High-level S4 containers for HTS data

Hervé Pagès

Fred Hutchinson Cancer Research Center

27-28 February 2012

High-level vs low-level

High-level containers for HTS data covered in this presentation (all defined in the *GenomicRanges* package):

- *GRanges*
- *GRangesList*
- *GappedAlignments*

Other high-level containers for HTS data:

- *SummarizedExperiment* (*GenomicRanges* package)
- *ShortRead*, *AlignedRead* (*ShortRead* package)

100+ low-level containers. Most of them defined in the *IRanges* package. Most frequently seen:

- Defined in the *IRanges* package: *Rle*, *IRanges*, *CharacterList*, *IntegerList*, *RleList*, *RleViews*, *RleViewsList*, *IRangesList* (not covered in this presentation), *DataFrame*.
- Defined in the *Biostrings* package (not covered in this presentation): *DNAString*, *DNAStringSet*.

About the implementation

S4 classes (aka *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 `@` (aka *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: `elementMetadata()`, `elementLengths()`, `seqinfo()`, etc...
- Consistent display (`show` methods).

Basic operations

Vector operations:

- Single-bracket subsetting: `[`
- Combining: `c()`
- Comparing: `==`, `!=`, `duplicated()`, `unique()`
- Ordering: `<=`, `>=`, `<`, `>`, `order()`, `sort()`, `rank()`

List operations:

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

Basic operations (continued)

Ranges operations:

- `shift()`, `narrow()`, `resize()`, `flank()`
- `disjoin()`
- `range()`, `reduce()`, `gaps()`
- `union()`, `intersect()`, `setdiff()`
- `punion()`, `pintersect()`, `psetdiff()`, `pgap()`

Coercion methods: `as()` and all the *S3 forms* (`as.vector()`, `as.character()`, `as.factor()`, etc...)

Splitting: `split()`

Advanced operations

- *Coverage and slicing*: `coverage()` and `slice()`
- *Finding/counting overlaps*: `findOverlaps()` and `countOverlaps()`
- and more...

Rle objects

Rle: Run Length Encoding

Supported basic operations:

- *Vector operations*: YES
- *List operations*: NO
- *Ranges operations*: NO
- *Coercion methods*: YES (to atomic vector, factor, or *IRanges*)
- *Splitting*: YES (produces an *RleList* object)

Rle objects (continued)

```
> library(IRanges)
> set.seed(2012)
> rle1 <- Rle(sample(c(-0.9, 0), 20, replace=TRUE))
> rle1

'numeric' Rle of length 20 with 12 runs
  Lengths:    1    1    1    7    1    1    1    2    1    1    2    1
  Values : -0.9    0 -0.9    0 -0.9    0 -0.9    0 -0.9    0 -0.9    0

> runLength(rle1)

 [1] 1 1 1 7 1 1 1 2 1 1 2 1

> runValue(rle1)

 [1] -0.9  0.0 -0.9  0.0 -0.9  0.0 -0.9  0.0 -0.9  0.0 -0.9  0.0

> as.vector(rle1)

 [1] -0.9  0.0 -0.9  0.0  0.0  0.0  0.0  0.0  0.0  0.0 -0.9  0.0 -0.9  0.0  0.0
[16] -0.9  0.0 -0.9 -0.9  0.0

> rle1[c(TRUE, FALSE)]

'numeric' Rle of length 10 with 5 runs
  Lengths:    2    3    2    2    1
  Values : -0.9    0 -0.9    0 -0.9
```

Rle objects (continued)

```
> sort(rle1)

'numeric' Rle of length 20 with 2 runs
  Lengths:    7   13
  Values : -0.9    0

> rle1 * 50.1

'numeric' Rle of length 20 with 12 runs
  Lengths:      1      1      1      7      1 ...      1      1      2      1
  Values : -45.09      0 -45.09      0 -45.09 ... -45.09      0 -45.09      0

> sum(rle1)

[1] -6.3

> cumsum(rle1)

'numeric' Rle of length 20 with 7 runs
  Lengths:    2    8    2    3    2    1    2
  Values : -0.9 -1.8 -2.7 -3.6 -4.5 -5.4 -6.3

> cumsum(rle1) <= -4.2

'logical' Rle of length 20 with 2 runs
  Lengths:    15     5
  Values : FALSE  TRUE

> rle1[cumsum(rle1) <= -4.2]

'numeric' Rle of length 5 with 4 runs
  Lengths:    1    1    2    1
  Values : -0.9    0 -0.9    0
```

Rle objects (continued)

```
> rle2 <- Rle(c("ch1", "chMT", "ch1", "ch2", "chMT"), c(4, 2, 1, 5, 1))
> rle2

'character' Rle of length 13 with 5 runs
  Lengths:      4      2      1      5      1
  Values :  "ch1" "chMT"  "ch1"  "ch2" "chMT"

> as.vector(rle2)

 [1] "ch1"  "ch1"  "ch1"  "ch1"  "chMT" "chMT" "ch1"  "ch2"  "ch2"  "ch2"  "ch2"
[12] "ch2"  "chMT"

> c(rle2, c("chMT", "chX"))

'character' Rle of length 15 with 6 runs
  Lengths:      4      2      1      5      2      1
  Values :  "ch1" "chMT"  "ch1"  "ch2" "chMT"  "chX"
```

Rle objects (continued)

```
> runValue(rle2) <- factor(runValue(rle2))
> rle2

'factor' Rle of length 13 with 5 runs
  Lengths:    4    2    1    5    1
  Values :  ch1 chMT  ch1  ch2 chMT
Levels(3): ch1 ch2 chMT

> runValue(rle2)

[1] ch1  chMT ch1  ch2  chMT
Levels: ch1 ch2 chMT

> as.vector(rle2)

 [1] "ch1"  "ch1"  "ch1"  "ch1"  "chMT" "chMT" "ch1"  "ch2"  "ch2"  "ch2"  "ch2"
[12] "ch2"  "chMT"

> as.factor(rle2)

 [1] ch1  ch1  ch1  ch1  chMT chMT ch1  ch2  ch2  ch2  ch2  ch2  chMT
Levels: ch1 ch2 chMT
```

Rle objects (continued)

```
> rle1 == 0

'logical' Rle of length 20 with 12 runs
  Lengths:     1     1     1     7     1     1     1     2     1     1     2     1
  Values : FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE

> as(rle1 == 0, "IRanges")

IRanges of length 6
    start end width
[1]     2   2     1
[2]     4  10     7
[3]    12  12     1
[4]    14  15     2
[5]    17  17     1
[6]    20  20     1
```

The purpose of the IRanges container is...

