Atelier Chip-Seq

Elodie Darbo, INSERM U1312, Bordeaux
Pascal Martin, INRAE UMR1332, Bordeaux

École de bioinformatique AVIESAN-IFB-Inserm n2 2025

Contents

- Introduction / the datasets
- File formats and corresponding objects in R/Bioconductor
- Working with biological sequences in R: the Biostrings package
- Working with genomic coordinates in R: the GenomicRanges package
- Annotations in R
- Heatmap and average profiles
- How to evaluate functional enrichment?

Connecting to RStudio (or Jupyter)

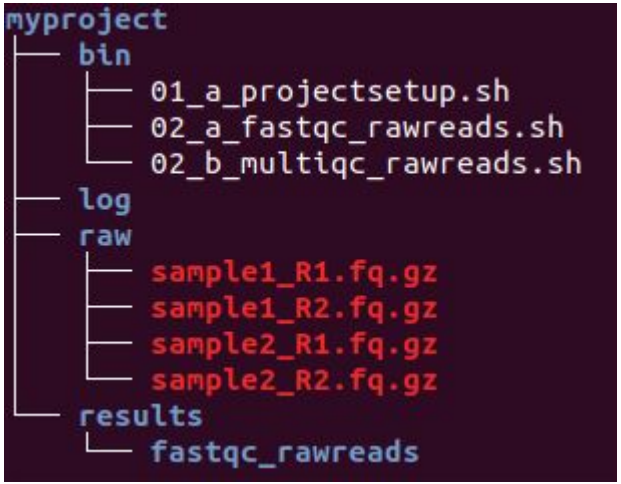
- https://ondemand.cluster.france-bioinformatique.fr/
- Log in
- Select: RStudio Server Core
- Select:
  - Your R version
  - Your account / projectID
  - The number of CPUs (at least 4)
  - The amount of memory (at least 32G)
  - Number of hours (depends how long you plan to work in RStudio)
- Select: Launch
- Wait for the job to be assigned and click on “Connect to RStudio Server”
- Change your working directory (your home dir by default)

More info: https://ifb-elixirfr.gitlab.io/cluster/doc/software/openondemand/starting_rstudio_server/

Before we start:

- There are numerous ways to analyze NGS data, almost as many as there are bioinformaticians, especially for ChIP-seq and alike. Everyone has its own tool set / code lines in R/python/etc. We propose you **one** way to do some “advanced” analyses on ChIP-seq data.
- **Be reproducible!** Keep track of all command lines that you run. You can for example create a simple text file with all your commands, or create a markdown / Rmarkdown file with all your commands
- **Comment your script**! Comments in an R script start with #. Use them extensively, your future you will thank you.

A few tips to start your project:



- Keep your project folder organized
- Keep the top of your R scripts organized:
  - Create a variable (e.g. `workdir`) with your project path
  - Use `setwd(workdir)`
  - Add some options if you want: `options(width=160)`
  - Place all `library()` / `source()` calls at the top of the script
- Save your script frequently (especially if you work with RStudio and/or on a server)
- Save / load individual R objects using `saveRDS` and `loadRDS`
- Use git / versioning
- Use bioconductor, your web browser, your favorite LLM before re-inventing the wheel
- Check stackoverflow, Biostars, SEQanswers,...

Basics and Tips in R

Introduction / the datasets

eLIFE
elifesciences.org

RESEARCH ARTICLE

Transcription factor MITF and remodeller BRG1 define chromatin organisation at regulatory elements in melanoma cells

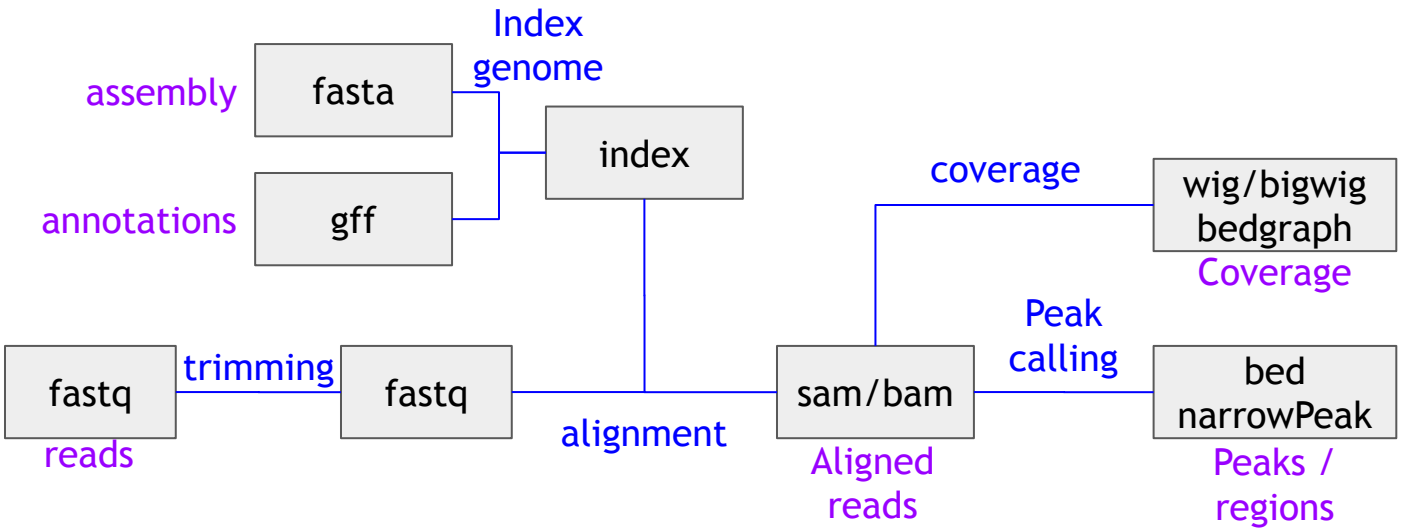
Patrick Laurette[1†], Thomas Strub[1,2†], Dana Koludrovic[1], Céline Keime[1], Stéphanie Le Gras[1], Hannah Seberg[3], Eric Van Otterloo[3], Hana Imrichova[4], Robert Siddaway[5], Stein Aerts[4], Robert A Cornell[3], Gabrielle Mengus[1], Irwin Davidson[1*]

The datasets:

- 3 ChIP-seq datasets: MITF, SOX10 and BRG1
  - Peak files (narrowPeak format)
  - Coverage files (bigwig format)

- RNA-seq dataset
  - data.frame with the results from the DGE analysis

Where do these files come from? (typical ChIP-seq pipeline)



File formats and corresponding R/BioC objects

File formats and objects in R/BioC

File format	Package	Container
fasta	Biostrings	XString / XStringSet
fastq	ShortRead	ShortReadQ
sam/bam	GenomicAlignments	GAlignments / GAlignmentPairs
bed / narrowPeak	GenomicRanges	GRanges / GRangesList
bedgraph / wig / bigwig	GenomicRanges	RleList
gtf / gff	GenomicFeatures / txdbmaker	GRanges / TxDb

See also:

- library(BSgenome) for whole genomes
- library(Rsamtools) interface to samtools
- library(GenomicFiles) for many/large files
- library(rtracklayer) for import/export

The fasta format (e.g. genome sequence):

Start symbol

Sequence ID (no white space)

Description (white space OK)

