Range-based containers in *Bioconductor*

Hervé Pagès
hpages@fhcrc.org

Fred Hutchinson Cancer Research Center
Seattle, WA, USA

21 January 2014

- Introduction
- `IRanges` objects
  - Constructor and accessors
  - Vector operations
  - Range-based operations
- `GRanges` objects
  - Constructor and accessors
  - Vector operations
  - Range-based operations
  - Splitting a `GRanges` object
- `GRangesList` objects
  - Constructor and accessors
  - Vector operations
  - List operations
  - Range-based operations
- `GAlignments` objects
  - Constructor and accessors
  - Coercion to `GRanges` or `GRangesList`
- `GAlignmentPairs` objects
  - Constructor and accessors
  - Coercion to `GRangesList`
- Advanced operations
  - Coverage and slicing
  - Finding/counting overlaps
- Resources

- Introduction
- `IRanges` objects
  - Constructor and accessors
  - Vector operations
  - Range-based operations
- `GRanges` objects
  - Constructor and accessors
  - Vector operations
  - Range-based operations
  - Splitting a `GRanges` object
- `GRangesList` objects
  - Constructor and accessors
  - Vector operations
  - List operations
  - Range-based operations
- `GAlignments` objects
  - Constructor and accessors
  - Coercion to `GRanges` or `GRangesList`
- `GAlignmentPairs` objects
  - Constructor and accessors
  - Coercion to `GRangesList`
- Advanced operations
  - Coverage and slicing
  - Finding/counting overlaps
- Resources

Range-based containers in *Bioconductor*

Implemented and documented in the *IRanges* package:

- `IRanges`

Implemented and documented in the *GenomicRanges* package:

- `GRanges`
- `GRangesList`
- `GAlignments`
- `GAlignmentPairs`
- `GAlignmentsList` (not covered in this presentation)

About the implementation

S4 classes (a.k.a. *formal* classes) –> relies heavily on the *methods* package.

Current implementation tries to provide an API that is as consistent as possible. In particular:

- The end-user should never need to use `new()`: a *constructor*, named as the container, is provided for each container. E.g. `GRanges()`.
- The end-user should never need to use `@` (a.k.a. *direct slot access*): slot *accessors* (*getters* and *setters*) are provided for each container. Not all getters have a corresponding setter!
- Standard functions/operators like `length()`, `names()`, `[`, `c()`, `[[`, `$`, etc... work almost everywhere and behave "as expected".
- Additional functions that work almost everywhere: `mcols()`, `elementLengths()`, `seqinfo()`, etc...
- Consistent display (`show` methods).

Basic operations

Vector operations

Operate on *vector-like* objects

(e.g. on `Rle`, `IRanges`, `GRanges`, `DNAStringSet`, etc... objects)

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

List operations

Operate on *list-like* objects[a]

(e.g. on `IRangesList`, `GRangesList`, `DNAStringSetList`, etc... objects)

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

---

[a] *list-like* objects are also *vector-like* objects

Coercion methods

- `as()`
- S3-style form: `as.vector()`, `as.character()`, `as.factor()`, etc...

Range-based operations

Range-based operations operate on *range-based* objects

(e.g. on `IRanges`, `IRangesList`, `GRanges`, `GRangesList`, etc... objects)

Intra range transformations
`shift()`, `narrow()`, `flank()`, `resize()`

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

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...

- Introduction
- `IRanges` objects
  - Constructor and accessors
  - Vector operations
  - Range-based operations
- `GRanges` objects
  - Constructor and accessors
  - Vector operations
  - Range-based operations
  - Splitting a `GRanges` object
- `GRangesList` objects
  - Constructor and accessors
  - Vector operations
  - List operations
  - Range-based operations
- `GAlignments` objects
  - Constructor and accessors
  - Coercion to `GRanges` or `GRangesList`
- `GAlignmentPairs` objects
  - Constructor and accessors
  - Coercion to `GRangesList`
- Advanced operations
  - Coverage and slicing
  - Finding/counting overlaps
- Resources

The purpose of the `IRanges` container is...

... to store a set of *integer ranges* (a.k.a. *integer intervals*).

- Each range can be defined by a *start* and an *end* value: both are included in the interval (except when the range is empty).
- The *width* of the range is the number of integer values in it: *width* = *end* - *start* + 1.
- *end* is always >= *start*, except for empty ranges (a.k.a. zero-width ranges) where *end* = *start* - 1.

Supported operations

- *Vector operations*: **YES** (splitting/relisting produces an `IRangesList` object)
- *List operations*: **YES** (not covered in this presentation)
- *Coercion methods*: **YES** (from logical or integer vector to `IRanges`)
- *Range-based operations*: **YES**

- Introduction
- IRanges objects
  - Constructor and accessors
  - Vector operations
  - Range-based operations
- GRanges objects
  - Constructor and accessors
  - Vector operations
  - Range-based operations
  - Splitting a GRanges object
- GRangesList objects
  - Constructor and accessors
  - Vector operations
  - List operations
  - Range-based operations
- GAlignments objects
  - Constructor and accessors
  - Coercion to GRanges or GRangesList
- GAlignmentPairs objects
  - Constructor and accessors
  - Coercion to GRangesList
- Advanced operations
  - Coverage and slicing
  - Finding/counting overlaps
- Resources

IRanges constructor and accessors

```
> library(IRanges)
> ir1 <- IRanges(start=c(12, -9, NA, 12),
+                end=c(NA, 0, 15, NA),
+                width=c(4, NA, 4, 3))
> ir1  # "show" method not yet consistent with the other "show" methods (TODO)

IRanges of length 4
    start end width
[1]    12  15     4
[2]    -9   0    10
[3]    12  15     4
[4]    12  14     3

> start(ir1)

[1] 12 -9 12 12

> end(ir1)

[1] 15  0 15 14

> width(ir1)

[1]  4 10  4  3

> successiveIRanges(c(10, 5, 38), from=101)

IRanges of length 3
    start end width
[1]   101 110    10
[2]   111 115     5
[3]   116 153    38
```

IRanges accessors (continued)

```
> names(ir1) <- LETTERS[1:4]
> names(ir1)

[1] "A" "B" "C" "D"

> mcols(ir1) <- DataFrame(score=11:14, GC=seq(1, 0, length=4))
> mcols(ir1)

DataFrame with 4 rows and 2 columns
      score        GC
  <integer> <numeric>
1        11 1.0000000
2        12 0.6666667
3        13 0.3333333
4        14 0.0000000

> ir1

IRanges of length 4
    start end width names
[1]    12  15     4     A
[2]    -9   0    10     B
[3]    12  15     4     C
[4]    12  14     3     D
```

Introduction

`IRanges` objects
- Constructor and accessors
- Vector operations
- Range-based operations

`GRanges` objects
- Constructor and accessors
- Vector operations
- Range-based operations
- Splitting a `GRanges` object

`GRangesList` objects
- Constructor and accessors
- Vector operations
- List operations
- Range-based operations

`GAlignments` objects
- Constructor and accessors
- Coercion to `GRanges` or `GRangesList`

`GAlignmentPairs` objects
- Constructor and accessors
- Coercion to `GRangesList`

