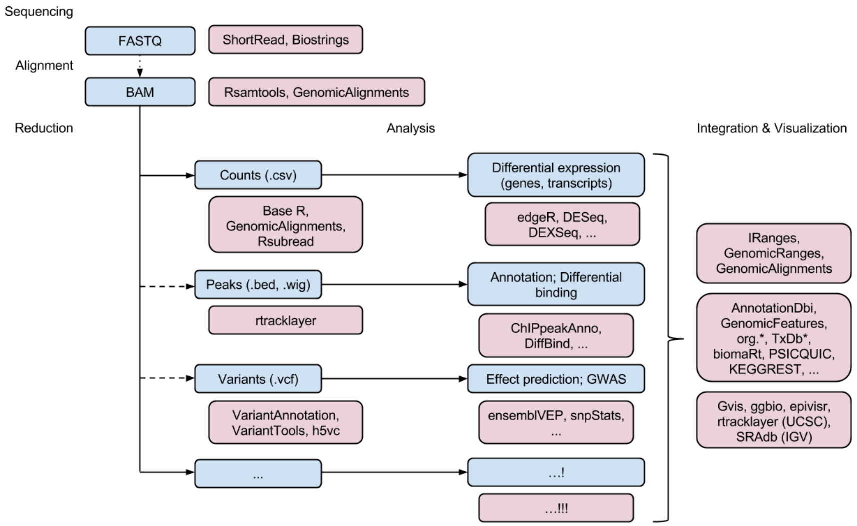
Bioconductor

Mikhail Dozmorov

Spring 2018

High-throughput sequence workflow


Sequencing
FASTQ
ShortRead, Biostrings
Alignment
BAM
Rsamtools, GenomicAlignments
Reduction
Analysis
Integration & Visualization
Counts (.csv)
Base R, GenomicAlignments, Rsubread
Differential expression (genes, transcripts)
edgeR, DESeq, DEXSeq, ...
Peaks (.bed, .wig)
rtracklayer
Annotation; Differential binding
ChIPpeakAnno, DiffBind, ...
Variants (.vcf)
VariantAnnotation, VariantTools, h5vc
Effect prediction; GWAS
ensemblVEP, snpStats, ...
...
...!
...!!!
IRanges, GenomicRanges, GenomicAlignments
AnnotationDbi, GenomicFeatures, org.*, TxDb*, biomaRt, PSICQUIC, KEGGREST, ...
Gvis, ggbio, epivisr, rtracklayer (UCSC), SRAdb (IGV)

Bioconductor

Analysis and comprehension of high-throughput genomic data

- Statistical analysis designed for large genomic data
- Interpretation: biological context, visualization, reproducibility
- Support for all high-throughput technologies
  - Sequencing: RNASeq, ChIPSeq, variants, copy number, . . .
  - Microarrays: expression, SNP, . . .
  - Flow cytometry, proteomics, images, . . .

Bioconductor cheat sheet https://github.com/mikelove/bioc-refcard

Bioconductor packages

- https://www.bioconductor.org/
- Over 1,400 packages
- Discover and navigate via `biocView`
- Informative package 'landing page'
    - Title, author / maintainer, short description, citation, installation instructions, . . . , download statistics
- 'Release' (every six months) and 'devel' branches

http://bioconductor.org/packages/release/BiocViews.html#___Software

Reference manuals, vignettes

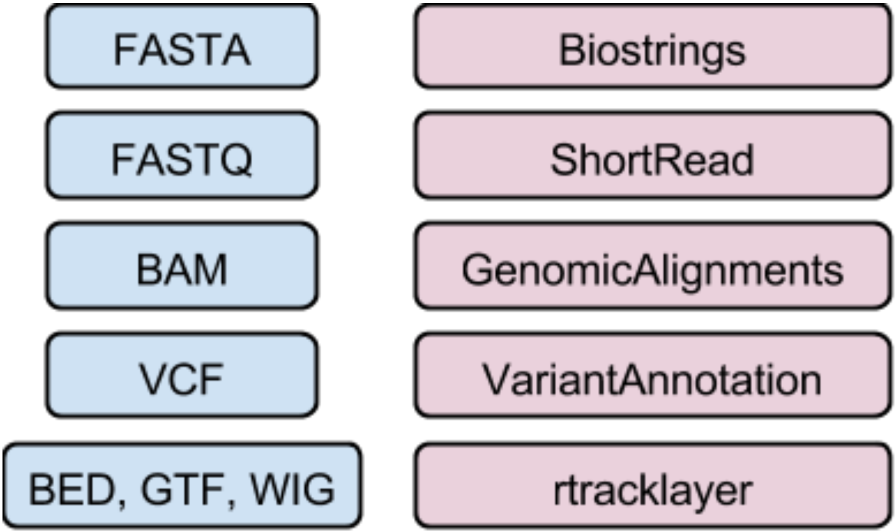
- All user-visible functions have help pages, most with runnable examples
- ‘Vignettes’ an important feature in *Bioconductor* – narrative documents illustrating how to use the package, with integrated code
    - Example: `AnnotationHub` landing page, “AnnotationHub HOW TO’s” vignette illustrating some fun use cases.