... to store a set of *integer ranges* (aka *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 where *end* = *start* - 1.

Supported basic operations:

- *Vector operations*: YES
- *List operations*: YES (not covered in this presentation)
- *Ranges operations*: YES
- *Coercion methods*: YES (from logical or integer vector to *IRanges*)
- *Splitting*: YES (produces an *IRangesList* object)

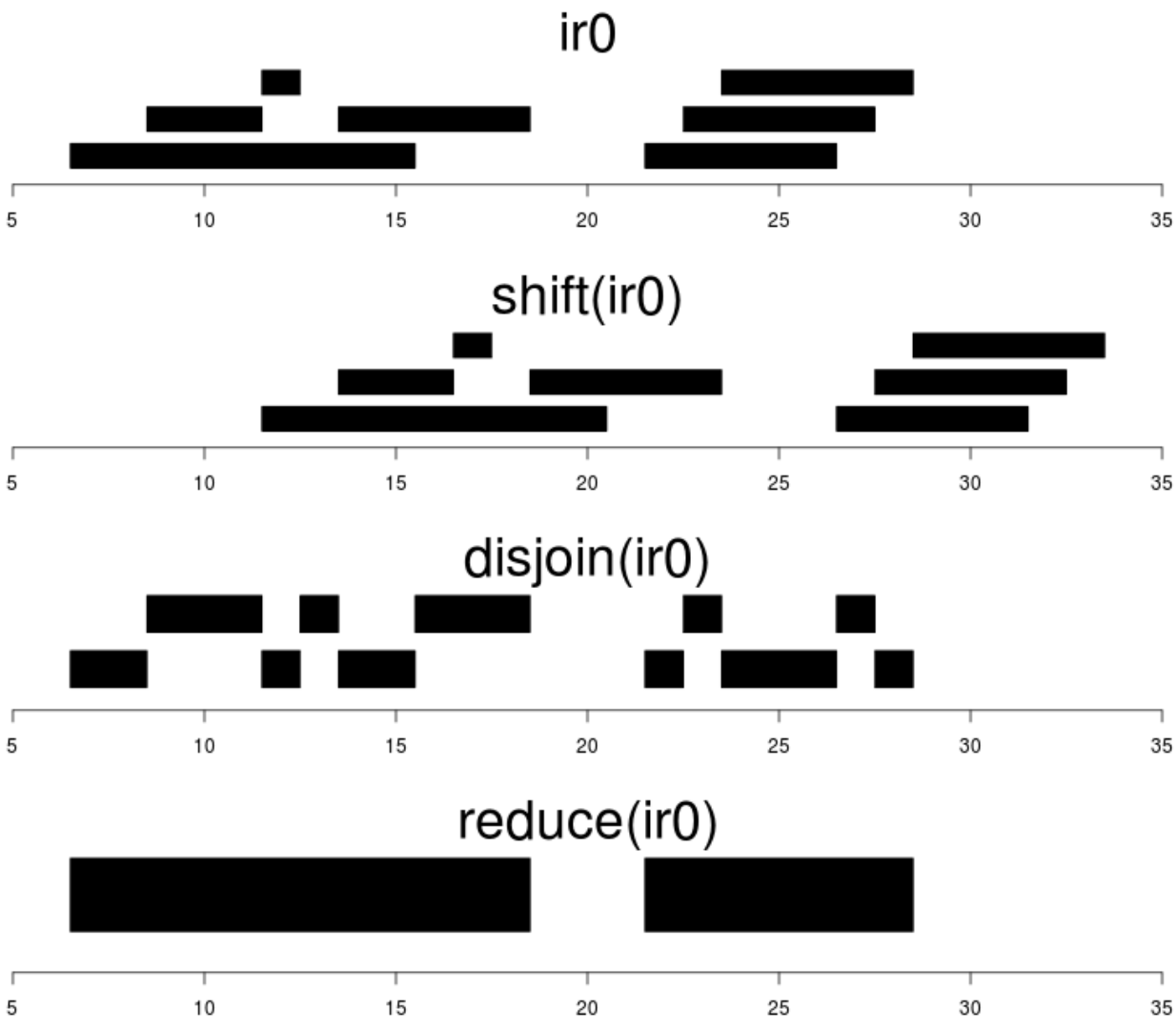
IRanges objects (continued)

```
> 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 objects (continued)

```
> ir1[-2]

IRanges of length 3
    start end width
[1]    12  15     4
[2]    12  15     4
[3]    12  14     3

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

IRanges of length 5
    start end width
[1]    12  15     4
[2]    -9   0    10
[3]    12  15     4
[4]    12  14     3
[5]   -10   0    11

> duplicated(ir2)

[1] FALSE FALSE  TRUE FALSE FALSE

> sort(ir2)

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

Ranges operations



IRanges objects (continued)

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

IRanges of length 4
    start end width
[1]     0   3     4
[2]     0   9    10
[3]     0   3     4
[4]     0   2     3