Advanced operations
- Coverage and slicing
- Finding/counting overlaps

Resources

Vector operations on IRanges objects

```
> ir1[-2]

IRanges of length 3
    start end width names
[1]    12  15     4     A
[2]    12  15     4     C
[3]    12  14     3     D

> ir2 <- c(ir1, IRanges(-10, 0))
> ir2

IRanges of length 5
    start end width names
[1]    12  15     4     A
[2]    -9   0    10     B
[3]    12  15     4     C
[4]    12  14     3     D
[5]   -10   0    11
```

```
> duplicated(ir2)

[1] FALSE FALSE  TRUE FALSE FALSE

> unique(ir2)

IRanges of length 4
    start end width names
[1]    12  15     4     A
[2]    -9   0    10     B
[3]    12  14     3     D
[4]   -10   0    11
```

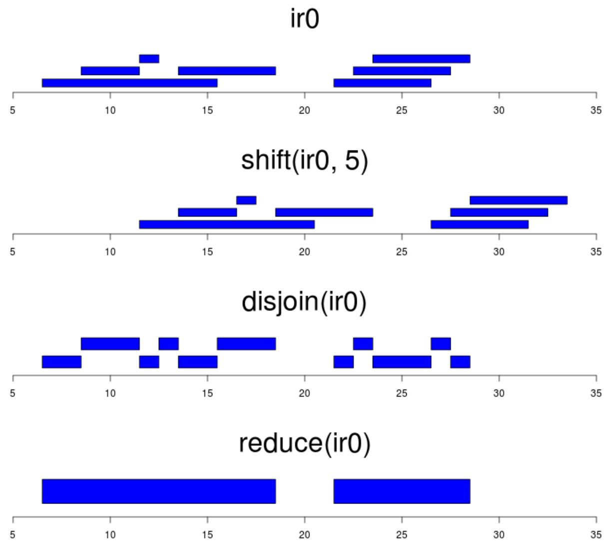
```
> order(ir2)

[1] 5 2 4 1 3

> sort(ir2)

IRanges of length 5
    start end width names
[1]   -10   0    11
[2]    -9   0    10     B
[3]    12  14     3     D
[4]    12  15     4     A
[5]    12  15     4     C
```

```
> ok <- c(FALSE, FALSE, TRUE, TRUE, TRUE, FALSE, FALSE, TRUE)
> ir4 <- as(ok, "IRanges")  # from logical vector to IRanges
> ir4

IRanges of length 2
    start end width
[1]     3   5     3
[2]     8   8     1
```

```
> as.data.frame(ir4)

  start end width
1     3   5     3
2     8   8     1
```

- Introduction
- `IRanges` objects
  - Constructor and accessors
  - Vector operations
  - Range-based operations
- `GRanges` objects
  - Constructor and accessors
  - Vector operations
  - Range-based operations
  - Splitting a `GRanges` object
- `GRangesList` objects
  - Constructor and accessors
  - Vector operations
  - List operations
  - Range-based operations
- `GAlignments` objects
  - Constructor and accessors
  - Coercion to `GRanges` or `GRangesList`
- `GAlignmentPairs` objects
  - Constructor and accessors
  - Coercion to `GRangesList`
- Advanced operations
  - Coverage and slicing
  - Finding/counting overlaps
- Resources

Range-based operations on `IRanges` objects



Range-based operations on IRanges objects (continued)

```
> ir1

IRanges of length 4
    start end width names
[1]    12  15     4     A
[2]    -9   0    10     B
[3]    12  15     4     C
[4]    12  14     3     D
```

```
> shift(ir1, -start(ir1))

IRanges of length 4
    start end width names
[1]     0   3     4     A
[2]     0   9    10     B
[3]     0   3     4     C
[4]     0   2     3     D

> flank(ir1, 10, start=FALSE)

IRanges of length 4
    start end width names
[1]    16  25    10     A
[2]     1  10    10     B
[3]    16  25    10     C
[4]    15  24    10     D
```

Range-based operations on IRanges objects (continued)

```
> ir1

IRanges of length 4
    start end width names
[1]    12  15     4     A
[2]    -9   0    10     B
[3]    12  15     4     C
[4]    12  14     3     D
```

```
> range(ir1)

IRanges of length 1
    start end width
[1]    -9  15    25

> reduce(ir1)

IRanges of length 2
    start end width
[1]    -9   0    10
[2]    12  15     4
```

```
> union(ir1, IRanges(-2, 6))

IRanges of length 2
    start end width
[1]    -9   6    16
[2]    12  15     4

> intersect(ir1, IRanges(-2, 13))

IRanges of length 2
    start end width
[1]    -2   0     3
[2]    12  13     2

> setdiff(ir1, IRanges(-2, 13))

IRanges of length 2
    start end width
[1]    -9  -3     7
[2]    14  15     2
```

Range-based operations on IRanges objects (continued)

```
> ir3 <- IRanges(5:1, width=12)
> ir3

IRanges of length 5
    start end width
[1]     5  16    12
[2]     4  15    12
[3]     3  14    12
[4]     2  13    12
[5]     1  12    12
```

```
> ir2

IRanges of length 5
    start end width names
[1]    12  15     4     A
[2]    -9   0    10     B
[3]    12  15     4     C
[4]    12  14     3     D
[5]   -10   0    11
```

```
> pintersect(ir3, ir2, resolve.empty="max.start")

IRanges of length 5
    start end width names
[1]    12  15     4     A
[2]     4   3     0     B
[3]    12  14     3     C
[4]    12  13     2     D
[5]     1   0     0
```

- Introduction
- `IRanges` objects
  - Constructor and accessors
  - Vector operations
  - Range-based operations
- **`GRanges` objects**
  - Constructor and accessors
  - Vector operations
  - Range-based operations
  - Splitting a `GRanges` object
- `GRangesList` objects
  - Constructor and accessors
  - Vector operations
  - List operations
  - Range-based operations
- `GAlignments` objects
  - Constructor and accessors
  - Coercion to `GRanges` or `GRangesList`
- `GAlignmentPairs` objects
  - Constructor and accessors
  - Coercion to `GRangesList`
- Advanced operations
  - Coverage and slicing
  - Finding/counting overlaps
- Resources

The purpose of the `GRanges` container is...

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

- Like for `IRanges` objects, each range can be defined by a *start* and an *end* value.
- In addition, each range is also assigned a chromosome name and a strand.
- *start* and *end* are both **1-based** positions relative to the 5' end of the plus strand of the chromosome (a.k.a. *reference sequence*), even when the range is on the minus strand.
- So the *start* is always the leftmost position and the *end* the rightmost, even when the range is on the minus strand.
- As a consequence, **if a genomic range represents a gene on the minus strand, the gene "starts" (biologically speaking) at the end of it**.