http://bioconductor.org/packages/devel/AnnotationHub

Objects

- *Bioconductor* makes extensive use of classes to represent complicated data types
- Classes foster interoperability – many different packages can work on the same data – but can be a bit intimidating
- Formal ‘S4’ object system
    - Often a class is described on a particular home page, e.g., `?GRanges`, and in vignettes, e.g., `vignette(package="GenomicRanges")`, `vignette("GenomicRangesIntroduction")`
    - Many methods and classes can be discovered interactively , e.g., `methods(class="GRanges")` to find out what one can do with a `GRanges` instance, and `methods(findOverlaps)` for classes that the `findOverlaps()` function operates on.
    - In more advanced cases, one can look at the actual definition of a class or method using `getClass()`, `getMethod()`
- Interactive help
    - `?findOverlaps,<tab>` to select help on a specific method, `?GRanges-class` for help on a class.

High-throughput sequence data



FASTA	Biostrings
FASTQ	ShortRead
BAM	GenomicAlignments
VCF	VariantAnnotation
BED, GTF, WIG	rtracklayer

DNA/amino acid sequences: FASTA files

- The `Biostrings` package, is used to represent DNA and other sequences, with many convenient sequence-related functions, e.g., `?consensusMatrix`.

Input & manipulation, FASTA file example:

```
>NM_078863_up_2000_chr2L_16764737_f chr2L:16764737-16766736
gttggtggcccaccagtgccaaaatacacaagaagaagaaacagcatctt
gacactaaaatgcaaaaattgctttgcgtcaatgactcaaaacgaaaatg
...
atgggtatcaagttgccccgtataaaaggcaagtttaccggttgcacggt
>NM_001201794_up_2000_chr2L_8382455_f chr2L:8382455-8384454
ttatttatgtaggcgcccgttcccgcagccaaagcactcagaattccggg
cgtgtagcgcaacgaccatctacaaggcaatattttgatcgcttgttagg
...
```

http://bioconductor.org/packages/Biostrings

Reads: FASTQ files

- The `ShortRead` package can be used for lower-level access to FASTQ files. `readFastq()`, `FastqStreamer()`, `FastqSampler()`

Input & manipulation, FASTQ file example:

```
@ERR127302.1703 HWI-EAS350_0441:1:1:1460:19184#0/1
CCTGAGTGAAGCTGATCTTGATCTACGAAGAGAGATAGATCTTGATCGTCGAGGAGATGCTC
+
HHGHHGHHHHHHHDGG<GDGGE@GDGGD<?B8??ADAD<BE@EE8EGDGA3CB85*,77@>
@ERR127302.1704 HWI-EAS350_0441:1:1:1460:16861#0/1
GCGGTATGCTGGAAGGTGCTCGAATGGAGAGCGCCAGCGCCCCGGCGCTGAGCCGCAGCCTC
+
DE?DD>ED4>EEE>DE8EEEDE8B?EB<@3;BA79?,881B?@73;1?#############
```

http://bioconductor.org/packages/ShortRead

Quality scores: 'phred-like', encoded. See http://en.wikipedia.org/wiki/FASTQ_format#Encoding

`Biostrings`, DNA or amino acid sequences

Classes

- `XString`, `XStringSet`, e.g., `DNAString` (genomes), `DNAStringSet` (reads)

Methods

- Manipulation, e.g., `reverseComplement()`
- Summary, e.g., `letterFrequency()`
- Matching, e.g., `matchPDict()`, `matchPWM()`

Related packages: `BSgenome` for working with whole genome sequences, e.g., `?"getSeq,BSgenome-method"`

http://bioconductor.org/packages/release/bioc/vignettes/Biostrings/inst/doc/BiostringsQuickOverview.pdf

http://bioconductor.org/packages/BSgenome

Aligned reads: SAM/BAM files

Input & manipulation:

- 'low-level' `Rsamtools` - `scanBam()`, `BamFile()`
- 'high-level' `GenomicAlignments` - `readGAlignments()`