Sequence

https://en.wikipedia.org/wiki/FASTA_format

```
>NM_028020| Mus musculus integrator complex subunit 11, mRNA
GCGCAGTGCGGAGGGACTCGCCGCGGCCGGGCGTCTGATGTGAGCGCGCGCGGGGTCTGCCTGCGGCTCT
CCTGCGGGCGGTGCACGCGAGCCGTGGAGCCGTGGTTCGAGGCGACCTCCCTGACCATGCCCGAGATTAG
GGTCACTCCCTTGGGGGCTGGCCAGGATGTAGGCCGAAGCTGCATCTTGGTCTCCATTTCGGGGAAGAAT
GTCATGTTGGACTGTGGGATGCACATGGGCTACAATGATGACAGGCGCTTTCCTGACTTTTCTTACATCA
CCCAGAGTGGCCGCCTGACTGACTTTCTGGACTGTGTGATCATCAGCCACTTCCACCTGGACCATTGTGG
GGCACTCCCCTACTTCAGTGAAATGGTGGGCTACGATGGACCCATCTATATGACCCATCCTACCCAGGCC
ATCTGCCCCATCCTGTTGGAAGACTACCGCAAGATTGCAGTGGACAAGAAGGGCGAGGCCAATTTCTTCA
CTTCTCAGATGATCAAAGACTGTATGAAGAAGGTGGTGGCTGTTCACCTGCATCAGACAGTTCAGGTGGA
```

Extension	Alphabet	Usage
.fasta, .fa	all	Generic
.fna	DNA	Large genomic regions (contigs, chromosomes, ...)
.ffn	DNA	Coding sequence
.faa	Protein	Amino acid sequence
.frn	DNA	Non coding RNA

Working with biological sequences in R: the Biostrings package

In R/BioC: Xstring / XstringSet

- Containers — library(Biostrings)
  - XString – BString, DNAString, RNAString, AAString
  - XStringSet – multiple sequences
  - XStringViews

```
seq=readDNAStringSet("mySeq.fasta")
#  A DNAStringSet instance of length 3
#    width seq                                names
#[1]  7454 GGAGACCCACAGCCACTGG...TTAAAAATTAAAAAAAAAA gi|164663878|ref|...
#[2]  7216 GGAGGCAGCCGCTTACGCC...TTAAAAATTAAAAAAAAAA gi|164663879|ref|...
#[3]  3240 GCCGCAGGCCGCGGCGGAC...AAAAAAAAAAAAAAAAAAA gi|89001112|ref|N...

mySeq = DNAString("AGTCTTGTGGTCGATGATTGGTTT")
#  24-letter "DNAString" instance
#seq: AGTCTTGTGGTCGATGATTGGTTT
```

For whole genomes: library(BSgenome)

Manipulate and analyze sequence data

- Handles large datasets : efficient storage of DNA, RNA, and protein sequences
- Tools for sequence operations: pattern detection, translation, alignment, etc.
- Identify key genetic features and compare sequences: Pattern matching, motif searching, basic sequence alignment
- Deeper analysis of evolution, function, and structure

mySeq.fasta

```
>NM_028020| Mus musculus integrator complex subunit 11, mRNA
GCGCAGTGCGGAGGGACTCGCCGCGGCCGGGCGTCTGATGTGAGCGCGCGCGGGGTCTGCCTGCG
CCTGCGGGCGGTGCACGCGAGCCGTGGAGCCGTGGTTCGAGGCGACCTCCCTGACCATGCCCGAG
GGTCACTCCCTTGGGGGCTGGCCAGGATGTAGGCCGAAGCTGCATCTTGGTCTCCATTTCGGGGA
GTCATGTTGGACTGTGGGATGCACATGGGCTACAATGATGACAGGCGCTTTCCTGACTTTTCTTA
CCCAGAGTGGCCGCCTGACTGACTTTCTGGACTGTGTGATCATCAGCCACTTCCACCTGGACCAT
```

reverseComplement

statistics

letterFrequency

alphabetFrequency

```
seq=readDNAStringSet("mySeq.fasta")
```

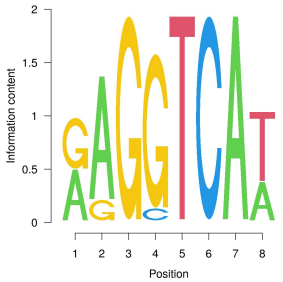
Extract sequences

Views

subseq

Predict occurrences

matchPWM



seqLogo::seqLogo

```
          1 2 3 4         5 6         7   8
A 0.6428571 0 0 1 0.00000000 0 0.0000000 0.0
C 0.0000000 0 0 0 0.92857143 1 0.1428571 0.5
G 0.0000000 0 1 0 0.07142857 0 0.0000000 0.0
T 0.3571429 1 0 0 0.00000000 0 0.8571429 0.5
```

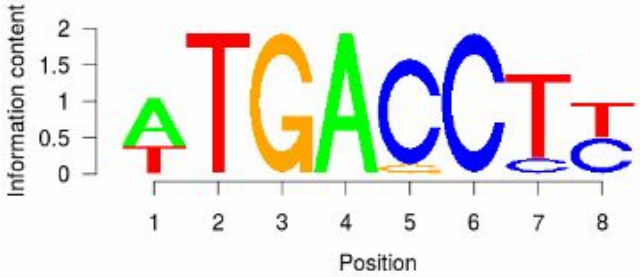
```
EcRMotifs=MotifDb::query(MotifDb,"EcR")
```

max score = .12 + .13 + .13 + .13 + .13 + .13 + .13 + .11 **= 1**

PWM:

	1	2	3	4	5	6	7	8
A	0.12	0.00	0.00	0.13	0.00	0.00	0.00	0.00
C	0.00	0.00	0.00	0.00	0.13	0.13	0.07	0.11
G	0.00	0.00	0.13	0.00	0.05	0.00	0.00	0.00
T	0.10	0.13	0.00	0.00	0.00	0.00	0.13	0.11

Sequence logo:



score = .10 +.13 +.13 +.13+.05+.13+ .07 +.11 **= 0.85** **= 85 % of max score**

...a c **T T G A G C C T** t g c...

DNA sequence subject

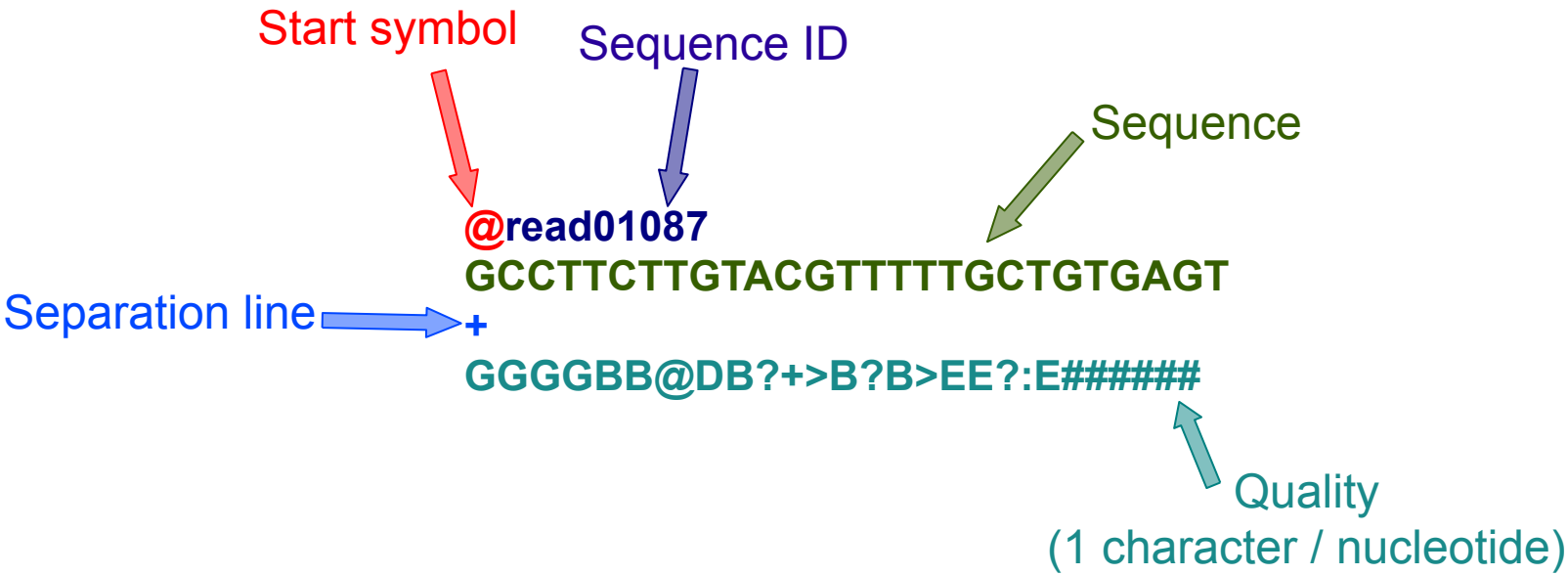
Using matchPWM function with parameter min.score="80%" => motif found
with parameter min.score="90%" => NO motif