Supported operations

- *Vector operations*: **YES** (splitting/relisting produces a `GRangesList` object)
- *List operations*: **NO**
- *Coercion methods*: to `IRangesList` (not covered in this presentation)
- *Range-based operations*: **YES**

- Introduction
- `IRanges` objects
    - Constructor and accessors
    - Vector operations
    - Range-based operations
- `GRanges` objects
    - Constructor and accessors
    - Vector operations
    - Range-based operations
    - Splitting a `GRanges` object
- `GRangesList` objects
    - Constructor and accessors
    - Vector operations
    - List operations
    - Range-based operations
- `GAlignments` objects
    - Constructor and accessors
    - Coercion to `GRanges` or `GRangesList`
- `GAlignmentPairs` objects
    - Constructor and accessors
    - Coercion to `GRangesList`
- Advanced operations
    - Coverage and slicing
    - Finding/counting overlaps
- Resources

GRanges constructor

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

GRanges 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]      *
  ---
  seqlengths:
    ch1 chMT
     NA   NA
```

GRanges accessors

```
> 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 of length 6
    start end width
[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 (continued)

```
> 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 (continued)

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

[1] "A" "B" "C" "D" "E" "F"

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

> gr1

GRanges 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
  ---
  seqlengths:
    ch1 chMT
     NA   NA
```

GRanges accessors (continued)

```
> seqinfo(gr1)

Seqinfo of length 2
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
```

Introduction

`IRanges` objects
- Constructor and accessors
- Vector operations
- Range-based operations

`GRanges` objects
- Constructor and accessors
- Vector operations
- Range-based operations
- Splitting a `GRanges` object

`GRangesList` objects
- Constructor and accessors
- Vector operations
- List operations
- Range-based operations

`GAlignments` objects
- Constructor and accessors
- Coercion to `GRanges` or `GRangesList`

`GAlignmentPairs` objects
- Constructor and accessors
- Coercion to `GRangesList`

Advanced operations
- Coverage and slicing
- Finding/counting overlaps

Resources

Vector operations on GRanges objects

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

GRanges 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
  ---
  seqlengths:
     ch1  chMT
   50000   800

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

GRanges 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
  ---
  seqlengths:
     ch1  chMT
   50000   800
```

Vector operations on GRanges objects (continued)

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

GRanges 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
  ---
  seqlengths:
     ch1  chMT
   50000   800
```

Vector operations on GRanges objects (continued)

```
> 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 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
  D     chMT  [19, 20]      - |        14       0.4
  F     chMT  [21, 20]      + |        16         0
         ch2  [ 2,  7]      * |        15         0
         ch2  [ 1,  6]      * |        14       0.2
         ch2  [ 2,  7]      * |        13       0.4
  ---
  seqlengths:
     ch1  chMT   ch2
   50000   800    NA
```

Vector operations on GRanges objects (continued)

```
> 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 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
  ---
  seqlengths:
     ch1  chMT   ch2
   50000   800    NA
```

Vector operations on GRanges objects (continued)

```
> sort(gr12)

GRanges 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
  F     chMT  [21, 20]      + |        16         0
  D     chMT  [19, 20]      - |        14       0.4
         ch2  [ 1,  6]      * |        14       0.2
         ch2  [ 2,  7]      * |        15         0
         ch2  [ 2,  7]      * |        13       0.4
  ---
  seqlengths:
     ch1  chMT   ch2
   50000   800    NA
```

- Introduction
- `IRanges` objects
    - Constructor and accessors
    - Vector operations
    - Range-based operations
- `GRanges` objects
    - Constructor and accessors
    - Vector operations
    - **Range-based operations**
    - Splitting a `GRanges` object
- `GRangesList` objects
    - Constructor and accessors
    - Vector operations
    - List operations
    - Range-based operations
- `GAlignments` objects
    - Constructor and accessors
    - Coercion to `GRanges` or `GRangesList`
- `GAlignmentPairs` objects
    - Constructor and accessors
    - Coercion to `GRangesList`
- Advanced operations
    - Coverage and slicing
    - Finding/counting overlaps
- Resources

Range-based operations on GRanges objects

```
> gr2

GRanges 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
  ---
  seqlengths:
   ch2
    NA

> shift(gr2, 50)

GRanges 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
  ---
  seqlengths:
   ch2
    NA

> narrow(gr2, start=2, end=-2)

GRanges with 3 ranges and 2 metadata columns:
      seqnames    ranges strand |     score        GC
         <Rle> <IRanges>  <Rle> | <integer> <numeric>
  [1]      ch2    [3, 6]      * |        15         0
  [2]      ch2    [2, 5]      * |        14       0.2
  [3]      ch2    [3, 6]      * |        13       0.4
  ---
  seqlengths:
   ch2
    NA
```

Range-based operations on GRanges objects (continued)

```
> gr1

GRanges 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
  ---
  seqlengths:
     ch1  chMT
   50000   800

> resize(gr1, 12)

GRanges 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
  ---
  seqlengths:
     ch1  chMT
   50000   800
```

Range-based operations on GRanges objects (continued)

```
> gr1

GRanges 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
  ---
  seqlengths:
     ch1  chMT
   50000   800

> flank(gr1, 3)

GRanges 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
  ---
  seqlengths:
     ch1  chMT
   50000   800
```

Range-based operations on GRanges objects (continued)

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

GRanges 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
  ---
  seqlengths:
     ch1  chMT
   50000   800

> range(gr3)

GRanges 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]      -
  ---
  seqlengths:
     ch1  chMT
   50000   800
```

Range-based operations on GRanges objects (continued)

```
> gr3

GRanges 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
  ---
  seqlengths:
     ch1  chMT
   50000   800

> disjoin(gr3)

GRanges with 6 ranges and 0 metadata columns:
      seqnames         ranges strand
         <Rle>      <IRanges>  <Rle>
  [1]      ch1 [   17,    20]      -
  [2]      ch1 [35016, 35020]      -
  [3]     chMT [   18,   120]      +
  [4]     chMT [  121,   134]      +
  [5]     chMT [  135,   237]      +
  [6]     chMT [   19,    20]      -
  ---
  seqlengths:
     ch1  chMT
   50000   800
```

Range-based operations on GRanges objects (continued)

```
> gr3

GRanges 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
  ---
  seqlengths:
     ch1  chMT
   50000   800

> reduce(gr3)

GRanges 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]      -
  ---
  seqlengths:
     ch1  chMT
   50000   800
```

Range-based operations on GRanges objects (continued)

```
> gr3
GRanges 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
  ---
  seqlengths:
     ch1  chMT
   50000   800

> gaps(gr3)
GRanges with 10 ranges and 0 metadata columns:
       seqnames         ranges strand
          <Rle>      <IRanges>  <Rle>
   [1]      ch1 [    1, 50000]      +
   [2]      ch1 [    1,    16]      -
   [3]      ch1 [   21, 35015]      -
   [4]      ch1 [35021, 50000]      -
   [5]      ch1 [    1, 50000]      *
   [6]     chMT [    1,    17]      +
   [7]     chMT [  238,   800]      +
   [8]     chMT [    1,    18]      -
   [9]     chMT [   21,   800]      -
  [10]     chMT [    1,   800]      *
  ---
  seqlengths:
     ch1  chMT
   50000   800
```

- Introduction
- IRanges objects
  - Constructor and accessors
  - Vector operations
  - Range-based operations
- GRanges objects
  - Constructor and accessors
  - Vector operations
  - Range-based operations
  - Splitting a GRanges object
- GRangesList objects
  - Constructor and accessors
  - Vector operations
  - List operations
  - Range-based operations
- GAlignments objects
  - Constructor and accessors
  - Coercion to GRanges or GRangesList
- GAlignmentPairs objects
  - Constructor and accessors
  - Coercion to GRangesList
- Advanced operations
  - Coverage and slicing
  - Finding/counting overlaps
- Resources

Splitting a GRanges object

```
> split(gr3, seqnames(gr3))

GRangesList of length 2:
$ch1
GRanges with 2 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

$chMT
GRanges with 3 ranges and 2 metadata columns:
    seqnames     ranges strand | score  GC
  C     chMT [ 18, 134]      + |    13 0.6
  D     chMT [ 19,  20]      - |    14 0.4
  F     chMT [121, 237]      + |    16   0