SAM Header example

```
@HD     VN:1.0  SO:coordinate
@SQ     SN:chr1 LN:249250621
@SQ     SN:chr10        LN:135534747
@SQ     SN:chr11        LN:135006516
...
@SQ     SN:chrY LN:59373566
@PG     ID:TopHat       VN:2.0.8b       CL:/home/hpages/tophat
```

http://bioconductor.org/packages/Rsamtools

http://bioconductor.org/packages/GenomicAlignments

`GenomicAlignments`, Aligned reads

The `GenomicAlignments` package is used to input reads aligned to a reference genome. See for instance the `?readGAlignments` help page and `vignette(package="GenomicAlignments", "summarizeOverlaps")`

Classes - `GenomicRanges`-like behaivor

- `GAlignments`, `GAlignmentPairs`, `GAlignmentsList`

Methods

- `readGAlignments()`, `readGAlignmentsList()`
  - Easy to restrict input, iterate in chunks
- `summarizeOverlaps()`

Called variants: VCF files

Input and manipulation:

- `VariantAnnotation` - `readVcf()`, `readInfo()`, `readGeno()` selectively with `ScanVcfParam()`.

http://bioconductor.org/packages/VariantAnnotation

VCF Header

```
##fileformat=VCFv4.2
##fileDate=20090805
##source=myImputationProgramV3.1
##reference=file:///seq/references/1000GenomesPilot-NCBI36.fas
##contig=<ID=20,length=62435964,assembly=B36,md5=f126cdf8a6e0c
##phasing=partial
##INFO=<ID=DP,Number=1,Type=Integer,Description="Total Depth">
##INFO=<ID=AF,Number=A,Type=Float,Description="Allele Frequenc
...
##FILTER=<ID=q10,Description="Quality below 10">
##FILTER=<ID=s50,Description="Less than 50% of samples have da
...
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Qu
```

VCF Location info

```
#CHROM POS     ID        REF    ALT     QUAL FILTER ...
20     14370   rs6054257 G      A       29   PASS   ...
20     17330   .         T      A       3    q10    ...
20     1110696 rs6040355 A      G,T     67   PASS   ...
20     1230237 .         T      .       47   PASS   ...
20     1234567 microsat1 GTC    G,GTCT  50   PASS   ...
```

VCF Variant INFO

```
#CHROM POS     ...  INFO                               ...
20     14370   ...  NS=3;DP=14;AF=0.5;DB;H2            ...
20     17330   ...  NS=3;DP=11;AF=0.017                ...
20     1110696 ...  NS=2;DP=10;AF=0.333,0.667;AA=T;DB  ...
20     1230237 ...  NS=3;DP=13;AA=T                    ...
20     1234567 ...  NS=3;DP=9;AA=G                     ...
```

Genotype FORMAT and samples

```
... POS     ...  FORMAT      NA00001        NA00002        NA(
... 14370   ...  GT:GQ:DP:HQ 0|0:48:1:51,51 1|0:48:8:51,51 1/1
... 17330   ...  GT:GQ:DP:HQ 0|0:49:3:58,50 0|1:3:5:65,3   0/(
... 1110696 ...  GT:GQ:DP:HQ 1|2:21:6:23,27 2|1:2:0:18,2   2/2
... 1230237 ...  GT:GQ:DP:HQ 0|0:54:7:56,60 0|0:48:4:51,51 0/(
... 1234567 ...  GT:GQ:DP    0/1:35:4       0/2:17:2       1/1
```

VariantAnnotation, Called variants

Classes - GenomicRanges-like behavior

- VCF – 'wide'
- VRanges – 'tall'

Methods

- I/O and filtering: readVcf(), readGeno(), readInfo(), readGT(), writeVcf(), filterVcf()
- Annotation: locateVariants() (variants overlapping ranges), predictCoding(), summarizeVariants()
- SNPs: genotypeToSnpMatrix(), snpSummary()

VCF-Related packages

- `ensemblVEP`- query the Ensembl Variant Effect Predictor
- `VariantTools` - Explore, diagnose, and compare variant calls.
- `VariantFiltering` - Filtering of coding and non-coding genetic variants.
- `h5vc` - has variant calling functionality.
- `snpStats` - Classes and statistical methods for large SNP association studies.

http://bioconductor.org/packages/ensemblVEP

http://bioconductor.org/packages/VariantTools

http://bioconductor.org/packages/VariantFiltering

http://bioconductor.org/packages/h5vc

https://bioconductor.org/packages/release/bioc/html/snpStats.html

Obenchain, V, Lawrence, M, Carey, V, Gogarten, S, Shannon, P, and Morgan, M. VariantAnnotation: a Bioconductor package for exploration and annotation of genetic variants. Bioinformatics, first published online March 28, 2014 doi:10.1093/bioinformatics/btu168, http://bioinformatics.oxfordjournals.org/content/early/2014/04/21/bioinformatics.btu168

Introduction to VariantAnnotation, http://bioconductor.org/packages/release/bioc/vignettes/ShortRead/inst/doc/Overview.pdf

Genome annotations: BED, WIG, GTF, etc. files

- The `rtracklayer`'s `import` and `export` functions can read in many common file types, e.g., `BED`, `WIG`, `GTF`, ..., in addition to querying and navigating the UCSC genome browser. Check out the `?import` page for basic usage.