Discover and manipulate *XStringSet* objects

The fastq format (e.g. reads):

Extension : .fq ou .fastq

https://en.wikipedia.org/wiki/FASTQ_format

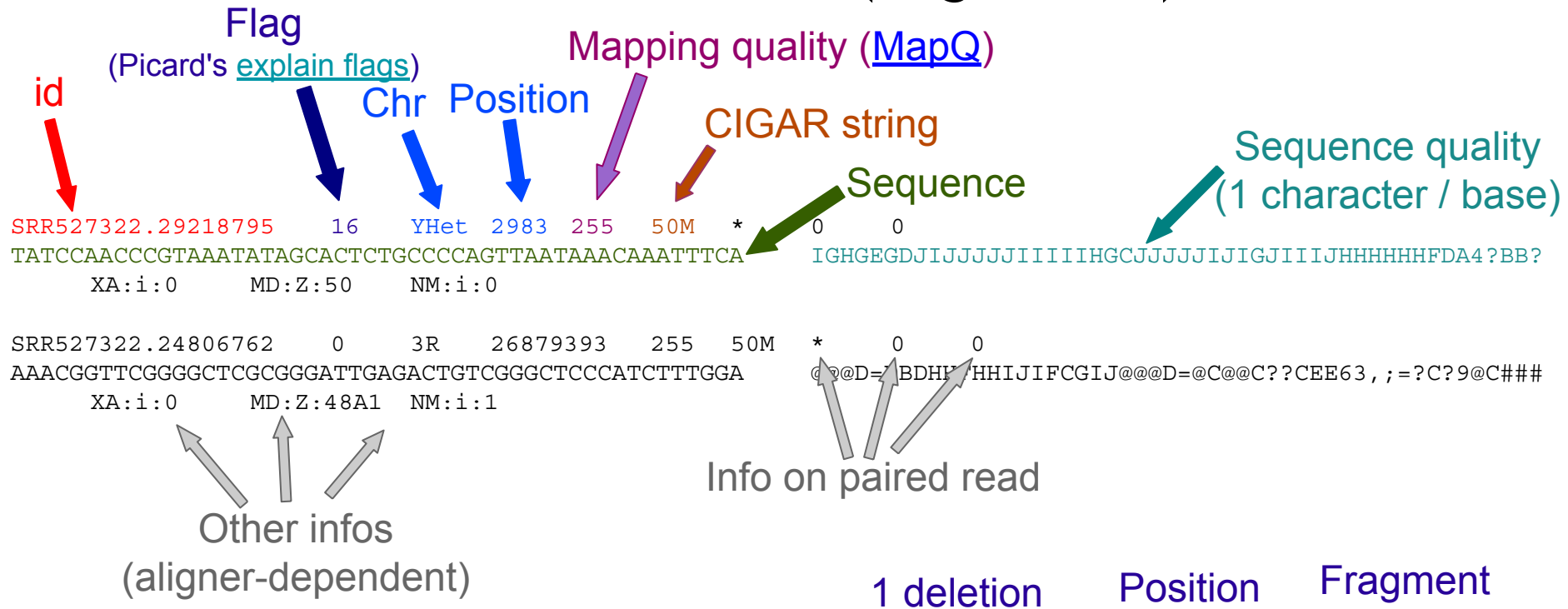


Quality values encoding (ASCII code, Phred+33) :

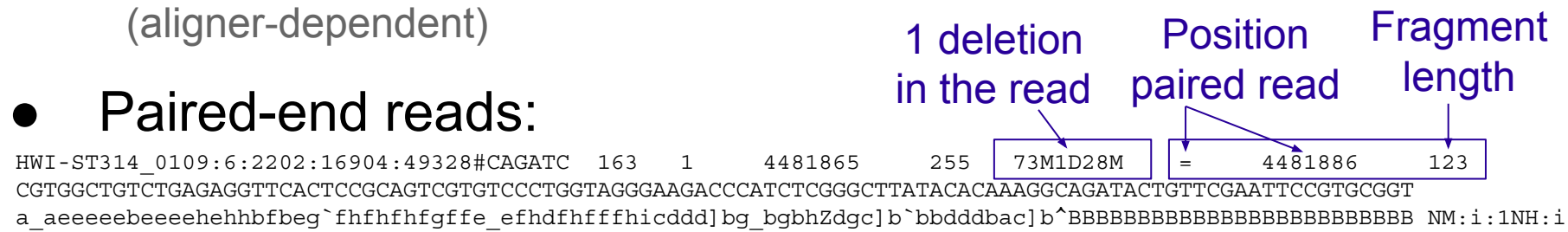
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJ

Q0	Q10	Q20	Q30	Q40
bad	*average*	*ok*	*good*	*excellent*

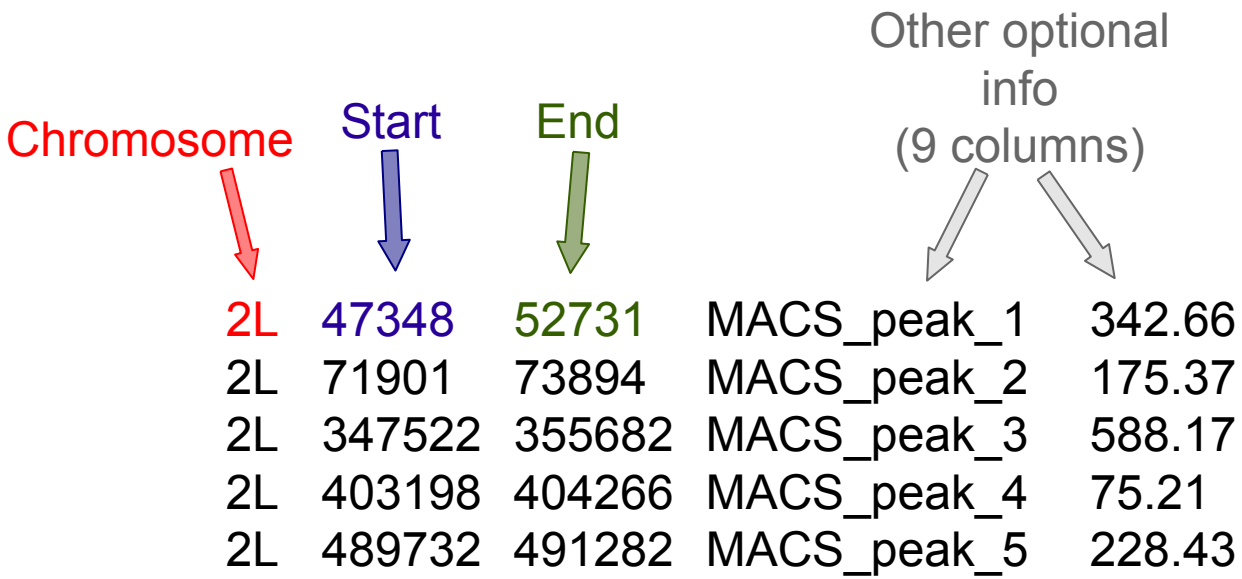
The sam/bam format (alignments):



- Paired-end reads:



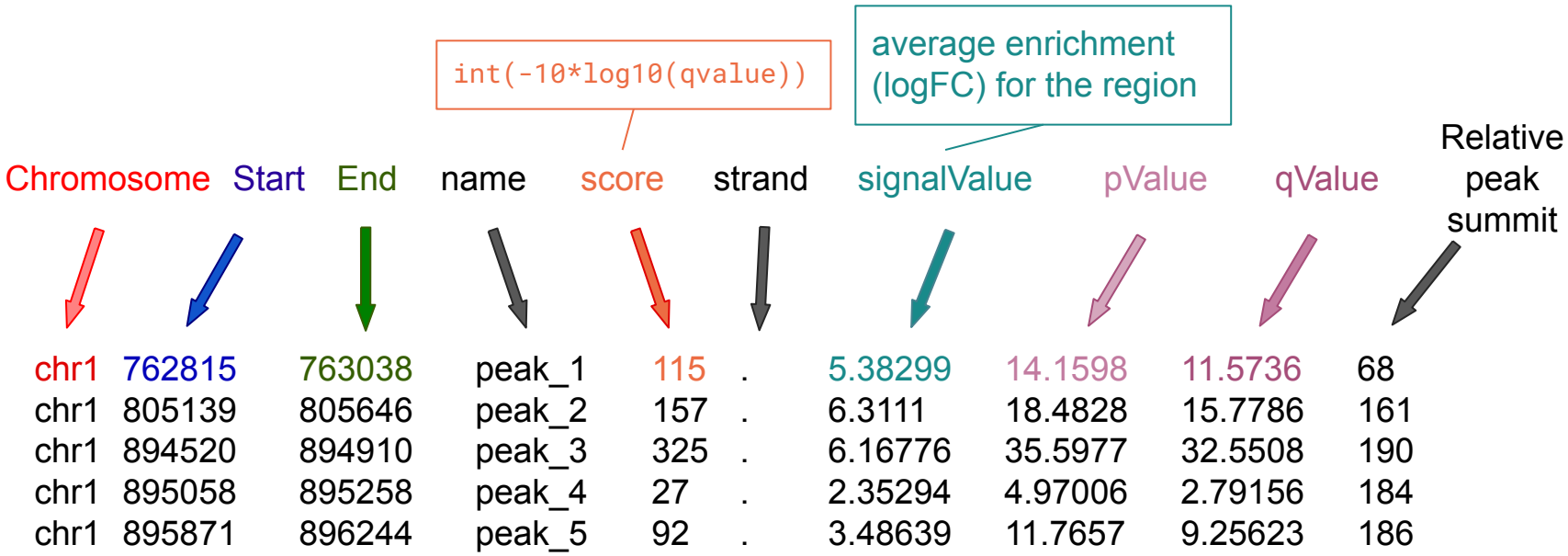
The bed format (genomic intervals/regions):



Column	Content
1	chrom
2	chromStart
3	chromEnd
4	name
5	score
6	strand
7	thickStart
8	thickEnd
9	itemRgb
10	blockCount
11	blockSizes
12	blockStarts

https://en.wikipedia.org/wiki/BED_(file_format)
https://genome.ucsc.edu/FAQ/FAQformat.html

The narrowPeak format (genomic intervals/regions):



chr1	762815	763038	peak_1	115	.	5.38299	14.1598	11.5736	68
chr1	805139	805646	peak_2	157	.	6.3111	18.4828	15.7786	161
chr1	894520	894910	peak_3	325	.	6.16776	35.5977	32.5508	190
chr1	895058	895258	peak_4	27	.	2.35294	4.97006	2.79156	184
chr1	895871	896244	peak_5	92	.	3.48639	11.7657	9.25623	186

The gtf/gff format (annotations):

Chrom	source	feature	Start	End	score	strand	frame	attributes
1	havana	gene	11869	14409	.	+	.	gene_id "ENSG00000223972"; gene_version "5"; gene_name "DDX11L1"; gene_source "havana"; gene_biotype "transcribed_unprocessed_pseudogene";
1	havana	transcript	11869	14409	.	+	.	gene_id "ENSG00000223972"; gene_version "5"; transcript_id "ENST00000456328"; transcript_version "2"; gene_name "DDX11L1"; gene_source "havana"; gene_biotype "transcribed_unprocessed_pseudogene"; transcript_name "DDX11L1-202"; transcript_source "havana"; transcript_biotype "processed_transcript"; tag "basic"; transcript_support_level "1";
1	havana	exon	11869	12227	.	+	.	gene_id "ENSG00000223972"; gene_version "5"; transcript_id "ENST00000456328"; transcript_version "2"; exon_number "1"; gene_name "DDX11L1"; gene_source "havana"; gene_biotype "transcribed_unprocessed_pseudogene"; transcript_name "DDX11L1-202"; transcript_source "havana"; transcript_biotype "processed_transcript"; exon_id "ENSE00002234944"; exon_version "1"; tag "basic"; transcript_support_level "1";

Working with genomic coordinates in R: the GenomicRanges package

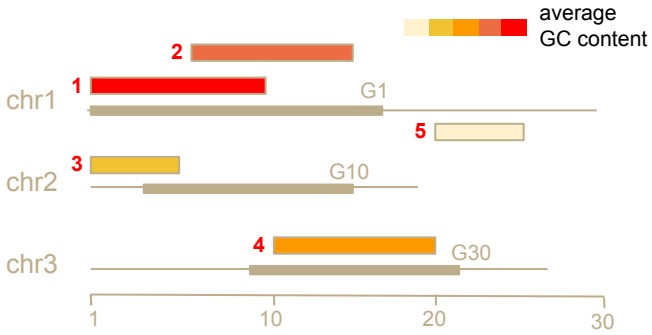
Genomic ranges are at the core of genomic analyses

- A large fraction of the data that we manipulate are genomic intervals (defined as Chr:start-end:strand)

- E.g. peaks, annotations, coverages are all genomic intervals, sometimes with quantitative values associated with them

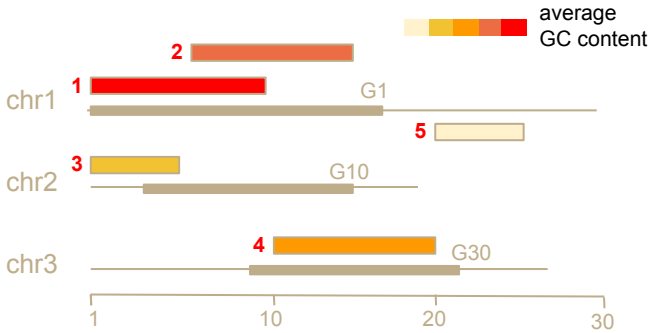
- Many biological questions reflect range-based queries

Genomic intervals with *GenomicRanges*



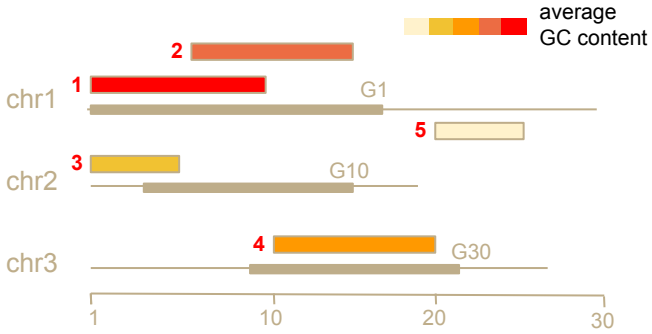
peaks ← GRanges(seqnames, range=IRanges(start,end), strand, gene, GC)

Build a *GRanges* object



peaks ← GRanges(seqnames, range=IRanges(start,end), strand, gene, GC)