---
seqlengths:
   ch1  chMT
 50000   800
```

- Introduction
- IRanges objects
  - Constructor and accessors
  - Vector operations
  - Range-based operations
- GRanges objects
  - Constructor and accessors
  - Vector operations
  - Range-based operations
  - Splitting a GRanges object
- GRangesList objects
  - Constructor and accessors
  - Vector operations
  - List operations
  - Range-based operations
- GAlignments objects
  - Constructor and accessors
  - Coercion to GRanges or GRangesList
- GAlignmentPairs objects
  - Constructor and accessors
  - Coercion to GRangesList
- Advanced operations
  - Coverage and slicing
  - Finding/counting overlaps
- Resources

The purpose of the `GRangesList` container is...

... to store 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).

Supported operations

- *Vector operations*: **partially supported** (no splitting/relisting, no comparing or ordering)
- *List operations*: **YES**
- *Coercion methods*: to `IRangesList` (not covered in this presentation)
- *Range-based operations*: **partially supported** (some operations like `gaps()` are missing but they could/will be added)

- Introduction
- `IRanges` objects
  - Constructor and accessors
  - Vector operations
  - Range-based operations
- `GRanges` objects
  - Constructor and accessors
  - Vector operations
  - Range-based operations
  - Splitting a `GRanges` object
- `GRangesList` objects
  - Constructor and accessors
  - Vector operations
  - List operations
  - Range-based operations
- `GAlignments` objects
  - Constructor and accessors
  - Coercion to `GRanges` or `GRangesList`
- `GAlignmentPairs` objects
  - Constructor and accessors
  - Coercion to `GRangesList`
- Advanced operations
  - Coverage and slicing
  - Finding/counting overlaps
- Resources

GRangesList constructor

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

GRangesList of length 2:
[[1]]
GRanges 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 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

---
seqlengths:
   ch1  chMT   ch2
 50000   800    NA
```

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 of length 5
    start   end width names
[1] 35016 35020     5     A
[2]    17    20     4     B
[3]    18   134   117     C
[4]    19    20     2     D
[5]   121   237   117     F

[[2]]
IRanges of length 3
    start end width names
[1]     2   7     6
[2]     1   6     6
[3]     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 of length 2:
$TX1
GRanges 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 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


---
seqlengths:
   ch1  chMT   ch2
 50000   800    NA
```

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 of length 2:
$TX1
GRanges 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 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

---
seqlengths:
   ch1  chMT   ch2
 50000   800    NA
```

GRangesList accessors (continued)

```
> seqinfo(grl)

Seqinfo of length 3
seqnames seqlengths isCircular genome
ch1           50000         NA   <NA>
chMT            800         NA   <NA>
ch2              NA         NA   <NA>
```

- Introduction
- IRanges objects
  - Constructor and accessors
  - Vector operations
  - Range-based operations
- GRanges objects
  - Constructor and accessors
  - Vector operations
  - Range-based operations
  - Splitting a GRanges object
- GRangesList objects
  - Constructor and accessors
  - Vector operations
  - List operations
  - Range-based operations
- GAlignments objects
  - Constructor and accessors
  - Coercion to GRanges or GRangesList
- GAlignmentPairs objects
  - Constructor and accessors
  - Coercion to GRangesList
- Advanced operations
  - Coverage and slicing
  - Finding/counting overlaps
- Resources

Vector operations on GRangesList objects

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

GRangesList of length 2:
$TX2
GRanges 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 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

---
seqlengths:
   ch1  chMT   ch2
 50000   800    NA
```

Vector operations on `GRangesList` objects (continued)

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

GRangesList of length 3:
$TX1
GRanges 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 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 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

---
seqlengths:
   ch1  chMT   ch2
 50000   800    NA
```

Introduction

`IRanges` objects
- Constructor and accessors
- Vector operations
- Range-based operations

`GRanges` objects
- Constructor and accessors
- Vector operations
- Range-based operations
- Splitting a `GRanges` object

`GRangesList` objects
- Constructor and accessors
- Vector operations
- List operations
- Range-based operations

`GAlignments` objects
- Constructor and accessors
- Coercion to `GRanges` or `GRangesList`

`GAlignmentPairs` objects
- Constructor and accessors
- Coercion to `GRangesList`

Advanced operations
- Coverage and slicing
- Finding/counting overlaps

Resources

List operations on GRangesList objects

```
> grl[[2]]

GRanges 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
  ---
  seqlengths:
     ch1  chMT   ch2
   50000   800    NA

> elementLengths(grl)

TX1 TX2
  5   3

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

GRanges 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
  D     chMT [   19,    20]      - |        14       0.4
  F     chMT [  121,   237]      + |        16         0
         ch2 [    2,     7]      * |        15         0
         ch2 [    1,     6]      * |        14       0.2
         ch2 [    2,     7]      * |        13       0.4
  ---
  seqlengths:
     ch1  chMT   ch2
   50000   800    NA
```

List operations on GRangesList objects (continued)

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

GRangesList of length 2:
$TX1
GRanges 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 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


---
seqlengths:
   ch1  chMT   ch2
 50000   800    NA
```

List operations on GRangesList objects (continued)

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

GRangesList of length 2:
$TX1
GRanges 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 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

---
seqlengths:
   ch1  chMT   ch2
 50000   800    NA

> mcols(grl100)

DataFrame with 2 rows and 0 columns

> mcols(grl100b)

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

Introduction

`IRanges` objects
  Constructor and accessors
  Vector operations
  Range-based operations

`GRanges` objects
  Constructor and accessors
  Vector operations
  Range-based operations
  Splitting a `GRanges` object

`GRangesList` objects
  Constructor and accessors
  Vector operations
  List operations
  Range-based operations

`GAlignments` objects
  Constructor and accessors
  Coercion to `GRanges` or `GRangesList`

`GAlignmentPairs` objects
  Constructor and accessors
  Coercion to `GRangesList`

Advanced operations
  Coverage and slicing
  Finding/counting overlaps

Resources

Range-based operations on `GRangesList` objects

```
> grl

GRangesList of length 2:
$TX1
GRanges 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 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

---
seqlengths:
   ch1 chMT   ch2
 50000  800    NA
```

```
> shift(grl, 100)

GRangesList of length 2:
$TX1
GRanges 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 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

---
seqlengths:
   ch1 chMT   ch2
 50000  800    NA
```

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

Range-based operations on `GRangesList` objects (continued)

```
> grl

GRangesList of length 2:
$TX1
GRanges 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 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

---
seqlengths:
   ch1 chMT   ch2
 50000  800    NA
```

```
> flank(grl, 10)

GRangesList of length 2:
$TX1
GRanges 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 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

---
seqlengths:
   ch1 chMT   ch2
 50000  800    NA
```

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

Range-based operations on `GRangesList` objects (continued)

```
> grl

GRangesList of length 2:
$TX1
GRanges 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 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

---
seqlengths:
   ch1  chMT   ch2
 50000   800    NA
```

```
> range(grl)

GRangesList of length 2:
$TX1
GRanges 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 with 1 range and 0 metadata columns:
      seqnames ranges strand
  [1]      ch2 [1, 7]      *

---
seqlengths:
   ch1  chMT   ch2
 50000   800    NA
```

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

Range-based operations on `GRangesList` objects (continued)