> flank(ir1, 10, start=FALSE)

IRanges of length 4
    start end width
[1]    16  25    10
[2]     1  10    10
[3]    16  25    10
[4]    15  24    10

> 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
```

IRanges objects (continued)

```
> 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
```

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
[1]    12  15     4
[2]    -9   0    10
[3]    12  15     4
[4]    12  14     3
[5]   -10   0    11

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

IRanges of length 5
    start end width
[1]    12  15     4
[2]     4   3     0
[3]    12  14     3
[4]    12  13     2
[5]     1   0     0
```

IRanges objects (continued)

```
> 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(which(ok), "IRanges")  # from integer vector to IRanges

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

> rle2[ir4]  # IRanges subscript

'factor' Rle of length 4 with 3 runs
  Lengths:    2    1    1
  Values :  ch1 chMT  ch2
Levels(3): ch1 ch2 chMT
```

DataFrame objects

DataFrame: An S4 version of `data.frame` that can hold almost anything it its columns.

Supported operations:

- All the `data.frame` operations. Just manipulate a *DataFrame* as a `data.frame`!
- *Coercion methods*: from almost anything to *DataFrame*, and from *DataFrame* to `data.frame`.
- *Splitting*: YES (produces a *SplitDataFrameList* object)

```
> library(Biostrings)
> dna <- DNAStringSet(c("AAA", "TGGATT", "CATTNGAGC", "TAATAG"))
> af <- alphabetFrequency(dna, baseOnly=TRUE)
> df <- DataFrame(dna, af)
> df
```

```
DataFrame with 4 rows and 6 columns
              dna         A         C         G         T     other
  <DNAStringSet> <integer> <integer> <integer> <integer> <integer>
1            AAA         3         0         0         0         0
2         TGGATT         1         0         2         3         0
3      CATTNGAGC         2         2         2         2         1
4         TAATAG         3         0         1         2         0
```

```
> df$G
```

```
[1] 0 2 2 1
```

DataFrame objects (continued)

```
> df$cds_id <- paste("CDS", 1:4, sep="")
> df$cds_range <- successiveIRanges(width(dna), from=51)
> df
```

```
DataFrame with 4 rows and 8 columns
              dna         A         C         G         T     other      cds_id
  <DNAStringSet> <integer> <integer> <integer> <integer> <integer> <character>
1            AAA         3         0         0         0         0        CDS1
2         TGGATT         1         0         2         3         0        CDS2
3      CATTNGAGC         2         2         2         2         1        CDS3
4         TAATAG         3         0         1         2         0        CDS4
  cds_range
  <IRanges>
1  [51, 53]
2  [54, 59]
3  [60, 68]
4  [69, 74]
```

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

```
        dna A C G T other cds_id cds_range.start cds_range.end cds_range.width
1       AAA 3 0 0 0     0   CDS1              51            53               3
2    TGGATT 1 0 2 3     0   CDS2              54            59               6
3 CATTNGAGC 2 2 2 2     1   CDS3              60            68               9
4    TAATAG 3 0 1 2     0   CDS4              69            74               6
```

CharacterList objects

An S4 virtual class for representing a list of character vectors.

Exists in 2 flavors (i.e. 2 different internal representations):

- *CompressedCharacterList*
- *SimpleCharacterList*

```
> ccl <- CharacterList(one=c("aaa", "bb", "c"),
+                      two=c("dd", "e", "fff", "gggg"))
> ccl

CompressedCharacterList of length 2
[["one"]] aaa bb c
[["two"]] dd e fff gggg

> length(ccl)

[1] 2

> as.list(ccl)

$one
[1] "aaa" "bb"  "c"

$two
[1] "dd"   "e"    "fff"  "gggg"

> ccl[[2]]

[1] "dd"   "e"    "fff"  "gggg"
```

CharacterList objects (continued)

```
> toupper(ccl)

CompressedCharacterList of length 2
[["one"]] AAA BB C
[["two"]] DD E FFF GGGG

> elementLengths(ccl)

one two
  3   4

> unlist(ccl)  # insane! will be changed soon...

   one   one1   one2    two   two1   two2   two3
 "aaa"   "bb"    "c"   "dd"    "e"  "fff" "gggg"

> unlist(ccl, use.names=FALSE)

[1] "aaa"  "bb"   "c"    "dd"   "e"    "fff"  "gggg"
```

IntegerList objects

An S4 virtual class for representing a list of integer vectors.

Exists in 2 flavors (i.e. 2 different internal representations):

- *CompressedIntegerList*
- *SimpleIntegerList*

```
> cil <- IntegerList(6:-2, 5, integer(0), 14:21)
> cil

CompressedIntegerList of length 4
[[1]] 6 5 4 3 2 1 0 -1 -2
[[2]] 5
[[3]] integer(0)
[[4]] 14 15 16 17 18 19 20 21

> cil * cil

CompressedIntegerList of length 4
[[1]] 36 25 16 9 4 1 0 1 4
[[2]] 25
[[3]] integer(0)
[[4]] 196 225 256 289 324 361 400 441
```

IntegerList objects (continued)

2 different ways to obtain the same result:

```
> cil * 100L - 2L
```

```
CompressedIntegerList of length 4
[[1]] 598 498 398 298 198 98 -2 -102 -202
[[2]] 498
[[3]] integer(0)
[[4]] 1398 1498 1598 1698 1798 1898 1998 2098
```

```
> relist(unlist(cil) * 100L - 2L, cil)
```

```
CompressedIntegerList of length 4
[[1]] 598 498 398 298 198 98 -2 -102 -202
[[2]] 498
[[3]] integer(0)
[[4]] 1398 1498 1598 1698 1798 1898 1998 2098
```

The above trick would not work here!

```
> cumsum(cil)
```

```
CompressedNumericList of length 4
[[1]] 6 11 15 18 20 21 21 20 18
[[2]] 5
[[3]] numeric(0)
[[4]] 14 29 45 62 80 99 119 140
```

RleList, RleViews and RleViewsList objects

Typically seen when doing *Coverage and slicing*.