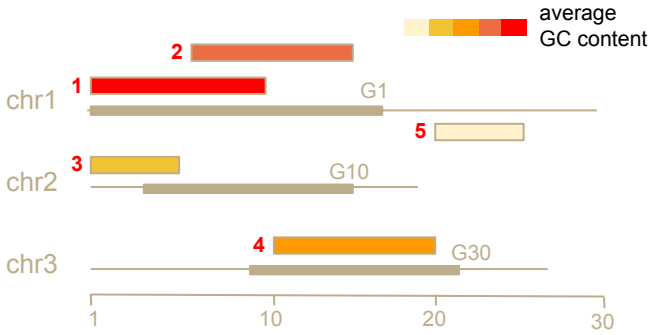
Build a *GRanges* object



	seqnames <Rle>	start <IRanges>	end <IRanges>	strand <Rle>	gene <character>	GC <numeric>
1	chr1	1	10	+	G1	40
2	chr1	6	15	+	G1	35
3	chr2	1	5	+	G10	26
4	chr3	10	20	*	G30	30
5	chr1	20	25	-	NA	23

peaks ← GRanges(seqnames, range=IRanges(start,end), strand, gene, GC)

GRanges: Extract information



numeric vectors

start()

end()

DataFrame with 10 rows and 2 columns

values()
mcols()

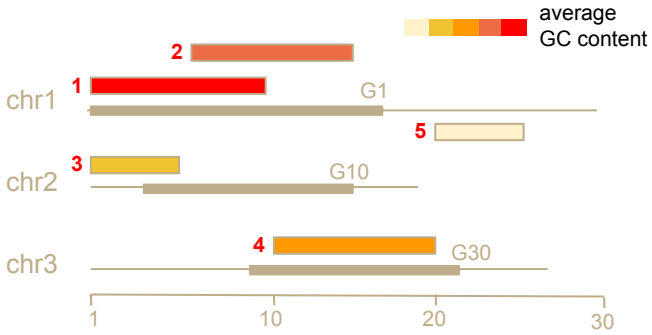
	seqnames <Rle>	start <IRanges>	end	strand <Rle>	gene <character>	GC <numeric>
1	chr1	1	10	+	G1	40
2	chr1	6	15	+	G1	35
3	chr2	1	5	+	G10	26
4	chr3	10	20	*	G30	30
5	chr1	20	25	-	NA	23

range()

GRanges object with 7 ranges and 0 metadata columns

peaks ← GRanges(seqnames, range=IRanges(start,end), strand, gene, GC)

GRanges: Extract information



numeric vectors

width() start() end()

DataFrame with 10 rows and 2 columns

values() mcols()

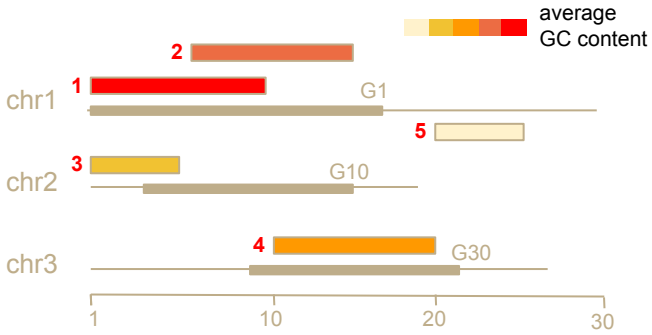
	seqnames <Rle>	start <IRanges>	end	strand <Rle>	gene <character>	GC <numeric>
1	chr1	1	10	+	G1	40
2	chr1	6	15	+	G1	35
3	chr2	1	5	+	G10	26
4	chr3	10	20	*	G30	30
5	chr1	20	25	-	NA	23

range()

GRanges object with 7 ranges and 0 metadata columns

peaks ← GRanges(seqnames, range=IRanges(start,end), strand, gene, GC)

GRanges: Extract information



numeric vectors

width()

start()

end()

	seqnames <Rle>	start <IRanges>	end <IRanges>	strand <Rle>	gene <character>	GC <numeric>
1	chr1	1	10	+	G1	40
2	chr1	6	15	+	G1	35
3	chr2	1	5	+	G10	26
4	chr3	10	20	*	G30	30
5	chr1	20	25	-	NA	23

DataFrame with 10 rows and 2 columns

values()
mcols()

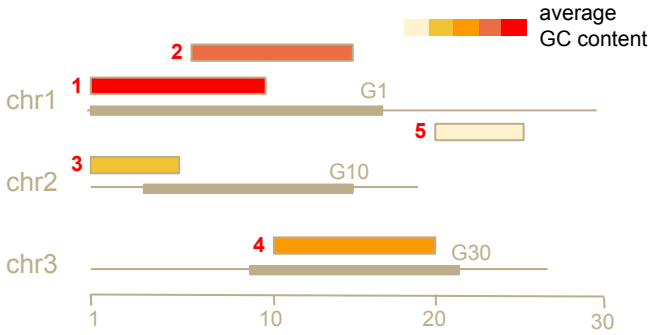
peaks$GC

range()

GRanges object with 7 ranges and 0 metadata columns

peaks ← GRanges(seqnames, range=IRanges(start,end), strand, gene, GC)

GRanges: Extract information



character vectors

"chr1"	"chr1"	"chr2"	"chr3"	"chr1"

as.vector()

character-Rle of length 5 with 4 runs

Lengths	2	1	1	1
Values	"chr1"	"chr2"	"chr3"	"chr1"

numeric vectors

width() start() end()

	seqnames <Rle>	start <IRanges>	end <IRanges>	strand <Rle>	gene <character>	GC <numeric>
1	chr1	1	10	+	G1	40
2	chr1	6	15	+	G1	35
3	chr2	1	5	+	G10	26
4	chr3	10	20	*	G30	30
5	chr1	20	25	-	NA	23

DataFrame with 10 rows and 2 columns

values() mcols()

peaks$GC

seqnames() range() strand()

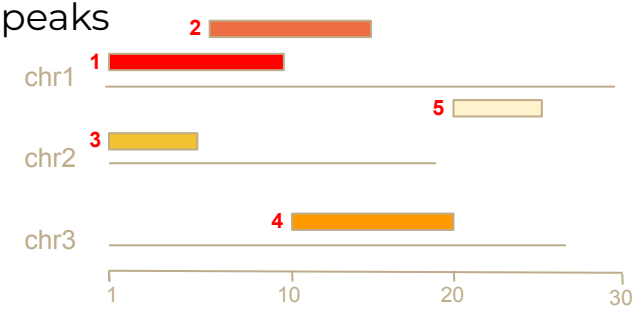
GRanges object with 7 ranges and 0 metadata columns

character-Rle of length 5 with 3 runs

Lengths	3	1	1
Values	"+"	"*"	"-"

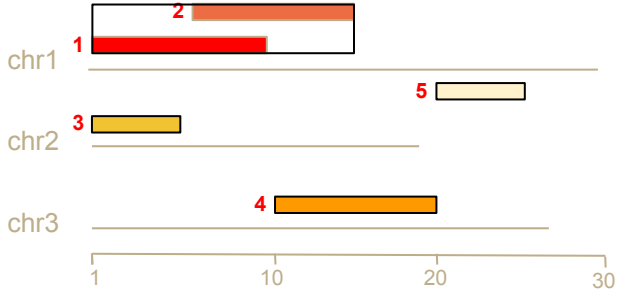
GRanges: Manipulate objects

peaks



GRanges object with 5 ranges and 1 metadata columns

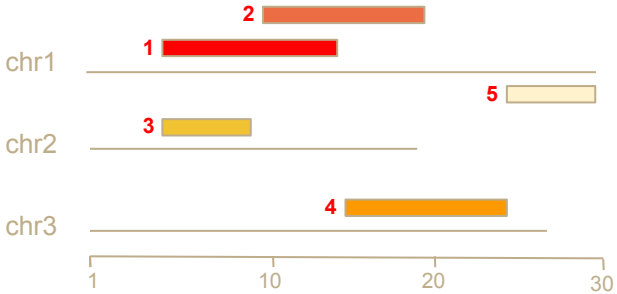
reduce(peaks)