```
> grl

GRangesList of length 2:
$TX1
GRanges 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 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

---
seqlengths:
   ch1  chMT   ch2
 50000   800    NA
```

```
> reduce(grl)

GRangesList of length 2:
$TX1
GRanges 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 with 1 range and 0 metadata columns:
      seqnames ranges strand
  [1]      ch2 [1, 7]      *

---
seqlengths:
   ch1  chMT   ch2
 50000   800    NA
```

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

Range-based operations on `GRangesList` objects (continued)

```
> grl2

GRangesList of length 2:
$TX1
GRanges 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 with 1 range and 2 metadata columns:
   seqnames ranges strand | score GC
        ch2 [2, 7]      * |    15  0

---
seqlengths:
   ch1  chMT   ch2
 50000   800    NA

> grl3

GRangesList of length 2:
[[1]]
GRanges 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 with 1 range and 2 metadata columns:
   seqnames ranges strand | score GC
        ch2 [2, 7]      * |    15  0

---
seqlengths:
   ch1  chMT   ch2
 50000   800    NA
```

```
> psetdiff(grl2, grl3)

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

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

---
seqlengths:
   ch1  chMT   ch2
 50000   800    NA
```

- Introduction
- `IRanges` objects
    - Constructor and accessors
    - Vector operations
    - Range-based operations
- `GRanges` objects
    - Constructor and accessors
    - Vector operations
    - Range-based operations
    - Splitting a `GRanges` object
- `GRangesList` objects
    - Constructor and accessors
    - Vector operations
    - List operations
    - Range-based operations
- `GAlignments` objects
    - Constructor and accessors
    - Coercion to `GRanges` or `GRangesList`
- `GAlignmentPairs` objects
    - Constructor and accessors
    - Coercion to `GRangesList`
- Advanced operations
    - Coverage and slicing
    - Finding/counting overlaps
- Resources

The purpose of the `GAlignments` container is...

... to store a set of genomic alignments (aligned reads, typically).

The alignments can be loaded from a BAM file with `readGAlignments()`. By default, only the following information is loaded for each alignment:

- RNAME field: name of the reference sequence to which the query is aligned.
- strand bit (from FLAG field): strand in the reference sequence to which the query is aligned.
- CIGAR field: a string in the "Extended CIGAR format" describing the "gemoetry" of the alignment (i.e. locations of insertions, deletions and gaps). See the SAM Spec for the details.
- POS field: **1-based** position of the leftmost mapped base.

In particular, the query sequences (SEQ) and qualities (QUAL) are not loaded by default.

Supported operations

- *Vector* operations: **partially supported** (no splitting/relisting, no comparing or ordering)
- *List* operations: **NO**
- *Ranges* operations: only `narrow()` and `qnarrow()` (`GAlignments` specific) are supported
- *Coercion methods*: to `GRanges` or `GRangesList`

Introduction

`IRanges` objects
    Constructor and accessors
    Vector operations
    Range-based operations

`GRanges` objects
    Constructor and accessors
    Vector operations
    Range-based operations
    Splitting a `GRanges` object

`GRangesList` objects
    Constructor and accessors
    Vector operations
    List operations
    Range-based operations

`GAlignments` objects
    Constructor and accessors
    Coercion to `GRanges` or `GRangesList`

`GAlignmentPairs` objects
    Constructor and accessors
    Coercion to `GRangesList`

Advanced operations
    Coverage and slicing
    Finding/counting overlaps

Resources

GAlignments constructor

Typically not used directly!

```
> gal0 <- GAlignments(seqnames=Rle(c("ch1", "ch2"), c(3, 1)),
+                     pos=1L + 10L*0:3,
+                     cigar=c("36M", "20M3D16M", "20M703N16M", "14M2I20M"),
+                     strand=strand(c("+", "-", "-", "+")))
> gal0

GAlignments with 4 alignments and 0 metadata columns:
      seqnames strand       cigar    qwidth     start       end     width      ngap
         <Rle>  <Rle> <character> <integer> <integer> <integer> <integer> <integer>
  [1]      ch1      +         36M        36         1        36        36         0
  [2]      ch1      -    20M3D16M        36        11        49        39         0
  [3]      ch1      -  20M703N16M        36        21       759       739         1
  [4]      ch2      +    14M2I20M        36        31        64        34         0
  ---
  seqlengths:
   ch1 ch2
    NA  NA
```

An N in the cigar indicates a gap (!= deletion).

readGAlignments()

```
> library(pasillaBamSubset)
> U1gal <- readGAlignments(untreated1_chr4())
> length(U1gal)

[1] 204355

> head(U1gal)

GAlignments with 6 alignments and 0 metadata columns:
      seqnames strand       cigar    qwidth     start       end     width      ngap
         <Rle>  <Rle> <character> <integer> <integer> <integer> <integer> <integer>
  [1]     chr4      -         75M        75       892       966        75         0
  [2]     chr4      -         75M        75       919       993        75         0
  [3]     chr4      +         75M        75       924       998        75         0
  [4]     chr4      +         75M        75       936      1010        75         0
  [5]     chr4      +         75M        75       949      1023        75         0
  [6]     chr4      -         75M        75       967      1041        75         0
  ---
  seqlengths:
      chr2L    chr2R    chr3L    chr3R     chr4     chrM     chrX  chrYHet
   23011544 21146708 24543557 27905053  1351857    19517 22422827   347038
```

GAlignments accessors

```
> seqnames(U1gal)

factor-Rle of length 204355 with 1 run
  Lengths: 204355
  Values :   chr4
Levels(8): chr2L chr2R chr3L chr3R chr4 chrM chrX chrYHet

> table(as.factor(seqnames(U1gal)))

  chr2L   chr2R   chr3L   chr3R    chr4    chrM    chrX chrYHet
      0       0       0       0  204355       0       0       0

> strand(U1gal)

factor-Rle of length 204355 with 53763 runs
  Lengths:  2  3  3  1  2  2  4  1  4  2  2  1 ... 13  1 13  1 17  1 20  3  3 40  2
  Values :  -  +  -  +  -  +  -  +  -  +  -  + ...  -  +  -  +  -  +  -  +  -  +  -
Levels(3): + - *

> table(as.factor(strand(U1gal)))

     +      -      *
102101 102254      0

> head(cigar(U1gal))

[1] "75M" "75M" "75M" "75M" "75M" "75M"

> head(qwidth(U1gal))

[1] 75 75 75 75 75 75

> table(qwidth(U1gal))

    75
204355
```

GAlignments accessors (continued)

```
> head(start(U1gal))

[1] 892 919 924 936 949 967

> head(end(U1gal))

[1]  966  993  998 1010 1023 1041

> head(width(U1gal))

[1] 75 75 75 75 75 75

> head(ngap(U1gal))

[1] 0 0 0 0 0 0

> table(ngap(U1gal))

     0      1      2
184039  20169    147
```

```
> mcols(U1gal)

DataFrame with 204355 rows and 0 columns

> seqinfo(U1gal)

Seqinfo of length 8
seqnames seqlengths isCircular genome
chr2L      23011544         NA   <NA>
chr2R      21146708         NA   <NA>
chr3L      24543557         NA   <NA>
chr3R      27905053         NA   <NA>
chr4        1351857         NA   <NA>
chrM          19517         NA   <NA>
chrX       22422827         NA   <NA>
chrYHet      347038         NA   <NA>
```

Loading additional information from the BAM file