Input: `rtracklayer::import()`

- `BED`: range-based annotation (see http://genome.ucsc.edu/FAQ/FAQformat.html for definition of this and related formats)
- `WIG/bigWig`: dense, continuous-valued data
- `GTF`: gene model

http://bioconductor.org/packages/rtracklayer

GTF Component coordinates

```
7   protein_coding  gene          27221129    27224842     .   -
...
7   protein_coding  transcript   27221134    27224835     .   -
7   protein_coding  exon          27224055    27224835     .   -
7   protein_coding  CDS           27224055    27224763     .   -
7   protein_coding  start_codon 27224761    27224763     .   -
7   protein_coding  exon          27221134    27222647     .   -
7   protein_coding  CDS           27222418    27222647     .   -
7   protein_coding  stop_codon  27222415    27222417     .   -
7   protein_coding  UTR           27224764    27224835     .   -
7   protein_coding  UTR           27221134    27222414     .   -
```

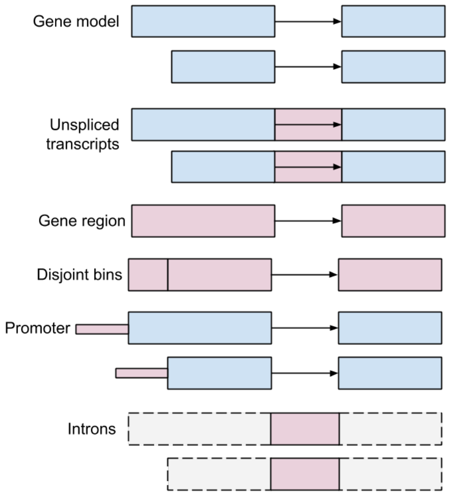
GTF Annotations

```
gene_id "ENSG00000005073"; gene_name "HOXA11"; gene_source "er
...
... transcript_id "ENST00000006015"; transcript_name "HOXA11-0
... exon_number "1"; exon_id "ENSE00001147062";
... exon_number "1"; protein_id "ENSP00000006015";
... exon_number "1";
... exon_number "2"; exon_id "ENSE00002099557";
... exon_number "2"; protein_id "ENSP00000006015";
... exon_number "2";
...
```

Read GTF file into R, https://davetang.org/muse/2017/08/04/read-gtf-file-r/

Data representation in R / Bioconductor

Ranges overview



```
gene_model <- GRangesList(
   transcript_1 = GRanges("A",
      IRanges(start=c(100, 700), end=c(500, 900))),
   transcript_2 = GRanges("A",
      IRanges(start=c(200, 700), end=c(500, 900))))
```

`range(gene_model)`

`reduce(unlist(gene_model))`

`disjoin(unlist(gene_model))`

`flank(range(gene_model), 100)`

`psetdiff(range(gene_model), gene_model)`

Ranges in Bioconductor

- IRanges
    - `start()` / `end()` / `width()`
    - List-like – `length()`, subset, etc.
    - 'metadata', `mcols()`
- GRanges
    - 'seqnames' (chromosome), 'strand'
    - `Seqinfo`, including `seqlevels` and `seqlengths`