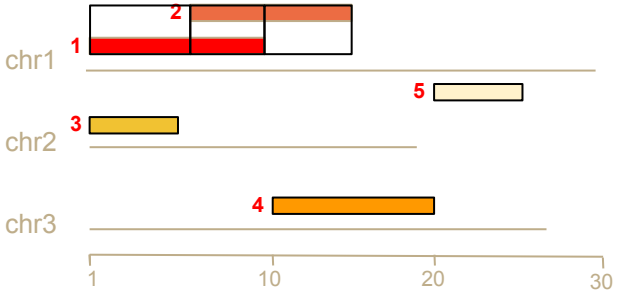
GRanges object with 4 ranges and 1 metadata columns

disjoin(peaks)

shift(peaks,5)

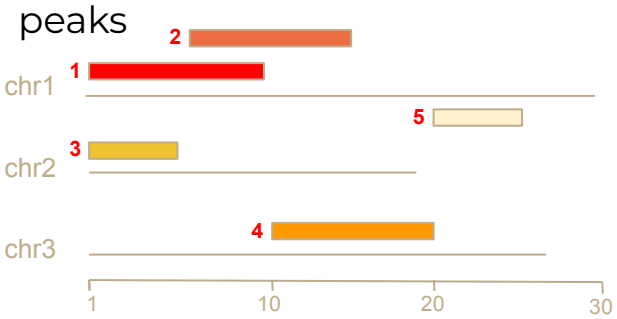


GRanges object with 5 ranges and 1 metadata columns

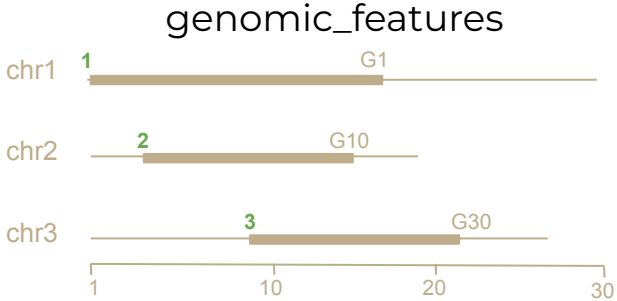


GRanges object with 6 ranges and 1 metadata columns

Detect overlapping elements between 2 *GRanges*



GRanges object with 5 ranges and 1 metadata columns



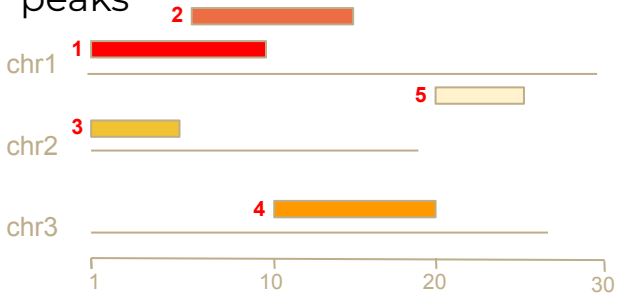
GRanges object with 3 ranges and 1 metadata columns

Detect overlapping elements between 2 *GRanges*

findOverlaps(peaks,genomic_features)

Hits object with 4 hits and 0 metadata columns

peaks

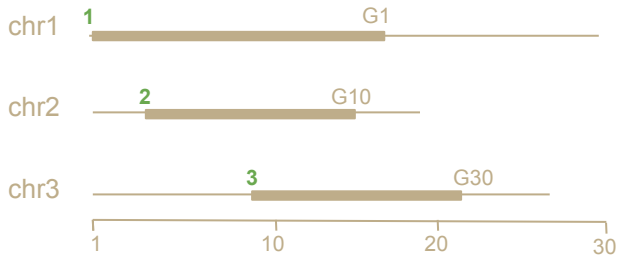


GRanges object with 5 ranges and 1 metadata columns

	queryHits <integer>	subjectHits <integer>
1	1	1
2	2	1
3	3	2
4	4	3

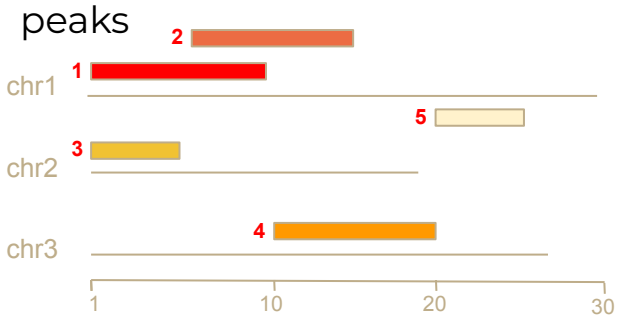
queryLength: 5 / subjectLength: 3

genomic_features

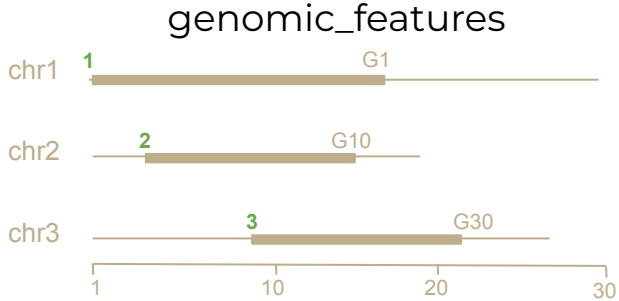


GRanges object with 3 ranges and 1 metadata columns

Detect overlapping elements between 2 *GRanges*



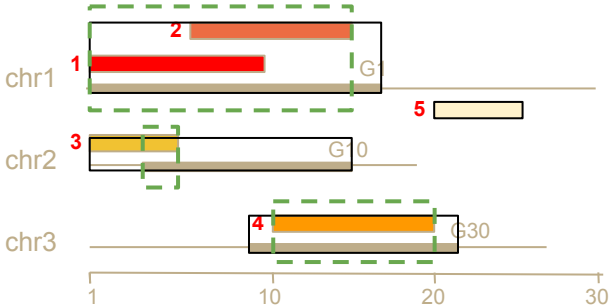
GRanges object with 5 ranges and 1 metadata columns



GRanges object with 3 ranges and 1 metadata columns

intersect(peaks,genomic_features)