```
> param <- ScanBamParam(what=c("flag", "mapq"), tag=c("NH", "NM"))
> U1gal <- readGAlignments(untreated1_chr4(),
+                          use.names=TRUE, param=param)
> U1gal[1:5]

GAlignments with 5 alignments and 4 metadata columns:
                   seqnames strand       cigar    qwidth     start       end
                      <Rle>  <Rle> <character> <integer> <integer> <integer>
  SRR031729.3941844     chr4      -         75M        75       892       966
  SRR031728.3674563     chr4      -         75M        75       919       993
  SRR031729.8532600     chr4      +         75M        75       924       998
  SRR031729.2779333     chr4      +         75M        75       936      1010
  SRR031728.2826481     chr4      +         75M        75       949      1023
                       width      ngap |      flag      mapq        NH        NM
                   <integer> <integer> | <integer> <integer> <integer> <integer>
  SRR031729.3941844        75         0 |        16      <NA>         1         1
  SRR031728.3674563        75         0 |        16      <NA>         1         3
  SRR031729.8532600        75         0 |         0         3         2         2
  SRR031729.2779333        75         0 |         0         3         2         1
  SRR031728.2826481        75         0 |         0         1         3         2
  ---
  seqlengths:
       chr2L     chr2R     chr3L     chr3R      chr4      chrM      chrX   chrYHet
    23011544  21146708  24543557  27905053   1351857     19517  22422827    347038

> any(duplicated(names(U1gal)))

[1] TRUE
```

- Introduction
- `IRanges` objects
  - Constructor and accessors
  - Vector operations
  - Range-based operations
- `GRanges` objects
  - Constructor and accessors
  - Vector operations
  - Range-based operations
  - Splitting a `GRanges` object
- `GRangesList` objects
  - Constructor and accessors
  - Vector operations
  - List operations
  - Range-based operations
- `GAlignments` objects
  - Constructor and accessors
  - Coercion to `GRanges` or `GRangesList`
- `GAlignmentPairs` objects
  - Constructor and accessors
  - Coercion to `GRangesList`
- Advanced operations
  - Coverage and slicing
  - Finding/counting overlaps
- Resources

From `GAlignments` to `GRanges`

Gaps are ignored, that is, each alignment is converted into a *single* genomic range defined by the *start* and *end* of the alignment.

```
> as(U1gal, "GRanges")

GRanges with 204355 ranges and 0 metadata columns:
                    seqnames             ranges strand
                       <Rle>          <IRanges>  <Rle>
  SRR031729.3941844     chr4       [892,  966]      -
  SRR031728.3674563     chr4       [919,  993]      -
  SRR031729.8532600     chr4       [924,  998]      +
  SRR031729.2779333     chr4       [936, 1010]      +
  SRR031728.2826481     chr4       [949, 1023]      +
                ...      ...                ...    ...
  SRR031728.1789947     chr4 [1348268, 1348342]      +
  SRR031728.4528492     chr4 [1348268, 1348342]      +
  SRR031729.5150849     chr4 [1348268, 1348342]      +
  SRR031729.9070096     chr4 [1348449, 1348523]      -
  SRR031729.9070096     chr4 [1350124, 1350198]      -
  ---
  seqlengths:
      chr2L    chr2R    chr3L    chr3R     chr4     chrM     chrX  chrYHet
   23011544 21146708 24543557 27905053  1351857    19517 22422827   347038
```

From GAlignments to GRangesList

Gaps are NOT ignored, that is, each alignment is converted into one or more genomic ranges (one more range than the number of gaps in the alignment).

```
> U1grl <- as(U1gal, "GRangesList")
> U1grl

GRangesList of length 204355:
$SRR031729.3941844
GRanges with 1 range and 0 metadata columns:
      seqnames     ranges strand
         <Rle>  <IRanges>  <Rle>
  [1]     chr4 [892, 966]      -

$SRR031728.3674563
GRanges with 1 range and 0 metadata columns:
      seqnames     ranges strand
  [1]     chr4 [919, 993]      -

$SRR031729.8532600
GRanges with 1 range and 0 metadata columns:
      seqnames     ranges strand
  [1]     chr4 [924, 998]      +


...
<204352 more elements>
---
seqlengths:
    chr2L    chr2R    chr3L    chr3R     chr4     chrM     chrX  chrYHet
 23011544 21146708 24543557 27905053  1351857    19517 22422827   347038
```

From `GAlignments` to `GRangesList` (continued)

One more range than the number of gaps in the alignment:

```
> all(elementLengths(U1grl) == ngap(U1gal) + 1)
[1] TRUE
```

- Introduction
- `IRanges` objects
  - Constructor and accessors
  - Vector operations
  - Range-based operations
- `GRanges` objects
  - Constructor and accessors
  - Vector operations
  - Range-based operations
  - Splitting a `GRanges` object
- `GRangesList` objects
  - Constructor and accessors
  - Vector operations
  - List operations
  - Range-based operations
- `GAlignments` objects
  - Constructor and accessors
  - Coercion to `GRanges` or `GRangesList`
- `GAlignmentPairs` objects
  - Constructor and accessors
  - Coercion to `GRangesList`
- Advanced operations
  - Coverage and slicing
  - Finding/counting overlaps
- Resources

The purpose of the `GAlignmentPairs` container is...

... to store a set of aligned *paired-end* reads.

- Implemented on top of the `GAlignments` class.
- The alignments can be loaded from a BAM file with `readGAlignmentPairs()`.
- `first(x)`, `last(x)`: extract the *first* and *last* ends in 2 separate `GAlignments` objects of the same length.

Supported operations

- *Vector* operations: **partially supported** (no splitting/relisting, no comparing or ordering)
- *List* operations: **YES**
- *Ranges* operations: **NO**
- *Coercion methods*: to `GRanges` or `GRangesList`

Introduction

`IRanges` objects
- Constructor and accessors
- Vector operations
- Range-based operations

`GRanges` objects
- Constructor and accessors
- Vector operations
- Range-based operations
- Splitting a `GRanges` object

`GRangesList` objects
- Constructor and accessors
- Vector operations
- List operations
- Range-based operations

`GAlignments` objects
- Constructor and accessors
- Coercion to `GRanges` or `GRangesList`

`GAlignmentPairs` objects
- Constructor and accessors
- Coercion to `GRangesList`

Advanced operations
- Coverage and slicing
- Finding/counting overlaps

Resources

`readGAlignmentPairs()`

```
> library(pasillaBamSubset)
> U3galp <- readGAlignmentPairs(untreated3_chr4())
> length(U3galp)

[1] 75346

> head(U3galp)

GAlignmentPairs with 6 alignment pairs and 0 metadata columns:
    seqnames strand :       ranges --       ranges
       <Rle>  <Rle> :    <IRanges> --    <IRanges>
[1]     chr4      + : [169,  205] -- [ 326,  362]
[2]     chr4      + : [943,  979] -- [1086, 1122]
[3]     chr4      + : [944,  980] -- [1119, 1155]
[4]     chr4      + : [946,  982] -- [ 986, 1022]
[5]     chr4      + : [966, 1002] -- [1108, 1144]
[6]     chr4      + : [966, 1002] -- [1114, 1150]
---
seqlengths:
    chr2L    chr2R    chr3L    chr3R     chr4     chrM     chrX  chrYHet
 23011544 21146708 24543557 27905053  1351857    19517 22422827   347038
```

GAlignmentPairs accessors

