
A      ch1 [35116, 35120]   - |         11         1
B      ch1 [ 117, 120]     - |         12        0.8
C      chMT [ 118, 234]    + |         13        0.6
D      chMT [ 119, 120]    - |         14        0.4
F      chMT [ 221, 337]    + |         16         0

$TX2
GRanges object with 3 ranges and 2 metadata columns:
      seqnames      ranges strand |      score      GC
      <Rle>      <IRanges> <Rle> | <integer> <numeric>
ch2 [102, 107]      * |         15         0
ch2 [101, 106]      * |         14        0.2
ch2 [102, 107]      * |         13        0.4

-----
seqinfo: 3 sequences from an unspecified genome

> mcols(grl100)

DataFrame with 2 rows and 0 columns

> mcols(grl100b)

DataFrame with 2 rows and 1 column
      geneid
  <character>
1      GENE1
2      GENE2
```


Range-based operations on GRangesList objects

```
> grl
```

```
GRangesList object of length 2:
```

```
$TX1
```

```
GRanges object with 5 ranges and 2 metadata columns:
```

| | seqnames | ranges | strand | score | GC |
|---|----------|----------------|--------|-----------|-----------|
| | <Rle> | <IRanges> | <Rle> | <integer> | <numeric> |
| A | ch1 | [35016, 35020] | - | 11 | 1 |
| B | ch1 | [17, 20] | - | 12 | 0.8 |
| C | chMT | [18, 134] | + | 13 | 0.6 |
| D | chMT | [19, 20] | - | 14 | 0.4 |
| F | chMT | [121, 237] | + | 16 | 0 |

```
$TX2
```

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

| | seqnames | ranges | strand | score | GC |
|--|----------|--------|--------|-------|-----|
| | ch2 | [2, 7] | * | 15 | 0 |
| | ch2 | [1, 6] | * | 14 | 0.2 |
| | ch2 | [2, 7] | * | 13 | 0.4 |

```
-----  
seqinfo: 3 sequences from an unspecified genome
```

```
> shift(grl, 100)
```

```
GRangesList object of length 2:
```

```
$TX1
```

```
GRanges object with 5 ranges and 2 metadata columns:
```

| | seqnames | ranges | strand | score | GC |
|---|----------|----------------|--------|-----------|-----------|
| | <Rle> | <IRanges> | <Rle> | <integer> | <numeric> |
| A | ch1 | [35116, 35120] | - | 11 | 1 |
| B | ch1 | [117, 120] | - | 12 | 0.8 |
| C | chMT | [118, 234] | + | 13 | 0.6 |
| D | chMT | [119, 120] | - | 14 | 0.4 |
| F | chMT | [221, 337] | + | 16 | 0 |

```
$TX2
```

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

| | seqnames | ranges | strand | score | GC |
|--|----------|------------|--------|-------|-----|
| | ch2 | [102, 107] | * | 15 | 0 |
| | ch2 | [101, 106] | * | 14 | 0.2 |
| | ch2 | [102, 107] | * | 13 | 0.4 |

```
-----  
seqinfo: 3 sequences from an unspecified genome
```

`shift(grl, 100)` is equivalent to `endoapply(grl, shift, 100)`

Range-based operations on GRangesList objects (continued)

```
> grl
```

```
GRangesList object of length 2:
```

```
$TX1
```

```
GRanges object with 5 ranges and 2 metadata columns:
```

| | seqnames | ranges | strand | score | GC |
|---|----------|----------------|--------|-----------|-----------|
| | <Rle> | <IRanges> | <Rle> | <integer> | <numeric> |
| A | ch1 | [35016, 35020] | - | 11 | 1 |
| B | ch1 | [17, 20] | - | 12 | 0.8 |
| C | chMT | [18, 134] | + | 13 | 0.6 |
| D | chMT | [19, 20] | - | 14 | 0.4 |
| F | chMT | [121, 237] | + | 16 | 0 |