Range methods

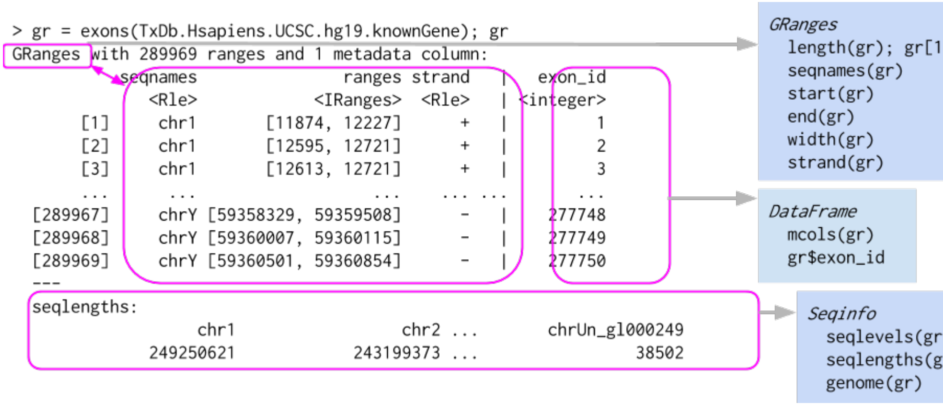
- Intra-range methods
  - Independent of other ranges in the same object
  - `shift()`, `narrow()`, `flank()`, `promoters()`, `resize()`, `restrict()`, `trim()`
  - See `?"intra-range-methods"`
- Inter-range methods
  - Depends on other ranges in the same object
  - `range()`, `reduce()`, `gaps()`, `disjoin()`, `coverage()`
  - See `?"inter-range-methods"`
- Between-range methods
  - Functions of two (or more) range objects
  - `findOverlaps()`, `countOverlaps()`, `summarizeOverlaps()`, `...`, `%over%`, `%within%`, `%outside%`; `union()`, `intersect()`, `setdiff()`, `punion()`, `pintersect()`, `psetdiff()`

IRanges

- The `IRanges` package defines an important class for specifying integer ranges
- There are many interesting operations to be performed on ranges, e.g, `flank()` identifies adjacent ranges
- `IRanges` extends the `Ranges` class

Genomic Ranges

The `GenomicRanges` package extends the notion of ranges to include features relevant to application of ranges in sequence analysis, particularly the ability to associate a range with a sequence name (e.g., chromosome) and a strand.



```
> 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
  ---
  seqlengths:
                  chr1                  chr2 ...      chrUn_gl000249
             249250621             243199373 ...               38502
```

```
GRanges
  length(gr); gr[1
  seqnames(gr)
  start(gr)
  end(gr)
  width(gr)
  strand(gr)
```

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

```
Seqinfo
  seqlevels(gr
  seqlengths(g
  genome(gr)
```

GenomicRanges

- Data (e.g., aligned reads, called peaks, copy number)
- Annotations (e.g., genes, exons, transcripts)
- Close relation to BED files (see `rtracklayer::import.bed()` and `HelloRanges`)
- Also vector interface – `length()`, `[`, etc.

Lists of Genomic Ranges

- List definition - all elements of the same type
- E.g., lists of exons-within-transcripts, alignments-within-reads
- Many *List-aware methods, but a common ‘trick’: apply a vectorized function to the unlisted representaion, then re-list

Lawrence M, Huber W, Pagès H, Aboyoun P, Carlson M, et al. (2013) Software for Computing and Annotating Genomic Ranges. PLoS Comput Biol 9(8): e1003118. doi:10.1371/journal.pcbi.1003118

Lists of Genomic Ranges

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

$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

$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

...
<82957 more elements>
---
 seqinfo: 93 sequences (1 circular) from hg19 genome
```

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

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

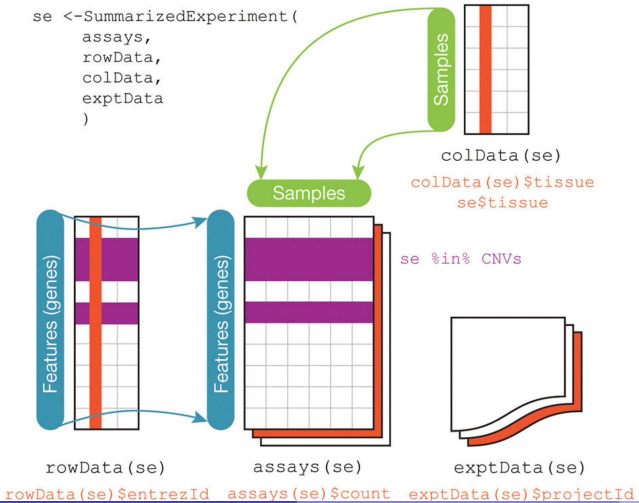
```
Two kinds of fun!
  introns =
    psetdiff(range(grl), grl)

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

'flesh' 'skeleton'

Summarized Experiments

`SummarizedExperiment` - Rows are indexed by a dataframe of *features*. Accessible with `rowData()`



RangedSummarizedExperiment

- `RangedSummarizedExperiment` - Rows are indexed by *genomic ranges*. Accessble with `rowRanges()`