```
> head(first(U3galp))

GAlignments with 6 alignments and 0 metadata columns:
      seqnames strand       cigar    qwidth     start       end     width      ngap
         <Rle>  <Rle> <character> <integer> <integer> <integer> <integer> <integer>
  [1]     chr4      +         37M        37       169       205        37         0
  [2]     chr4      +         37M        37       943       979        37         0
  [3]     chr4      +         37M        37       944       980        37         0
  [4]     chr4      +         37M        37       946       982        37         0
  [5]     chr4      +         37M        37       966      1002        37         0
  [6]     chr4      +         37M        37       966      1002        37         0
  ---
  seqlengths:
      chr2L    chr2R    chr3L    chr3R     chr4     chrM     chrX  chrYHet
   23011544 21146708 24543557 27905053  1351857    19517 22422827   347038

> head(last(U3galp))

GAlignments with 6 alignments and 0 metadata columns:
      seqnames strand       cigar    qwidth     start       end     width      ngap
         <Rle>  <Rle> <character> <integer> <integer> <integer> <integer> <integer>
  [1]     chr4      -         37M        37       326       362        37         0
  [2]     chr4      -         37M        37      1086      1122        37         0
  [3]     chr4      -         37M        37      1119      1155        37         0
  [4]     chr4      -         37M        37       986      1022        37         0
  [5]     chr4      -         37M        37      1108      1144        37         0
  [6]     chr4      -         37M        37      1114      1150        37         0
  ---
  seqlengths:
      chr2L    chr2R    chr3L    chr3R     chr4     chrM     chrX  chrYHet
   23011544 21146708 24543557 27905053  1351857    19517 22422827   347038
```

Currently, `readGAlignmentPairs()` drops pairs where the *first* and *last* ends have incompatible sequence names and/or strands (a rare situation).

GAlignmentPairs accessors (continued)

```
> seqnames(U3galp)

factor-Rle of length 75346 with 1 run
  Lengths: 75346
  Values :  chr4
Levels(8): chr2L chr2R chr3L chr3R chr4 chrM chrX chrYHet

> strand(U3galp)

factor-Rle of length 75346 with 18999 runs
  Lengths:  6  6  3  1  6  1  1  2  2  1  1  3 ...  3  2  3  1  2  1  5  6  2  7  3
  Values :  +  -  +  -  +  -  +  -  +  -  +  - ...  +  -  +  -  +  -  +  -  +  -  +
Levels(3): + - *

> head(ngap(U3galp))

[1] 0 0 0 0 0 0

> table(ngap(U3galp))

    0     1     2
72949  2291   106
```

- Introduction
- `IRanges` objects
  - Constructor and accessors
  - Vector operations
  - Range-based operations
- `GRanges` objects
  - Constructor and accessors
  - Vector operations
  - Range-based operations
  - Splitting a `GRanges` object
- `GRangesList` objects
  - Constructor and accessors
  - Vector operations
  - List operations
  - Range-based operations
- `GAlignments` objects
  - Constructor and accessors
  - Coercion to `GRanges` or `GRangesList`
- `GAlignmentPairs` objects
  - Constructor and accessors
  - Coercion to `GRangesList`
- Advanced operations
  - Coverage and slicing
  - Finding/counting overlaps
- Resources

From GAlignmentPairs to GRangesList

```
> U3grl <- as(U3galp, "GRangesList")
> U3grl

GRangesList of length 75346:
[[1]]
GRanges with 2 ranges and 0 metadata columns:
      seqnames     ranges strand
         <Rle>  <IRanges>  <Rle>
  [1]     chr4 [169, 205]      +
  [2]     chr4 [326, 362]      +

[[2]]
GRanges with 2 ranges and 0 metadata columns:
      seqnames       ranges strand
  [1]     chr4 [ 943,  979]      +
  [2]     chr4 [1086, 1122]      +

[[3]]
GRanges with 2 ranges and 0 metadata columns:
      seqnames       ranges strand
  [1]     chr4 [ 944,  980]      +
  [2]     chr4 [1119, 1155]      +

...
<75343 more elements>
---
seqlengths:
     chr2L    chr2R    chr3L    chr3R     chr4     chrM     chrX  chrYHet
  23011544 21146708 24543557 27905053  1351857    19517 22422827   347038
```

From GAlignmentPairs to GRangesList (continued)

```
> U3grl[ngap(U3galp) != 0]

GRangesList of length 2397:
[[1]]
GRanges with 3 ranges and 0 metadata columns:
      seqnames         ranges strand
         <Rle>      <IRanges>  <Rle>
  [1]     chr4 [74403, 74435]      -
  [2]     chr4 [77050, 77053]      -
  [3]     chr4 [13711, 13747]      -

[[2]]
GRanges with 3 ranges and 0 metadata columns:
      seqnames         ranges strand
  [1]     chr4 [56932, 56968]      +
  [2]     chr4 [57072, 57083]      +
  [3]     chr4 [57142, 57166]      +

[[3]]
GRanges with 3 ranges and 0 metadata columns:
      seqnames         ranges strand
  [1]     chr4 [56932, 56968]      +
  [2]     chr4 [57065, 57083]      +
  [3]     chr4 [57142, 57159]      +

...
<2394 more elements>
---
seqlengths:
     chr2L    chr2R    chr3L    chr3R     chr4     chrM     chrX  chrYHet
  23011544 21146708 24543557 27905053  1351857    19517 22422827   347038
```

- Introduction
- `IRanges` objects
  - Constructor and accessors
  - Vector operations
  - Range-based operations
- `GRanges` objects
  - Constructor and accessors
  - Vector operations
  - Range-based operations
  - Splitting a `GRanges` object
- `GRangesList` objects
  - Constructor and accessors
  - Vector operations
  - List operations
  - Range-based operations
- `GAlignments` objects
  - Constructor and accessors
  - Coercion to `GRanges` or `GRangesList`
- `GAlignmentPairs` objects
  - Constructor and accessors
  - Coercion to `GRangesList`
- Advanced operations
  - Coverage and slicing
  - Finding/counting overlaps
- Resources

- Introduction
- IRanges objects
  - Constructor and accessors
  - Vector operations
  - Range-based operations
- GRanges objects
  - Constructor and accessors
  - Vector operations
  - Range-based operations
  - Splitting a GRanges object
- GRangesList objects
  - Constructor and accessors
  - Vector operations
  - List operations
  - Range-based operations
- GAlignments objects
  - Constructor and accessors
  - Coercion to GRanges or GRangesList
- GAlignmentPairs objects
  - Constructor and accessors
  - Coercion to GRangesList
- Advanced operations
  - Coverage and slicing
  - Finding/counting overlaps
- Resources

Coverage

```
> U1cvg <- coverage(U1grl)
> U1cvg

RleList of length 8
$chr2L
integer-Rle of length 23011544 with 1 run
  Lengths: 23011544
  Values :        0

$chr2R
integer-Rle of length 21146708 with 1 run
  Lengths: 21146708
  Values :        0

$chr3L
integer-Rle of length 24543557 with 1 run
  Lengths: 24543557
  Values :        0

$chr3R
integer-Rle of length 27905053 with 1 run
  Lengths: 27905053
  Values :        0