```
$TX2
```

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

| | seqnames | ranges | strand | score | GC |
|--|----------|--------|--------|-------|-----|
| | ch2 | [2, 7] | * | 15 | 0 |
| | ch2 | [1, 6] | * | 14 | 0.2 |
| | ch2 | [2, 7] | * | 13 | 0.4 |

```
-----  
seqinfo: 3 sequences from an unspecified genome
```

```
> flank(grl, 10)
```

```
GRangesList object of length 2:
```

```
$TX1
```

```
GRanges object with 5 ranges and 2 metadata columns:
```

| | seqnames | ranges | strand | score | GC |
|---|----------|----------------|--------|-----------|-----------|
| | <Rle> | <IRanges> | <Rle> | <integer> | <numeric> |
| A | ch1 | [35021, 35030] | - | 11 | 1 |
| B | ch1 | [21, 30] | - | 12 | 0.8 |
| C | chMT | [8, 17] | + | 13 | 0.6 |
| D | chMT | [21, 30] | - | 14 | 0.4 |
| F | chMT | [111, 120] | + | 16 | 0 |

```
$TX2
```

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

| | seqnames | ranges | strand | score | GC |
|--|----------|---------|--------|-------|-----|
| | ch2 | [-8, 1] | * | 15 | 0 |
| | ch2 | [-9, 0] | * | 14 | 0.2 |
| | ch2 | [-8, 1] | * | 13 | 0.4 |

```
-----  
seqinfo: 3 sequences from an unspecified genome
```

`flank(grl, 10)` is equivalent to `endoapply(grl, flank, 10)`

Range-based operations on GRangesList objects (continued)

```
> grl

GRangesList object of length 2:
$TX1
GRanges object with 5 ranges and 2 metadata columns:
      seqnames      ranges strand |      score      GC
      <Rle>        <IRanges> <Rle> | <integer> <numeric>
A      ch1 [35016, 35020]    - |         11         1
B      ch1 [  17,   20]    - |         12        0.8
C      chMT [  18,  134]    + |         13        0.6
D      chMT [  19,   20]    - |         14        0.4
F      chMT [ 121,  237]    + |         16         0

$TX2
GRanges object with 3 ranges and 2 metadata columns:
      seqnames ranges strand | score  GC
      <Rle>   <IRanges> <Rle> | <integer> <numeric>
ch2 [2, 7]      * |    15  0
ch2 [1, 6]      * |    14 0.2
ch2 [2, 7]      * |    13 0.4

-----
seqinfo: 3 sequences from an unspecified genome
```

```
> range(grl)

GRangesList object of length 2:
$TX1
GRanges object with 3 ranges and 0 metadata columns:
      seqnames      ranges strand
      <Rle>        <IRanges> <Rle>
[1]      ch1 [17, 35020]    -
[2]     chMT [18,  237]     +
[3]     chMT [19,   20]     -

$TX2
GRanges object with 1 range and 0 metadata columns:
      seqnames ranges strand
[1]      ch2 [1, 7]      *

-----
seqinfo: 3 sequences from an unspecified genome
```

`range(grl)` is equivalent to `endoapply(grl, range)`

Range-based operations on GRangesList objects (continued)

```
> grl
```

```
GRangesList object of length 2:
```

```
$TX1
```

```
GRanges object with 5 ranges and 2 metadata columns:
```

| | seqnames | ranges | strand | score | GC |
|---|----------|----------------|--------|-----------|-----------|
| | <Rle> | <IRanges> | <Rle> | <integer> | <numeric> |
| A | ch1 | [35016, 35020] | - | 11 | 1 |
| B | ch1 | [17, 20] | - | 12 | 0.8 |
| C | chMT | [18, 134] | + | 13 | 0.6 |
| D | chMT | [19, 20] | - | 14 | 0.4 |
| F | chMT | [121, 237] | + | 16 | 0 |