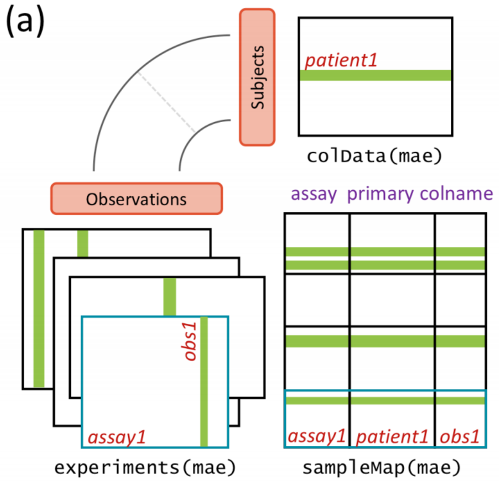
SingleCellExperiment

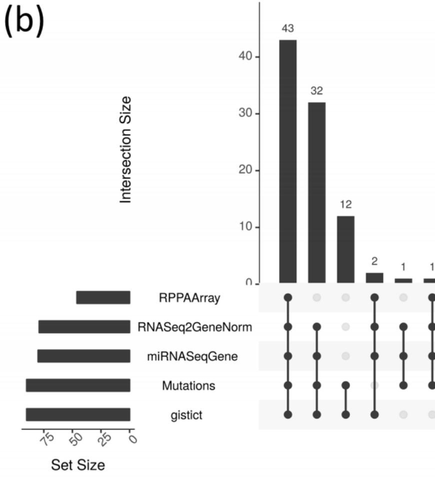
- `SingleCellExperiment` - an extension of `RangedSummarizedExperiment` with several internal slots
  - Has a slot for spike-in measures
  - Can store reduced dimensionality representation of the data

https://bioconductor.org/packages/3.7/bioc/vignettes/SingleCellExperiment/inst/doc/intro.html

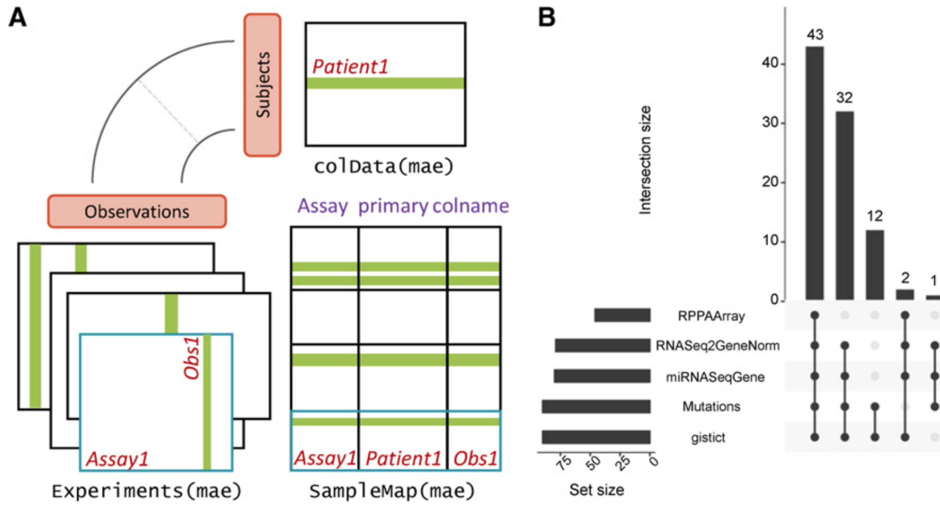
MultiAssayExperiment

Bioconductor package for management of multi-assay data. Especially useful for integrating TCGA datasets.





MultiAssayExperiment



Ramos, Marcel, Lucas Schiffer, Angela Re, Rimsha Azhar, Azfar Basunia, Carmen Rodriguez Cabrera, Tiffany Chan, et al. "Software For The Integration Of Multi-Omics Experiments In Bioconductor," June 1, 2017. doi:10.1101/144774. http://biorxiv.org/content/early/2017/06/01/144774

Annotation packages

- *Bioconductor* provides extensive access to ‘annotation’ resources, see the “AnnotationData” biocViews hierarchy.
- `AnnotationDBI` - is a cornerstone of “AnnotationData” packages, provides user interface and database connection code for annotation data packages using SQLite data storage.

https://bioconductor.org/packages/release/BiocViews.html#___AnnotationData

http://bioconductor.org/packages/AnnotationDbi

Annotation packages

- **org** packages (e.g., `org.Hs.eg.db`) contain maps between different gene identifiers, e.g., ENTREZ and SYMBOL. The basic interface to these packages is described on the help page `?select`
- **TxDb** packages (e.g., `TxDb.Hsapiens.UCSC.hg38.knownGene`) contain gene models (exon coordinates, exon / transcript relationships, etc) derived from common sources such as the hg38 `knownGene` track of the UCSC genome browser. These packages can be queried, e.g., as described on the `?exonsBy` page to retrieve all exons grouped by gene or transcript.