RleList: An S4 virtual class for representing a list of *Rle* objects. Exists in 2 flavors (i.e. 2 different internal representations):

- *CompressedRleList*
- *SimpleRleList*

RleViews: An S4 class for representing a set of *views* (i.e. ranges) defined on an *Rle subject*.

RleViewsList: An S4 virtual class for representing a list of *RleViews* objects. Exists only in 1 flavor: *SimpleRleViewsList*.

The purpose of the GRanges container is...

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

- Like for *IRanges* objects, each range can be defined by a *start* and an *end* value.
- *start* and *end* are both 1-based positions relative to the 5' end of the plus strand of the chromosome (aka *reference sequence*), even when the range is on the minus strand.
- The *start* is the leftmost position and the *end* is the rightmost, even when the range is on the minus strand.
- Each range is assigned a chromosome name and a strand.

Supported basic operations:

- *Vector operations*: YES
- *List operations*: NO
- *Ranges operations*: YES
- *Coercion methods*: to *RangedData* or *IRangesList* (both not covered in this presentation)
- *Splitting*: YES (produces a *GRangesList* object)

GRanges constructor

```
> library(GenomicRanges)
> gr1 <- GRanges(seqnames=rep(c("ch1", "chMT"), c(2, 4)),
+                ranges=IRanges(start=16:21, end=20),
+                strand=rep(c("+", "-", "*"), 2))
> gr1
GRanges with 6 ranges and 0 elementMetadata cols:
      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"
> elementMetadata(gr1) <- DataFrame(score=11:16, GC=seq(1, 0, length=6))
> elementMetadata(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 elementMetadata cols:
    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
```

Vector operations on GRanges objects

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

GRanges with 2 ranges and 2 elementMetadata cols:
    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 elementMetadata cols:
    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 elementMetadata cols:
    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 elementMetadata cols:
    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 elementMetadata cols:
    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
  6      ch2  [ 2,  7]      * |        15         0
  7      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 elementMetadata cols:
    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
  6      ch2  [ 1,  6]      * |        14       0.2
  7      ch2  [ 2,  7]      * |        15         0
  8      ch2  [ 2,  7]      * |        13       0.4
  ---
  seqlengths:
     ch1  chMT   ch2
   50000   800    NA
```

Ranges operations on GRanges objects

```
> gr2

GRanges with 3 ranges and 2 elementMetadata cols:
      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 elementMetadata cols:
      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 elementMetadata cols:
      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
```

Ranges operations on GRanges objects (continued)

```
> gr1
```

```
GRanges with 5 ranges and 2 elementMetadata cols:
    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 elementMetadata cols:
    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
```

Ranges operations on GRanges objects (continued)

```
> gr1
```

```
GRanges with 5 ranges and 2 elementMetadata cols:
    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 elementMetadata cols:
    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
```

Ranges 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 elementMetadata cols:
    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 [  120,   236]      + |        16         0
  ---
  seqlengths:
     ch1  chMT
   50000   800

> range(gr3)
GRanges with 3 ranges and 0 elementMetadata cols:
      seqnames      ranges strand
         <Rle>   <IRanges>  <Rle>
  [1]      ch1 [17, 35020]      -
  [2]     chMT [18,   236]      +
  [3]     chMT [19,    20]      -
  ---
  seqlengths:
     ch1  chMT
   50000   800
```

Ranges operations on GRanges objects (continued)

```
> gr3
```

```
GRanges with 5 ranges and 2 elementMetadata cols:
    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 [  120,   236]      + |        16         0
  ---
  seqlengths:
     ch1  chMT
   50000   800
```

```
> disjoin(gr3)
```

```
GRanges with 6 ranges and 0 elementMetadata cols:
      seqnames          ranges strand
         <Rle>       <IRanges>  <Rle>
  [1]      ch1 [   17,    20]      -
  [2]      ch1 [35016, 35020]      -
  [3]     chMT [   18,   119]      +
  [4]     chMT [  120,   134]      +
  [5]     chMT [  135,   236]      +
  [6]     chMT [   19,    20]      -
  ---
  seqlengths:
     ch1  chMT
   50000   800
```

Ranges operations on GRanges objects (continued)

```
> gr3

GRanges with 5 ranges and 2 elementMetadata cols:
    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 [  120,   236]      + |        16         0
  ---
  seqlengths:
     ch1  chMT
   50000   800

> reduce(gr3)

GRanges with 4 ranges and 0 elementMetadata cols:
      seqnames         ranges strand
         <Rle>      <IRanges>  <Rle>
  [1]      ch1 [   17,    20]      -
  [2]      ch1 [35016, 35020]      -
  [3]     chMT [   18,   236]      +
  [4]     chMT [   19,    20]      -
  ---
  seqlengths:
     ch1  chMT
   50000   800
```

Ranges operations on GRanges objects (continued)

```
> gr3

GRanges with 5 ranges and 2 elementMetadata cols:
    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 [  120,   236]      + |        16         0
  ---
  seqlengths:
     ch1  chMT
   50000   800

> gaps(gr3)

GRanges with 10 ranges and 0 elementMetadata cols:
       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 [  237,   800]      +
   [8]     chMT [    1,    18]      -
   [9]     chMT [   21,   800]      -
  [10]     chMT [    1,   800]      *
  ---
  seqlengths:
     ch1  chMT
   50000   800
```

Splitting a GRanges object

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

GRangesList of length 2:
$ch1
GRanges with 2 ranges and 2 elementMetadata cols:
    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 elementMetadata cols:
    seqnames     ranges strand | score  GC
  C     chMT [ 18, 134]      + |    13 0.6
  D     chMT [ 19,  20]      - |    14 0.4
  F     chMT [120, 236]      + |    16   0

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

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 columns in their `elementMetadata` slot.