```
$TX2
```

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

| | seqnames | ranges | strand | score | GC |
|--|----------|--------|--------|-------|-----|
| | ch2 | [2, 7] | * | 15 | 0 |
| | ch2 | [1, 6] | * | 14 | 0.2 |
| | ch2 | [2, 7] | * | 13 | 0.4 |

```
-----  
seqinfo: 3 sequences from an unspecified genome
```

```
> reduce(grl)
```

```
GRangesList object of length 2:
```

```
$TX1
```

```
GRanges object with 4 ranges and 0 metadata columns:
```

| | seqnames | ranges | strand |
|-----|----------|----------------|--------|
| | <Rle> | <IRanges> | <Rle> |
| [1] | ch1 | [17, 20] | - |
| [2] | ch1 | [35016, 35020] | - |
| [3] | chMT | [18, 237] | + |
| [4] | chMT | [19, 20] | - |

```
$TX2
```

```
GRanges object with 1 range and 0 metadata columns:
```

| | seqnames | ranges | strand |
|-----|----------|--------|--------|
| [1] | ch2 | [1, 7] | * |

```
-----  
seqinfo: 3 sequences from an unspecified genome
```

`reduce(grl)` is equivalent to `endoapply(grl, reduce)`

Range-based operations on GRangesList objects (continued)

```
> grl2

GRangesList object of length 2:
$TX1
GRanges object with 1 range and 2 metadata columns:
  seqnames      ranges strand |   score      GC
   <Rle> <IRanges> <Rle> | <integer> <numeric>
C      chMT [18, 134]      + |      13      0.6

$TX2
GRanges object with 1 range and 2 metadata columns:
  seqnames ranges strand | score GC
   <Rle> <IRanges> <Rle> | <integer> <numeric>
ch2 [2, 7]      * |    15  0

-----
seqinfo: 3 sequences from an unspecified genome

> grl3

GRangesList object of length 2:
[[1]]
GRanges object with 1 range and 2 metadata columns:
  seqnames      ranges strand |   score      GC
   <Rle> <IRanges> <Rle> | <integer> <numeric>
chMT [22, 130]      + |      13      0.6

[[2]]
GRanges object with 1 range and 2 metadata columns:
  seqnames ranges strand | score GC
   <Rle> <IRanges> <Rle> | <integer> <numeric>
ch2 [2, 7]      * |    15  0

-----
seqinfo: 3 sequences from an unspecified genome
```

```
> setdiff(grl2, grl3)

GRangesList object of length 2:
$TX1
GRanges object with 2 ranges and 0 metadata columns:
  seqnames      ranges strand
   <Rle> <IRanges> <Rle>
[1]      chMT [ 18,  21]      +
[2]      chMT [131, 134]      +

$TX2
GRanges object with 0 ranges and 0 metadata columns:
  seqnames ranges strand

-----
seqinfo: 3 sequences from an unspecified genome
```

`setdiff(grl2, grl)` is equivalent to `mendoapply(setdiff, grl2, grl)`

Other resources

- ▶ Great slides from Michael on ranges sequences and alignments:
http://bioconductor.org/help/course-materials/2014/CSAMA2014/2_Tuesday/lectures/Ranges_Sequences_and_Alignments-Lawrence.pdf
- ▶ Vignettes in the *GenomicRanges* package (`browseVignettes("GenomicRanges")`).
- ▶ `GRanges` and `GRangesList` man pages in the *GenomicRanges* package.
- ▶ Vignettes and `GAlignments` man page in the *GenomicAlignments* package.
- ▶ *Bioconductor* support site: <http://support.bioconductor.org/>
- ▶ The *genomic ranges* paper: Michael Lawrence, Wolfgang Huber, Hervé Pagès, Patrick Aboyoun, Marc Carlson, Robert Gentleman, Martin T. Morgan, Vincent J. Carey. Software for Computing and Annotating Genomic Ranges. *PLOS Computational Biology*, 4(3), 2013.