https://bioconductor.org/packages/org.Hs.eg.db

https://bioconductor.org/packages/TxDb.Hsapiens.UCSC.hg38.knownGene

Annotation packages

- **EnsDb** packages and databases (e.g. `EnsDb.Hsapiens.v86`) provide, similar to TxDb packages, gene models, but also protein annotations (protein sequences and protein domains within these) and additional annotation columns such as `"gene_biotype"` or `"tx_biotype"` defining the *biotype* of the features (e.g. lincRNA, protein_coding, miRNA etc). `EnsDb` databases are designed for Ensembl annotations and contain annotations for all genes (protein coding and non-coding) for a specific Ensembl release.
- **BSgenome** packages (e.g., `BSgenome.Hsapiens.UCSC.hg19`) contain whole genomes of model organisms. See `available.genomes()` for pre-packaged genomes.

annotation work flow, http://bioconductor.org/help/workflows/annotation/annotation/

https://bioconductor.org/packages/release/data/annotation/html/EnsDb.Hsapiens.v86.html

https://bioconductor.org/packages/release/data/annotation/html/BSgenome.Hsapiens.UCSC.hg19.html

Annotation methods

- Annotation packages usually contain an object named after the package itself. These objects are collectively called `AnnotationDb` objects with more specific classes named `OrgDb`, `ChipDb` or `TranscriptDb` objects.
- Methods that can be applied to these objects include `cols()`, `keys()`, `keytypes()` and `select()`.

Annotation methods

Category	Function	Description
Discover	`columns()`	List the kinds of columns that can be returned
	`keytypes()`	List columns that can be used as keys
	`keys()`	List values that can be expected for a given keytyp
	`select()`	Retrieve annotations matching `keys`, `keytype` anc

Annotation methods

Category	Function	Description
Manipulate	`setdiff()`, `union()`, `intersect()`	Operations on sets
	`duplicated()`, `unique()`	Mark or remove duplica
	`%in%`, `match()`	Find matches
	`any()`, `all()`	Are any `TRUE`? Are all?
	`merge()`	Combine two different d

Annotation methods

Category	Function	Description
GRanges*	`transcripts()`, `exons()`, `cds()`	Features (transcripts, exons, coding sequence) as `GRanges`.
	`transcriptsBy()` , `exonsBy()` `cdsBy()`	Features group by gene, transcript, etc., as `GRangesList`.

Biomart

- Biomart R package, `biomaRt`, workflow:
  - Discover and select a mart and dataset,
  - Select filters, which IDs to convert from
  - Select attributes, which IDs to convert to
  - Run the query
- Biomart has a web interface, operating on the same principles

https://bioconductor.org/packages/biomaRt

http://bioconductor.org/packages/release/bioc/vignettes/biomaRt/inst/doc/biomaRt.html#selecting-a-biomart-database-and-dataset

http://bioconductor.org/packages/release/bioc/vignettes/biomaRt/inst/doc/biomaRt.html#annotate-a-set-of-entrezgene-identifiers-with-go-annotation

http://www.ensembl.org/biomart

KEGG

- KEGG: Kyoto Encyclopedia of Genes and Genomes
- KEGG API R package, `KEGGREST`
    - Essential operations outlined in the vignette

http://www.genome.jp/kegg/pathway.html

https://bioconductor.org/packages/KEGGREST

http://bioconductor.org/packages/release/bioc/vignettes/KEGGREST/inst/doc/KEGGREST-vignette.html

PSICQUIC

- `PSICQUIC` (the Proteomics Standard Initiative Common QUery InterfaCe) - standardized access to molecular interaction databases.
- Protein-protein interaction databases like “BioGrid”, “Intact”, “Reactome”, “STRING”, “BIND”
- Supports the molecular interaction query language (MIQL)

https://bioconductor.org/packages/release/bioc/html/PSICQUIC.html

AnnotationHub

- `AnnotationHub` package - curated database of large-scale whole-genome resources, e.g., regulatory elements from the Roadmap Epigenomics project, Ensembl GTF and FASTA files for model and other organisms, and the NHLBI `grasp2db` data base of GWAS results. Examples of use include:
  - Easily access and import Roadmap Epigenomics files.
  - `liftOver` genomic range-based annotations from one coordinate system (e.g, `hg19`) to another (e.g., `GRCh38`).
  - Create `TranscriptDb` and `BSgenome`-style annotation resources 'on the fly' for a diverse set of organisms.
  - Programmatically access the genomic coordiantes of clinically relevant variants cataloged in dbSNP.
- Related packages: `ExperimentHub` - curated data sets

https://bioconductor.org/packages/AnnotationHub

AnnotationHub HOW-TOs, http://bioconductor.org/packages/devel/bioc/vignettes/AnnotationHub/inst/doc/AnnotationHub-HOWTO.html

https://bioconductor.org/packages/ExperimentHub

Domain-specific packages

- Important packages for analysis of **differential expression** include `edgeR` and `DESeq2`; both have excellent vignettes for exploration.