Supported basic operations:

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

GRangesList constructor

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

```
GRangesList of length 2:
[[1]]
GRanges with 5 ranges and 2 elementMetadata cols:
    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 [  120,   236]      + |        16         0

[[2]]
GRanges with 3 ranges and 2 elementMetadata cols:
    seqnames ranges strand | score  GC
  1      ch2 [2, 7]      * |    15   0
  2      ch2 [1, 6]      * |    14 0.2
  3      ch2 [2, 7]      * |    13 0.4

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

GRangesList accessors

```
> length(grl)
[1] 2
> seqnames(grl)
CompressedRleList 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)
CompressedRleList 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)

CompressedIRangesList 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]   120   236   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)

CompressedIntegerList of length 2
[[1]] 35016 17 18 19 120
[[2]] 2 1 2

> width(grl)

CompressedIntegerList 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 elementMetadata cols:
    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 [  120,   236]      + |        16         0

$TX2
GRanges with 3 ranges and 2 elementMetadata cols:
    seqnames ranges strand | score  GC
  1      ch2 [2, 7]      * |    15   0
  2      ch2 [1, 6]      * |    14 0.2
  3      ch2 [2, 7]      * |    13 0.4

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

GRangesList accessors (continued)

```
> elementMetadata(grl)$geneid <- c("GENE1", "GENE2")
> elementMetadata(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 elementMetadata cols:
    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 [  120,   236]      + |        16         0

$TX2
GRanges with 3 ranges and 2 elementMetadata cols:
    seqnames ranges strand | score  GC
  1      ch2 [2, 7]      * |    15   0
  2      ch2 [1, 6]      * |    14 0.2
  3      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>
```

Vector operations on GRangesList objects

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

GRangesList of length 2:
$TX2
GRanges with 3 ranges and 2 elementMetadata cols:
    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

$TX1
GRanges with 5 ranges and 2 elementMetadata cols:
    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 [  120,   236]      + |    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 elementMetadata cols:
    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 [  120,   236]      + |        16         0

$TX2
GRanges with 3 ranges and 2 elementMetadata cols:
    seqnames ranges strand | score  GC
  1      ch2 [2, 7]      * |    15   0
  2      ch2 [1, 6]      * |    14 0.2
  3      ch2 [2, 7]      * |    13 0.4

[[3]]
GRanges with 5 ranges and 2 elementMetadata cols:
    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 [  120,   236]      + |    16   0

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

List operations on GRangesList objects

```
> grl[[2]]

GRanges with 3 ranges and 2 elementMetadata cols:
    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:
     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 elementMetadata cols:
    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 [  120,   236]      + |        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 elementMetadata cols:
    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 [  220,   336]      + |        16         0

$TX2
GRanges with 3 ranges and 2 elementMetadata cols:
    seqnames     ranges strand | score  GC
  1      ch2 [102, 107]      * |    15   0
  2      ch2 [101, 106]      * |    14 0.2
  3      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 elementMetadata cols:
    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 [  220,   336]      + |        16         0

$TX2
GRanges with 3 ranges and 2 elementMetadata cols:
    seqnames     ranges strand | score  GC
  1      ch2 [102, 107]      * |    15   0
  2      ch2 [101, 106]      * |    14 0.2
  3      ch2 [102, 107]      * |    13 0.4

---
seqlengths:
   ch1  chMT   ch2
 50000   800    NA

> elementMetadata(grl100)

DataFrame with 2 rows and 0 columns

> elementMetadata(grl100b)

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

Ranges operations on GRangesList objects

```
> grl
GRangesList of length 2:
$TX1
GRanges with 5 ranges and 2 elementMetadata cols:
    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 [  120,   236]      + |        16         0

$TX2
GRanges with 3 ranges and 2 elementMetadata cols:
    seqnames ranges strand | score  GC
  1      ch2 [2, 7]      * |    15   0
  2      ch2 [1, 6]      * |    14 0.2
  3      ch2 [2, 7]      * |    13 0.4

---
seqlengths:
   ch1  chMT   ch2
 50000   800    NA
> shift(grl, 100)  # equivalent to endoapply(grl, shift, 100)
GRangesList of length 2:
$TX1
GRanges with 5 ranges and 2 elementMetadata cols:
    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 [  220,   336]      + |        16         0

$TX2
GRanges with 3 ranges and 2 elementMetadata cols:
    seqnames     ranges strand | score  GC
  1      ch2 [102, 107]      * |    15   0
  2      ch2 [101, 106]      * |    14 0.2
  3      ch2 [102, 107]      * |    13 0.4
```

Ranges operations on GRangesList objects (continued)

```
> grl
GRangesList of length 2:
$TX1
GRanges with 5 ranges and 2 elementMetadata cols:
    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 [  120,   236]      + |        16         0

$TX2
GRanges with 3 ranges and 2 elementMetadata cols:
    seqnames ranges strand | score  GC
  1      ch2 [2, 7]      * |    15   0
  2      ch2 [1, 6]      * |    14 0.2
  3      ch2 [2, 7]      * |    13 0.4

---
seqlengths:
   ch1  chMT   ch2
 50000   800    NA
> flank(grl, 10)  # equivalent to endoapply(grl, flank, 10)
GRangesList of length 2:
$TX1
GRanges with 5 ranges and 2 elementMetadata cols:
    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 [  110,   119]      + |        16         0

$TX2
GRanges with 3 ranges and 2 elementMetadata cols:
    seqnames  ranges strand | score  GC
  1      ch2 [-8, 1]      * |    15   0
  2      ch2 [-9, 0]      * |    14 0.2
  3      ch2 [-8, 1]      * |    13 0.4
```

Ranges operations on GRangesList objects (continued)