GRanges object with 3 ranges and 0 metadata columns

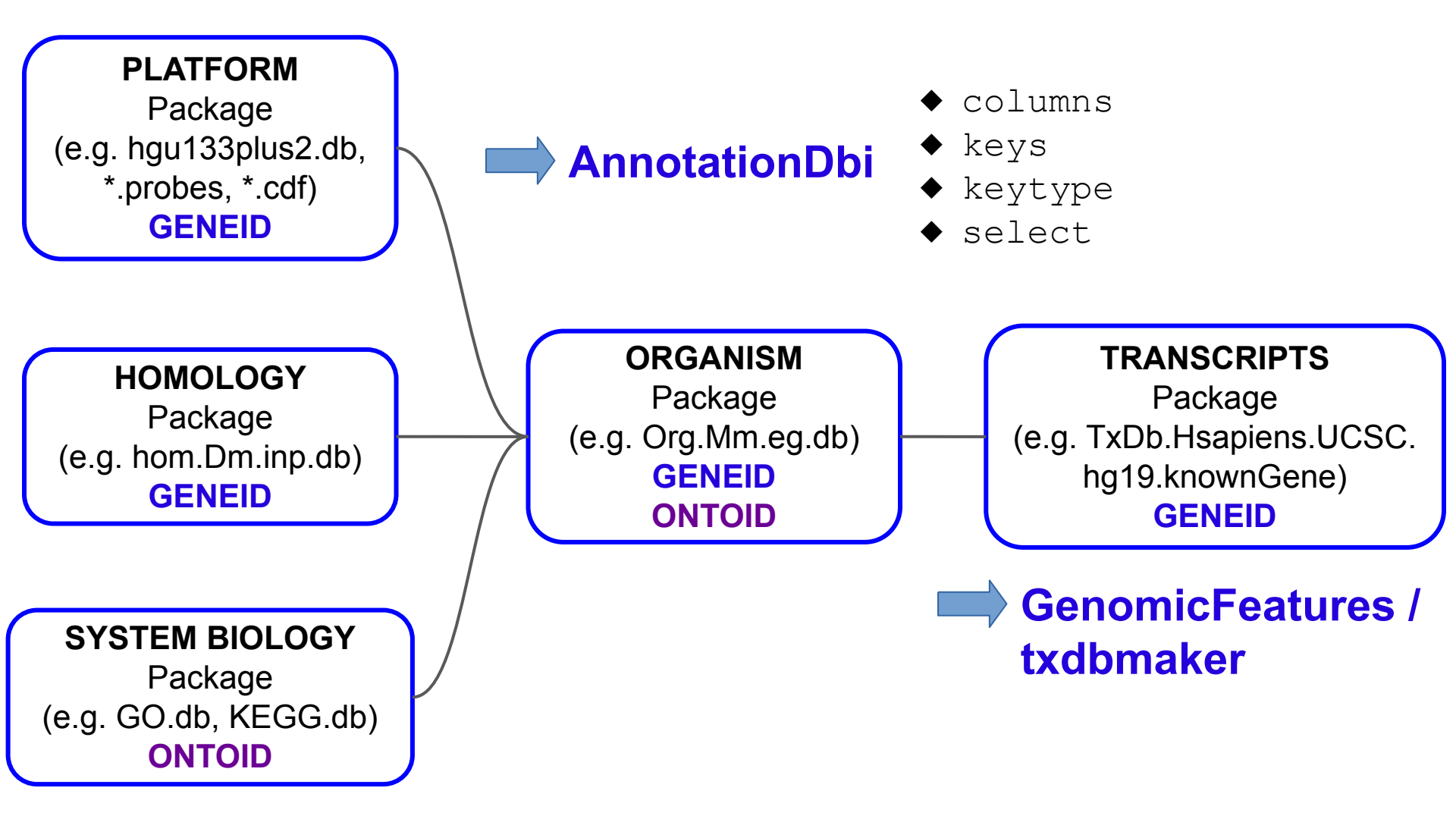


union(peaks,genomic_features)

GRanges object with 4 ranges and 0 metadata columns

Discover and manipulate *GenomicRanges* objects

Annotations in R


PLATFORM
Package
(e.g. hgu133plus2.db,
*.probes, *.cdf)
GENEID
HOMOLOGY
Package
(e.g. hom.Dm.inp.db)
GENEID
SYSTEM BIOLOGY
Package
(e.g. GO.db, KEGG.db)
ONTOID
AnnotationDbi
columns
keys
keytype
select
ORGANISM
Package
(e.g. Org.Mm.eg.db)
GENEID
ONTOID
TRANSCRIPTS
Package
(e.g. TxDb.Hsapiens.UCSC.
hg19.knownGene)
GENEID
GenomicFeatures /
txdbmaker

Annotation objects: *GenomicFeatures, BSgenome*

By species and genome version

Genomic sequences

Transcript-oriented annotations (genes, transcripts, exons, etc.)

Functional annotations of genes

GenomicFeatures: Build a TxDB object “from scratch”

makeTxDb(transcripts, splicings, genes)

data.frame with 6 columns and a row per exon

tx_id	exon_rank	exon_start	exon_end	cds_start	cds_end
1	1	1	999	1	999
2	1	2001	2085	2022	2085
2	2	2101	2144	2101	2144
2	3	2131	2199	2131	2193
3	1	2001	2085	NA	NA
3	2	2131	2199	NA	NA

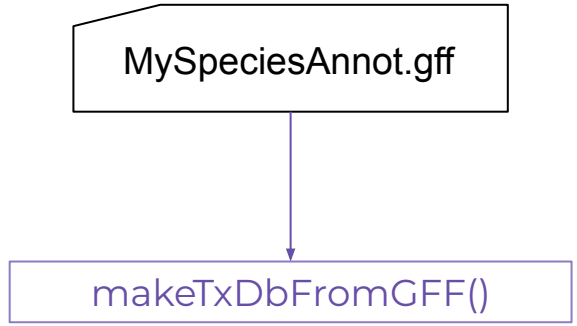
data.frame with 5 columns and a row per transcript

tx_id	tx_chrom	tx_strand	tx_start	tx_end
1	chr1	-	1	999
2	chr1	+	2001	2199
3	chr1	+	2001	2199

data.frame with 2 columns and a row per gene

tx_id	gene_id
1	gene1
2	gene2
3	gene3

Txdbmaker: Build a TxDB from a gtf/gff file containing annotations



bio mart

(http://www.biomart.org/)
(http://www.ensembl.org/biomart/martview/)

makeTxDbFromBiomart()

Needs internet connection

WARNING

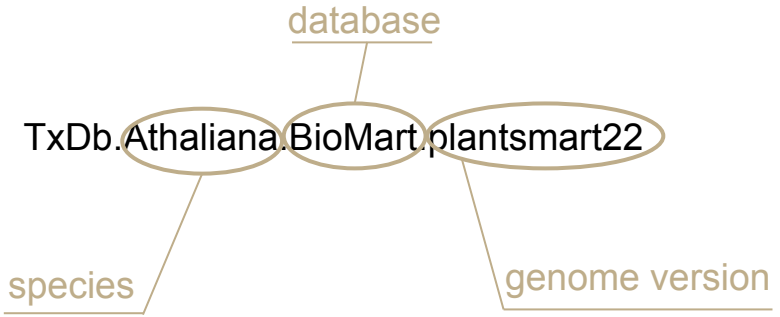
UCSC

makeTxDbFromUCSC()

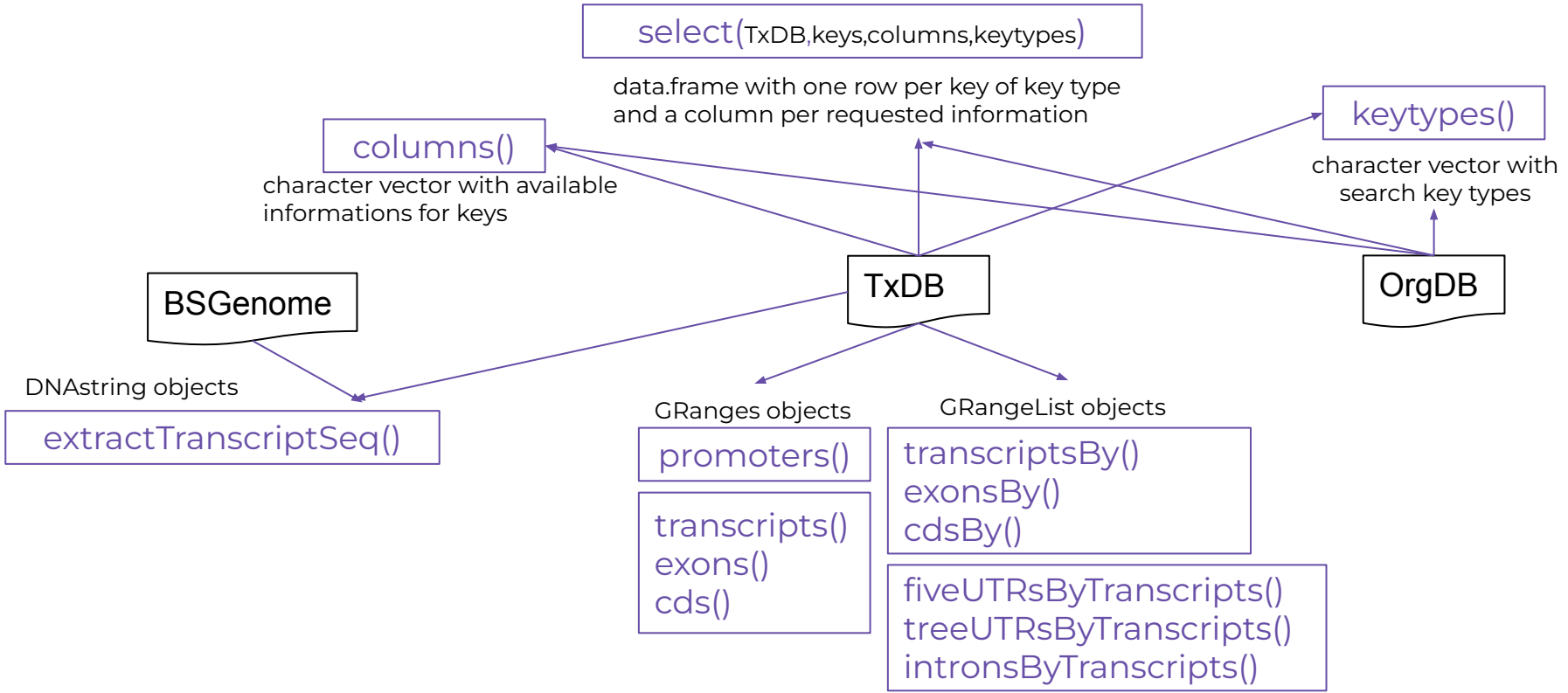
GenomicFeatures: TxDB objects "Ready-to-use"

