

# Computing with Sequences and Ranges

Martin Morgan ([martin.morgan@roswellpark.org](mailto:martin.morgan@roswellpark.org))  
Roswell Park Cancer Institute  
Buffalo, NY, USA

12 July, 2016

## Sequences: representation

*DNAMStringSet*: Vector of sequences, e.g., sequence of each exon in the UCSC knownGene track

A *DNAMStringSet* instance of length 289969

width seq

```
[1]    354 CTTGCCGTCAGCCTTT...TCACAACCTAGGCCA
[2]    127 GTCCTGTCTCCCCC...CCCAGTGTTGCAGAG
[3]    109 GTGTGTGGTGATGCCA...CCCAGTGTTGCAGAG
...    ...
[289968] 109 GTGTGTGGTGATGCCA...CCCAGTGTTGCAGAG
[289969] 354 CTTGCCGTCAGCCTTT...TGACAACCTAGGCCA
```

- ▶ Acts like a *vector*, e.g., `length()`, `[`, `[[`
- ▶ Many methods – `methods(class="DNAMStringSet")` – e.g., `reverseComplement()`, `letterFrequency()`, ...

# Sequences: packages

**Biostrings** General purpose biological sequence representation.

**BSgenome** Whole-genome representation.

**ShortRead** High-throughput sequencing.

## Sequences: classes

- `DNASTring` Single DNA sequence, e.g., chromosome
- `DNASTringSet` Vector of DNA sequences. Actually, `XString`,  
`XStringSet`: X could be DNA, RNA, AA)
- `BSgenome` Collection of (large) DNA sequences
- `ShortReadQ` High-throughput reads & their qualities

## Sequences: file references

TwoBitFile, FaFile .2bit (in *rtracklayer*) or .fa (in *Rsamtools*)  
indexed genome-scale fasta files.

FastqFile , e.g., *FastqStreamer* (in *ShortRead*)

Effectively manage large data

- ▶ *Restrict* input to specific genomic locations
- ▶ *Iterate* through large files in chunks

# Sequences: annotations

*BSgenome.\** packages

- ▶ E.g., *BSgenome.Hsapiens.UCSC.hg19*
- ▶ Packages containing whole-genome sequences for model organisms

*AnnotationHub* resources

- ▶ e.g., Ensembl FASTA files

# Ranges: *GRanges* representation

```
> gr = exons(TxDb.Hsapiens.UCSC.hg19.knownGene); gr
```

```
GRanges with 289969 ranges and 1 metadata column:
```

	seqnames	ranges	strand	exon_id
	<Rle>	<IRanges>	<Rle>	<integer>
[1]	chr1	[11874, 12227]	+	1
[2]	chr1	[12595, 12721]	+	2
[3]	chr1	[12613, 12721]	+	3
...	...	...	...	...
[289967]	chrY	[59358329, 59359508]	-	277748
[289968]	chrY	[59360007, 59360115]	-	277749
[289969]	chrY	[59360501, 59360854]	-	277750

```
---  
seqinfo: 93 sequences (1 circular) from hg19 genome
```

## *GRanges*

```
length(gr); gr[1:5]  
seqnames(gr)  
start(gr)  
end(gr)  
width(gr)  
strand(gr)
```

## *DataFrame*

```
mcols(gr)  
gr$exon_id
```

## *Seqinfo*

```
seqlevels(gr)  
seqlengths(gr)  
genome(gr)
```

- ▶ Data: aligned reads, called peaks, SNP locations, CNVs, ...
- ▶ Annotation: gene models, variants, regulatory regions, ...

# Ranges: *GRangesList* representation

```
> gr1 = exonsBy(TxDb.Hsapiens.UCSC.hg19.knownGene, "tx", use.names=TRUE); gr1
```

```
GRangesList of length 82960:
```

```
$uc001aaa.3
```

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

	seqnames	ranges	strand	exon_id	exon_name	exon_rank
	<Rle>	<IRanges>	<Rle>	<integer>	<character>	<integer>
[1]	chr1	[11874, 12227]	+	1	<NA>	1
[2]	chr1	[12613, 12721]	+	3	<NA>	2
[3]	chr1	[13221, 14409]	+	5	<NA>	3

```
GRangesList  
(list of GRanges)  
length(gr1)  
gr1[1:3]  
shift(gr1, 1)  
range(gr1)
```

```
$uc010nxq.1
```

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

	seqnames	ranges	strand	exon_id	exon_name	exon_rank
[1]	chr1	[11874, 12227]	+	1	<NA>	1
[2]	chr1	[12595, 12721]	+	2	<NA>	2
[3]	chr1	[13403, 14409]	+	6	<NA>	3

```
GRanges  
gr1[[2]]  
gr1[["uc010nxq.1"]]
```

```
$uc010nxr.1
```

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

	seqnames	ranges	strand	exon_id	exon_name	exon_rank
[1]	chr1	[11874, 12227]	+	1	<NA>	1
[2]	chr1	[12646, 12697]	+	4	<NA>	2
[3]	chr1	[13221, 14409]	+	5	<NA>	3