```
> grl
GRangesList of length 2:
$TX1
GRanges with 5 ranges and 2 elementMetadata cols:
    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 [  120,   236]      + |        16         0

$TX2
GRanges with 3 ranges and 2 elementMetadata cols:
    seqnames ranges strand | score  GC
  1      ch2 [2, 7]      * |    15   0
  2      ch2 [1, 6]      * |    14 0.2
  3      ch2 [2, 7]      * |    13 0.4

---
seqlengths:
   ch1  chMT   ch2
 50000   800    NA
> range(grl)  # equivalent to endoapply(grl, range)
GRangesList of length 2:
$TX1
GRanges with 3 ranges and 0 elementMetadata cols:
      seqnames      ranges strand
         <Rle>   <IRanges>  <Rle>
  [1]      ch1 [17, 35020]      -
  [2]     chMT [18,   236]      +
  [3]     chMT [19,    20]      -

$TX2
GRanges with 1 range and 0 elementMetadata cols:
      seqnames ranges strand
  [1]      ch2 [1, 7]      *

---
seqlengths:
   ch1  chMT   ch2
```

Ranges operations on GRangesList objects (continued)

```
> grl
GRangesList of length 2:
$TX1
GRanges with 5 ranges and 2 elementMetadata cols:
    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 [  120,   236]      + |        16         0

$TX2
GRanges with 3 ranges and 2 elementMetadata cols:
    seqnames ranges strand | score  GC
  1      ch2 [2, 7]      * |    15   0
  2      ch2 [1, 6]      * |    14 0.2
  3      ch2 [2, 7]      * |    13 0.4

---
seqlengths:
   ch1  chMT   ch2
 50000   800    NA
> reduce(grl)  # equivalent to endoapply(grl, reduce)
GRangesList of length 2:
$TX1
GRanges with 4 ranges and 0 elementMetadata cols:
      seqnames         ranges strand
         <Rle>      <IRanges>  <Rle>
  [1]      ch1 [   17,    20]      -
  [2]      ch1 [35016, 35020]      -
  [3]     chMT [   18,   236]      +
  [4]     chMT [   19,    20]      -

$TX2
GRanges with 1 range and 0 elementMetadata cols:
      seqnames ranges strand
  [1]      ch2 [1, 7]      *

---
seqlengths:
```

Ranges operations on GRangesList objects (continued)

```
> grl2 <- grl; start(grl2[[1]]) <- start(grl2[[1]]) - 4:0; grl2
GRangesList of length 2:
$TX1
GRanges with 5 ranges and 2 elementMetadata cols:
    seqnames         ranges strand |     score        GC
       <Rle>      <IRanges>  <Rle> | <integer> <numeric>
  A      ch1 [35012, 35020]      - |        11         1
  B      ch1 [   14,    20]      - |        12       0.8
  C     chMT [   16,   134]      + |        13       0.6
  D     chMT [   18,    20]      - |        14       0.4
  F     chMT [  120,   236]      + |        16         0

$TX2
GRanges with 3 ranges and 2 elementMetadata cols:
    seqnames ranges strand | score  GC
  1      ch2 [2, 7]      * |    15   0
  2      ch2 [1, 6]      * |    14 0.2
  3      ch2 [2, 7]      * |    13 0.4

---
seqlengths:
   ch1  chMT   ch2
 50000   800    NA
> psetdiff(grl2, grl)  # equivalent to mendoapply(setdiff, grl2, grl)
GRangesList of length 2:
$TX1
GRanges with 4 ranges and 0 elementMetadata cols:
      seqnames         ranges strand
         <Rle>      <IRanges>  <Rle>
  [1]      ch1 [   14,    16]      -
  [2]      ch1 [35012, 35015]      -
  [3]     chMT [   16,    17]      +
  [4]     chMT [   18,    18]      -

$TX2
GRanges with 0 ranges and 0 elementMetadata cols:
     seqnames ranges strand

---
seqlengths:
   ch1  chMT   ch2
```

The purpose of the GappedAlignments container is...

... to store a set of genomic alignments.

Those alignments are typically loaded from a BAM file (with `readGappedAlignments()`). 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 basic operations:

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

GappedAlignments constructor

Typically not used directly!

```
> gal0 <- GappedAlignments(rname=Rle(c("ch1", "ch2"), c(3, 1)),
+                          pos=1L + 10L*0:3,
+                          cigar=c("36M", "20M3D16M", "20M703N16M", "14M2I20M"),
+                          strand=strand(c("+", "-", "-", "+")))
> gal0
GappedAlignments with 4 alignments and 0 elementMetadata cols:
      seqnames strand       cigar    qwidth     start       end     width
         <Rle>  <Rle> <character> <integer> <integer> <integer> <integer>
  [1]      ch1      +         36M        36         1        36        36
  [2]      ch1      -    20M3D16M        36        11        49        39
  [3]      ch1      -  20M703N16M        36        21       759       739
  [4]      ch2      +    14M2I20M        36        31        64        34
          ngap
     <integer>
  [1]         0
  [2]         0
  [3]         1
  [4]         0
  ---
  seqlengths:
   ch1 ch2
    NA  NA
```

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

readGappedAlignments()

```
> library(SeattleAdvancedWorkshop2012Data)
> gal4 <- readGappedAlignments(pathto_untreated3_chr4())
> length(gal4)

[1] 175346

> head(gal4)