http://bioconductor.org/packages/edgeR

http://bioconductor.org/packages/DESeq2

- Popular **ChIP-seq** packages include `DiffBind` and `csaw` for comparison of peaks across samples, `ChIPQC` for quality assessment, and `ChIPpeakAnno` and `ChIPseeker` for annotating results (e.g., discovering nearby genes).

http://bioconductor.org/packages/DiffBind

http://bioconductor.org/packages/csaw

http://bioconductor.org/packages/ChIPQC

http://bioconductor.org/packages/ChIPpeakAnno

http://bioconductor.org/packages/ChIPseeker

Domain-specific packages

- Working with called variants (VCF files) is facilitated by packages such as `VariantAnnotation`, `VariantFiltering` and `ensemblVEP`.
- Packages for calling variants include, e.g., `h5vc` and `VariantTools`.

https://bioconductor.org/packages/release/bioc/html/VariantAnnotation.html

http://bioconductor.org/packages/VariantFiltering.html

https://bioconductor.org/packages/release/bioc/html/ensemblVEP.html

https://bioconductor.org/packages/release/bioc/html/h5vc.html

https://bioconductor.org/packages/release/bioc/html/VariantTools.html

Domain-specific packages

- **Single-cell 'omics'** are increasingly important. From the `biocView` page, enter 'single cell' in the 'search table' field.
- Several packages identify **copy number variants** from sequence data, including `cn.mops`. The `CNTools` package provides some useful facilities for comparison of segments across samples.
- **Microbiome and metagenomic** analysis is facilitated by packages such as `phyloseq` and `metagenomeSeq`.
- **Metabolomics, chemoinformatics, image analysis**, and many other high-throughput analysis domains are also represented in *Bioconductor*; explore these via `biocViews` and title searches.

https://bioconductor.org/packages/release/BiocViews.html#___Software

https://bioconductor.org/packages/release/bioc/html/cn.mops.html

http://bioconductor.org/packages/CNTools

http://bioconductor.org/packages/phyloseq

http://bioconductor.org/packages/metagenomeSeq

Visiualization

A number of *Bioconductor* packages help with visualization and reporting, in addition to functions provided by individual packages.

- `Gviz` provides a track-like visualization of genomic regions.
- `ComplexHeatmap` does an amazing job of all sorts of heatmaps, including OncoPrint-style summaries.
- `ReportingTools` provides a flexible way to generate static and dynamic HTML-based reports.

http://bioconductor.org/packages/Gviz

http://bioconductor.org/packages/ComplexHeatmap

http://bioconductor.org/packages/ReportingTools

Working with 'big data'

- Much Bioinformatic data is very large and often cannot fit into memory
- Several general strategies for working with large data

Restriction to specific genomic regions

- e.g., `ScanBamParam()` limits input to desired data at specific genomic ranges

Iteration over pieces of genomic data

- e.g., `yieldSize` argument of `BamFile()`, or `FastqStreamer()` allows iteration through large files.

Working with 'big data'

Compression

- Genomic vectors represented as `Rle` (run-length encoding) class
- Lists e.g., `GRangesList` are efficiently maintain the illusion that vector elements are grouped.

Parallel processing

- e.g., via `BiocParallel` package

https://bioconductor.org/packages/release/bioc/html/BiocParallel.html

Lawrence, M and Morgan, M. Scalable Genomic Computing and Visualization with *R* and *Bioconductor*. Statistical Science 29 (2) (2014), 214-226.

Code optimization

- `aprof` - Amdahl's Profiler, Directed Optimization Made Easy.
- `profvis` - Visualize R profiling data.
- `microbenchmark` - Accurate Timing Functions.

http://cran.r-project.org/web/packages/aprof/index.html

https://rstudio.github.io/profvis. Examples: https://rpubs.com/wch/123888

http://cran.r-project.org/web/packages/microbenchmark/index.html

Summary

- *Bioconductor* is a large collection of R packages for the analysis and comprehension of high-throughput genomic data.
- *Bioconductor* relies on formal classes to represent genomic data, so it is important to develop a rudimentary comfort with classes, including seeking help for classes and methods.
- *Bioconductor* uses vignettes to augment traditional help pages; these can be very valuable in illustrating overall package use.