Two kinds of fun!

```
introns =  
  psetdiff(range(gr1), gr1)
```

```
grr = unlist(gr1)  
## transform grr, then...  
gr1 = relist(grr, gr1)
```

'flesh' 'skeleton'

```
...  
<82957 more elements>
```

```
----
```

```
seqinfo: 93 sequences (1 circular) from hg19 genome
```



## Ranges: packages

**GenomicRanges** Essential representation and operations

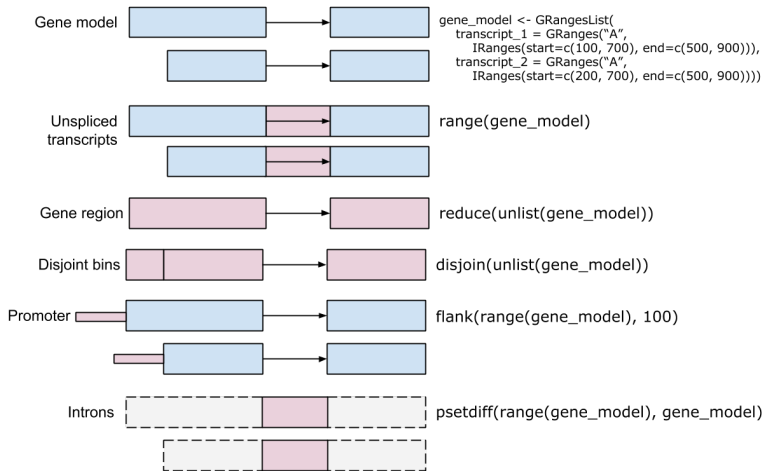
**GenomicAlignments** Aligned reads as genomic ranges

**GenomicFeatures** Annotations as genomic ranges

**rtracklayer** Annotation (e.g., BED, GTF) input

A little more advanced usage: *IRanges* (); *S4Vectors* (underling conceptual ideas)

# Ranges: operations



- ▶ Many more, e.g., `methods(class="GRanges")`

## Ranges: `findOverlaps()`

- ▶ Overlaps between query and subject genomic ranges
- ▶ Different types of overlap, e.g., 'any', 'within', ...

```
> q <- GRanges("chr1", IRanges(10, 20))
> s <- GRanges("chr1", IRanges(5, width=c(3, 6, 9)))
> findOverlaps(q, s)
```

Hits object with 2 hits and 0 metadata columns:

```
      queryHits subjectHits
      <integer>  <integer>
[1]           1           2
[2]           1           3
```

-----

```
queryLength: 1
subjectLength: 3
```

- ▶ *Hits* object describing many-to-many relationship between overlapping ranges.

## Ranges: working with files

`import` (*rtracklayer*) for BED, GTF, and other common web file import functions. *BEDFile*, *GTFFile*, etc.

`readGAlignments / readGAlignmentsList` (*GenomicAlignments*) for aligned reads in BAM files

`BamFile` (*Rsamtools*) for lower-level access to BAM files, e.g., restriction and iteration

# Ranges: annotation

*TxDb.\** packages

- ▶ E.g., *TxDb.Hsapiens.UCSC.hg19.knownGene*
- ▶ Genomic ranges for exons, transcripts, coding sequences, and how these are ordered into gene models, e.g., exons grouped by transcript

*AnnotationHub* resources

- ▶ Ensembl gene models
- ▶ Roadmap Epigenomics regulatory marks
- ▶ Many other range-based resources

# What's to love about a data.frame?

- ▶ Coordinated data management
- ▶ Familiarity
- ▶ Interoperability

## And yet...

- ▶ Not all columns are equal – seqlevels, start, width, strand are required, other columns are optional.
- ▶ Columns have special meaning – e.g., widths of genomic ranges are  $\geq 0$
- ▶ Not every column is a base *R* vector – *DNASTringSet*
- ▶ Data from different tables are often related – information about genomes

## Principled versus precocious?

- ▶ The tidyverse as principled – a few well-designed orthogonal functions operating exclusively (?) on `data.frame` derivatives
- ▶ *R* and *Bioconductor* as precocious – a large number of classes and specialized functions
- ▶ Some of precociousness is lack of principle, but some reflects high-level summary of complex work flows



## Other resources

- ▶ [Workflows](#) & package vignettes
- ▶ [GenomicRanges](#) and other 'cheat sheets'
- ▶ Course material
- ▶ Support site [tutorials](#)

# Acknowledgments

- ▶ Core: Valerie Obenchain, Hervé Pagès, (Dan Tenenbaum), Lori Shepherd, Marcel Ramos, Yubo Cheng.
- ▶ The research reported in this presentation was supported by the National Cancer Institute and the National Human Genome Research Institute of the National Institutes of Health under Award numbers U24CA180996 and U41HG004059. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health or the National Science Foundation.

<https://bioconductor.org>,

<https://support.bioconductor.org>