GappedAlignments with 6 alignments and 0 elementMetadata cols:
      seqnames strand       cigar    qwidth     start       end     width
         <Rle>  <Rle> <character> <integer> <integer> <integer> <integer>
  [1]     chr4      +         37M        37       169       205        37
  [2]     chr4      -         37M        37       184       220        37
  [3]     chr4      -         37M        37       187       223        37
  [4]     chr4      +         37M        37       193       229        37
  [5]     chr4      -         37M        37       326       362        37
  [6]     chr4      +         37M        37       943       979        37
          ngap
     <integer>
  [1]         0
  [2]         0
  [3]         0
  [4]         0
  [5]         0
  [6]         0
  ---
  seqlengths:
      chr2L     chr2R     chr3L     chr3R      chr4      chrM      chrX   chrYHet
   23011544  21146708  24543557  27905053   1351857     19517  22422827    347038
```

GappedAlignments accessors

```
> seqnames(gal4)

'factor' Rle of length 175346 with 1 run
  Lengths: 175346
  Values :   chr4
Levels(8): chr2L chr2R chr3L chr3R chr4 chrM chrX chrYHet

> table(as.factor(seqnames(gal4)))

  chr2L   chr2R   chr3L   chr3R    chr4    chrM    chrX chrYHet
      0       0       0       0  175346       0       0       0

> strand(gal4)

'factor' Rle of length 175346 with 37319 runs
  Lengths:  1  2  1  1  3  2  3 10  3  1  4 ...  2  7 26  1  2  1  1  1  1  1  3
  Values :  +  -  +  -  +  -  +  -  +  -  + ...  +  -  +  -  +  -  +  -  +  -  +
Levels(3): + - *

> table(as.factor(strand(gal4)))

    +     -     *
84871 90475     0

> head(cigar(gal4))

[1] "37M" "37M" "37M" "37M" "37M" "37M"

> head(qwidth(gal4))

[1] 37 37 37 37 37 37

> table(qwidth(gal4))

    37
175346
```

GappedAlignments accessors (continued)

```
> head(start(gal4))
[1] 169 184 187 193 326 943
> head(end(gal4))
[1] 205 220 223 229 362 979
> head(width(gal4))
[1] 37 37 37 37 37 37
> head(ngap(gal4))
[1] 0 0 0 0 0 0
> table(ngap(gal4))
     0      1
172529   2817
> seqinfo(gal4)
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"))
> gal4 <- readGappedAlignments(pathto_untreated3_chr4(),
+                              use.names=TRUE, param=param)
> head(gal4)
```

```
GappedAlignments with 6 alignments and 4 elementMetadata cols:
                    seqnames strand       cigar    qwidth     start       end
                       <Rle>  <Rle> <character> <integer> <integer> <integer>
  SRR031715.1138209     chr4      +         37M        37       169       205
   SRR031714.776678     chr4      -         37M        37       184       220
  SRR031715.3258011     chr4      -         37M        37       187       223
  SRR031715.4791418     chr4      +         37M        37       193       229
  SRR031715.1138209     chr4      -         37M        37       326       362
   SRR031714.756385     chr4      +         37M        37       943       979
                        width      ngap |      flag      mapq        NH        NM
                    <integer> <integer> | <integer> <integer> <integer> <integer>
  SRR031715.1138209        37         0 |        99      <NA>         1         0
   SRR031714.776678        37         0 |       153      <NA>         1         2
  SRR031715.3258011        37         0 |        89      <NA>         1         1
  SRR031715.4791418        37         0 |       137      <NA>         1         1
  SRR031715.1138209        37         0 |       147      <NA>         1         0
   SRR031714.756385        37         0 |        99      <NA>         1         0
  ---
  seqlengths:
      chr2L    chr2R    chr3L    chr3R     chr4     chrM     chrX  chrYHet
   23011544 21146708 24543557 27905053  1351857    19517 22422827   347038
```

```
> any(duplicated(names(gal4)))
```

```
[1] TRUE
```

Exercise I

1. Find the SAM Spec online and investigate the meaning of predefined tags NH and NM.
2. Load BAM file untreated3_chr4.bam into a *GappedAlignments* object and subset this object to keep only the alignments satisfying the 2 following conditions:
   - The alignment corresponds to a query with a *unique alignment* (aka *unique match* or *unique hit*).
   - The alignment is a *perfect match* (i.e. no insertion, no deletion, no mismatch).
3. Do those alignments have gaps?

From GappedAlignments 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(gal4, "GRanges")
```

```
GRanges with 175346 ranges and 0 elementMetadata cols:
                    seqnames              ranges strand
                       <Rle>           <IRanges>  <Rle>
  SRR031715.1138209     chr4         [169, 205]      +
   SRR031714.776678     chr4         [184, 220]      -
  SRR031715.3258011     chr4         [187, 223]      -
  SRR031715.4791418     chr4         [193, 229]      +
  SRR031715.1138209     chr4         [326, 362]      -
   SRR031714.756385     chr4         [943, 979]      +
  SRR031714.2355189     chr4         [944, 980]      +
  SRR031714.5054563     chr4         [946, 982]      +
  SRR031715.4533153     chr4         [946, 982]      -
                ...      ...                ...    ...
  SRR031715.3832729     chr4 [1348349, 1348385]      +
  SRR031715.4873052     chr4 [1348350, 1348386]      -
  SRR031714.1650928     chr4 [1349196, 1349232]      +
  SRR031714.1650928     chr4 [1349326, 1349362]      -
  SRR031714.1650928     chr4 [1349708, 1349744]      +
  SRR031714.1650928     chr4 [1349838, 1349874]      -
  SRR031714.5192891     chr4 [1351640, 1351676]      +
  SRR031715.2351056     chr4 [1351640, 1351676]      +
   SRR031714.864195     chr4 [1351760, 1351796]      +
  ---
  seqlengths:
      chr2L    chr2R    chr3L    chr3R     chr4     chrM     chrX  chrYHet
   23011544 21146708 24543557 27905053  1351857    19517 22422827   347038
```