$chr4
integer-Rle of length 1351857 with 122061 runs
  Lengths:  891   27    5   12   13   45    5 ...    3  106   75 1600   75 1659
  Values :    0    1    2    3    4    5    4 ...    6    0    1    0    1    0

...
<3 more elements>
```

Coverage (continued)

```
> mean(U1cvg)
   chr2L    chr2R    chr3L    chr3R     chr4     chrM     chrX  chrYHet
 0.00000  0.00000  0.00000  0.00000 11.33746  0.00000  0.00000  0.00000
> max(U1cvg)
  chr2L   chr2R   chr3L   chr3R    chr4    chrM    chrX chrYHet
      0       0       0       0    5627       0       0       0
```

Slicing the coverage

```
> U1sl <- slice(U1cvg, lower=10)
> U1sl

RleViewsList of length 8
names(8): chr2L chr2R chr3L chr3R chr4 chrM chrX chrYHet

> elementLengths(U1sl)

  chr2L   chr2R   chr3L   chr3R    chr4    chrM    chrX chrYHet
      0       0       0       0    1183       0       0       0

> head(U1sl$chr4)

Views on a 1351857-length Rle subject

views:
    start  end width
[1]  4936 5077   142 [11 12 12 13 13 14 16 16 17 18 18 18 18 19 19 19 19 19 ...]
[2]  5211 5245    35 [10 10 10 10 10 10 10 10 10 10 10 10 10 12 12 13 13 13 ...]
[3]  5334 5337     4 [10 10 10 10]
[4]  5736 5744     9 [10 10 10 10 10 10 10 10 10]
[5]  5752 5754     3 [10 10 10]
[6]  5756 5882   127 [10 11 11 11 11 11 11 11 11 11 11 11 11 11 11 12 12 13 ...]

> head(mean(U1sl$chr4))

[1] 23.88028 11.60000 10.00000 10.00000 10.00000 25.65354

> head(max(U1sl$chr4))

[1] 39 13 10 10 10 38
```

- Introduction
- `IRanges` objects
  - Constructor and accessors
  - Vector operations
  - Range-based operations
- `GRanges` objects
  - Constructor and accessors
  - Vector operations
  - Range-based operations
  - Splitting a `GRanges` object
- `GRangesList` objects
  - Constructor and accessors
  - Vector operations
  - List operations
  - Range-based operations
- `GAlignments` objects
  - Constructor and accessors
  - Coercion to `GRanges` or `GRangesList`
- `GAlignmentPairs` objects
  - Constructor and accessors
  - Coercion to `GRangesList`
- Advanced operations
  - Coverage and slicing
  - Finding/counting overlaps
- Resources

Finding/counting overlaps

A typical use case: count the number of *hits* (a.k.a. *overlaps*) per transcript.

Typical input

- A BAM file with the aligned reads (*single-* or *paired-end*).
- Transcript annotations *for the same reference genome* that was used to align the reads.

Typical tools

- `readGAlignments()` or `readGAlignmentPairs()` to load the reads in a `GAlignments` or `GAlignmentPairs` object.
- A `TranscriptDb` object containing the transcript annotations.
- The `exonsBy()` extractor (defined in the *GenomicFeatures* package) to extract the exons ranges grouped by transcript from the `TranscriptDb` object. The exons ranges are returned in a `GRangesList` object with 1 top-level element per transcript.
- The `findOverlaps()` and/or `countOverlaps()` functions.

Load the transcripts

```
> library(TxDb.Dmelanogaster.UCSC.dm3.ensGene)
> txdb <- TxDb.Dmelanogaster.UCSC.dm3.ensGene
> exbytx <- exonsBy(txdb, by="tx", use.names=TRUE)
> exbytx

GRangesList of length 29173:
$FBtr0300689
GRanges with 2 ranges and 3 metadata columns:
      seqnames       ranges strand |   exon_id   exon_name exon_rank
         <Rle>    <IRanges>  <Rle> | <integer> <character> <integer>
  [1]    chr2L [7529, 8116]      + |         1        <NA>         1
  [2]    chr2L [8193, 9484]      + |         3        <NA>         2

$FBtr0300690
GRanges with 3 ranges and 3 metadata columns:
      seqnames       ranges strand | exon_id exon_name exon_rank
  [1]    chr2L [7529, 8116]      + |       1      <NA>         1
  [2]    chr2L [8193, 8589]      + |       2      <NA>         2
  [3]    chr2L [8668, 9484]      + |       5      <NA>         3

$FBtr0330654
GRanges with 2 ranges and 3 metadata columns:
      seqnames       ranges strand | exon_id exon_name exon_rank
  [1]    chr2L [7529, 8116]      + |       1      <NA>         1
  [2]    chr2L [8229, 9484]      + |       4      <NA>         2

...
<29170 more elements>
---
seqlengths:
```

Single-end overlaps

```
> U1txhits <- countOverlaps(exbytx, U1grl)
> length(U1txhits)

[1] 29173

> head(U1txhits)

FBtr0300689 FBtr0300690 FBtr0330654 FBtr0309810 FBtr0306539 FBtr0306536
          0           0           0           0           0           0

> sum(U1txhits)  # total nb of hits

[1] 284609

> head(sort(U1txhits, decreasing=TRUE))

FBtr0308296 FBtr0089175 FBtr0089176 FBtr0112904 FBtr0289951 FBtr0089243
      20399       20330       20330        6018        5982        5979
```

Rough counting!

- More than 1 alignment per read can be reported in the BAM file (sometimes the same read hits the same transcript many times).
- A hit is counted even if it's not *compatible* with the splicing of the transcript.

Paired-end overlaps

```
> U3txhits <- countOverlaps(exbytx, U3grl)
> length(U3txhits)

[1] 29173

> head(U3txhits)

FBtr0300689 FBtr0300690 FBtr0330654 FBtr0309810 FBtr0306539 FBtr0306536
          0           0           0           0           0           0

> sum(U3txhits)  # total nb of hits

[1] 106947

> head(sort(U3txhits, decreasing=TRUE))

FBtr0308296 FBtr0089175 FBtr0089176 FBtr0112904 FBtr0289951 FBtr0089243
       6806        6791        6791        2617        2610        2609
```

Note that exons that fall within the *inter-read* gap are NOT considered to overlap.

- Introduction
- `IRanges` objects
  - Constructor and accessors
  - Vector operations
  - Range-based operations
- `GRanges` objects
  - Constructor and accessors
  - Vector operations
  - Range-based operations
  - Splitting a `GRanges` object
- `GRangesList` objects
  - Constructor and accessors
  - Vector operations
  - List operations
  - Range-based operations
- `GAlignments` objects
  - Constructor and accessors
  - Coercion to `GRanges` or `GRangesList`
- `GAlignmentPairs` objects
  - Constructor and accessors
  - Coercion to `GRangesList`
- Advanced operations
  - Coverage and slicing
  - Finding/counting overlaps
- Resources

Resources

Resources

- Vignettes in the *GenomicRanges* package (`browseVignettes("GenomicRanges")`).
- `GRanges`, `GRangesList`, `GAlignments`, and `GAlignmentPairs` man pages in the *GenomicRanges* package.
- SAMtools website: `http://samtools.sourceforge.net/`
- *Bioconductor* mailing lists: `http://bioconductor.org/help/mailing-list/`

Where to look next

- `summarizeOverlaps()` function in the *GenomicRanges* package for counting overlaps between reads and genomic features, and resolve reads that overlap multiple features.

THANKS!