From GappedAlignments 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).

```
> grl4 <- as(gal4, "GRangesList")
> grl4
```

```
GRangesList of length 175346:
$SRR031715.1138209
GRanges with 1 range and 0 elementMetadata cols:
      seqnames     ranges strand
         <Rle>  <IRanges>  <Rle>
  [1]     chr4 [169, 205]      +

$SRR031714.776678
GRanges with 1 range and 0 elementMetadata cols:
      seqnames     ranges strand
  [1]     chr4 [184, 220]      -

$SRR031715.3258011
GRanges with 1 range and 0 elementMetadata cols:
      seqnames     ranges strand
  [1]     chr4 [187, 223]      -

...
<175343 more elements>
---
seqlengths:
    chr2L     chr2R     chr3L     chr3R      chr4      chrM      chrX   chrYHet
 23011544 21146708 24543557 27905053   1351857     19517 22422827    347038
```

From GappedAlignments to GRangesList (continued)

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

```
> all(elementLengths(grl4) == ngap(gal4) + 1)
```

```
[1] TRUE
```

Coverage

```
> cvg4 <- coverage(grl4)
> cvg4

SimpleRleList 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 104680 runs
  Lengths:  168   15    3    6   13   15    3 ...   37 1765   37   83   37   61
  Values :    0    1    2    3    4    3    2 ...    1    0    2    0    1    0

...
<3 more elements>
```

Coverage (continued)

```
> mean(cvg4)
```

```
   chr2L    chr2R    chr3L    chr3R     chr4     chrM     chrX  chrYHet
0.000000 0.000000 0.000000 0.000000 4.799178 0.000000 0.000000 0.000000
```

```
> max(cvg4)
```

```
  chr2L   chr2R   chr3L   chr3R    chr4    chrM    chrX chrYHet
      0       0       0       0    7317       0       0       0
```

Slicing the coverage

```
> sl4 <- slice(cvg4, lower=10)
> sl4

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

> elementLengths(sl4)

  chr2L   chr2R   chr3L   chr3R    chr4    chrM    chrX chrYHet
      0       0       0       0    2322       0       0       0

> head(sl4$chr4)

Views on a 1351857-length Rle subject

views:
    start  end width
[1]  5968 6004    37 [12 12 12 13 13 13 13 13 13 14 14 14 14 14 15 15 15 15 ...]
[2]  6607 6634    28 [10 11 11 11 11 11 11 11 11 11 11 11 12 12 12 12 12 12 ...]
[3]  6790 6868    79 [14 13 13 13 13 20 20 22 23 23 24 26 26 25 28 28 29 29 ...]
[4]  6874 6874     1 [10]
[5]  6917 6917     1 [10]
[6]  6920 6939    20 [10 10 10 11 11 11 11 11 11 10 10 10 11 11 11 11 11 11 ...]

> head(mean(sl4$chr4))

[1] 13.40541 11.00000 21.65823 10.00000 10.00000 10.70000

> head(max(sl4$chr4))

[1] 15 12 37 10 10 11
```

Finding/counting overlaps

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

Typical input:

- A BAM file with the aligned reads.
- Transcript annotations *for the same reference genome* that was used to align the reads.

Typical tools:

- The `readGappedAlignments()` function to load the reads in a *GappedAlignments* object.
- A *TranscriptDb* object containing the transcript annotations.
- The `exonBy()` 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.

Finding/counting overlaps (continued)

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

GRangesList of length 23017:
$FBtr0089116
GRanges with 11 ranges and 3 elementMetadata cols:
       seqnames           ranges strand |   exon_id   exon_name exon_rank
          <Rle>        <IRanges>  <Rle> | <integer> <character> <integer>
   [1]     chr4 [251356, 251521]      + |         1        <NA>         1
   [2]     chr4 [252561, 252603]      + |         2        <NA>         2
   [3]     chr4 [252905, 253474]      + |         3        <NA>         3
   [4]     chr4 [254891, 254971]      + |         4        <NA>         4
   [5]     chr4 [255490, 255570]      + |         5        <NA>         5
   [6]     chr4 [257021, 257101]      + |         6        <NA>         6
   [7]     chr4 [257895, 258185]      + |         7        <NA>         7
   [8]     chr4 [260940, 261024]      + |         8        <NA>         8
   [9]     chr4 [263892, 264211]      + |         9        <NA>         9
  [10]     chr4 [264260, 264374]      + |        10        <NA>        10
  [11]     chr4 [265806, 266500]      + |        11        <NA>        11

...
<23016 more elements>
---
seqlengths:
     chr2L  chr2LHet     chr2R  chr2RHet ...   chrXHet   chrYHet      chrM
  23011544    368872  21146708   3288761 ...    204112    347038     19517
```

Finding/counting overlaps (continued)

```
> txhits <- countOverlaps(exbytx, grl4)
> length(txhits)

[1] 23017

> head(txhits)

FBtr0089116 FBtr0300800 FBtr0300796 FBtr0300799 FBtr0300798 FBtr0300797
        365         406         410         370         410         407

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

FBtr0089175 FBtr0089176 FBtr0089177 FBtr0112904 FBtr0289951 FBtr0089243
      14376       14051       13811        5433        5411        5410
```

Rough counting!

- The fact that the reads are actually *paired-end* is ignored.
- 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.

Exercise II

Use the *TxDb.Dmelanogaster.UCSC.dm3.ensGene* package and the result of Exercise I to count the number of *unique hits* per transcript, that is, the number of hits from reads with a *unique alignment*.

Final notes

Under active development:

- Facilities for dealing with *paired-end* reads (*GappedAlignmentPairs* container).
- Facilities for detecting/counting hits (from *single-end* or *paired-end* reads) that are *compatible* with the splicing of the transcript.

Resources:

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