There is a wide collection of TxDB objects already available in BioConductor.
Check the corresponding BioCView:
https://www.bioconductor.org/packages/release/BiocViews.html#___TxDb

TxDb.Athaliana.BioMart.plantsmart25, TxDb.Athaliana.BioMart.plantsmart28,
TxDb.Btaurus.UCSC.bosTau8.refGene, TxDb.Celegans.UCSC.ce11.refGene,
TxDb.Celegans.UCSC.ce6.ensGene, TxDb.Cfamiliaris.UCSC.canFam3.refGene,
TxDb.Dmelanogaster.UCSC.dm3.ensGene,
TxDb.Dmelanogaster.UCSC.dm6.ensGene, TxDb.Drerio.UCSC.danRer10.refGene,
TxDb.Ggallus.UCSC.galGal4.refGene, TxDb.Ggallus.UCSC.galGal5.refGene,
TxDb.Hsapiens.BioMart.igis, TxDb.Hsapiens.UCSC.hg18.knownGene,
TxDb.Hsapiens.UCSC.hg19.knownGene,
TxDb.Hsapiens.UCSC.hg19.lincRNAsTranscripts,
TxDb.Hsapiens.UCSC.hg38.knownGene, TxDb.Mmulatta.UCSC.rheMac3.refGene,
TxDb.Mmulatta.UCSC.rheMac8.refGene, TxDb.Mmusculus.UCSC.mm10.ensGene,
TxDb.Mmusculus.UCSC.mm10.knownGene,
TxDb.Mmusculus.UCSC.mm9.knownGene,
TxDb.Ptroglodytes.UCSC.panTro4.refGene, TxDb.Rnorvegicus.BioMart.igis,
TxDb.Rnorvegicus.UCSC.rn4.ensGene, TxDb.Rnorvegicus.UCSC.rn5.refGene,
TxDb.Rnorvegicus.UCSC.rn6.refGene, TxDb.Scerevisiae.UCSC.sacCer2.sgdGene,
TxDb.Scerevisiae.UCSC.sacCer3.sgdGene, TxDb.Sscrofa.UCSC.susScr3.refGene



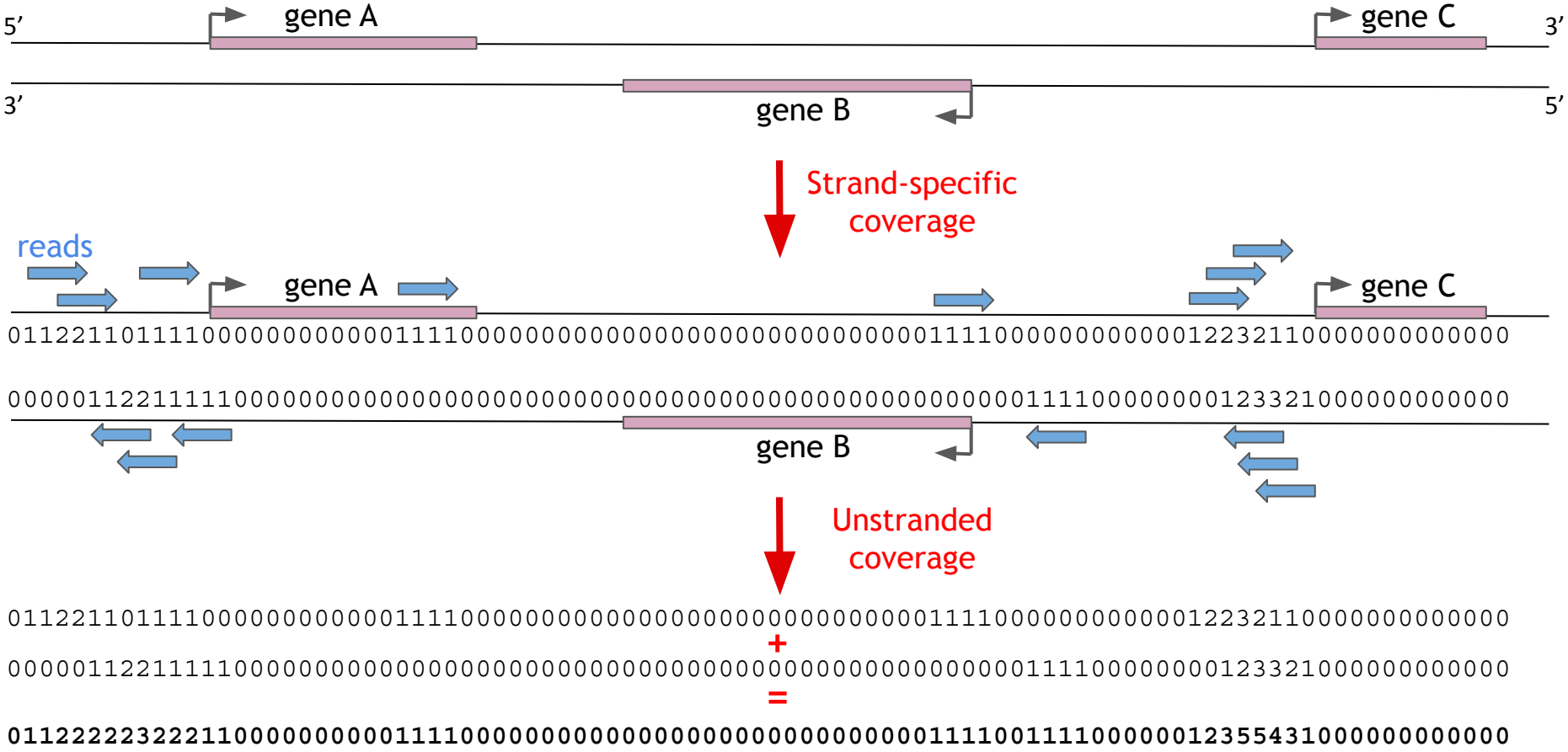
GenomicFeatures: Extract information



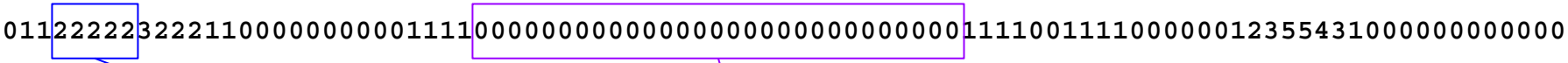
https://bioconductor.org/packages/3.7/bioc/vignettes/GenomicFeatures/inst/doc/GenomicFeatures.pdf

Coverages, heatmaps and average profiles

Coverage

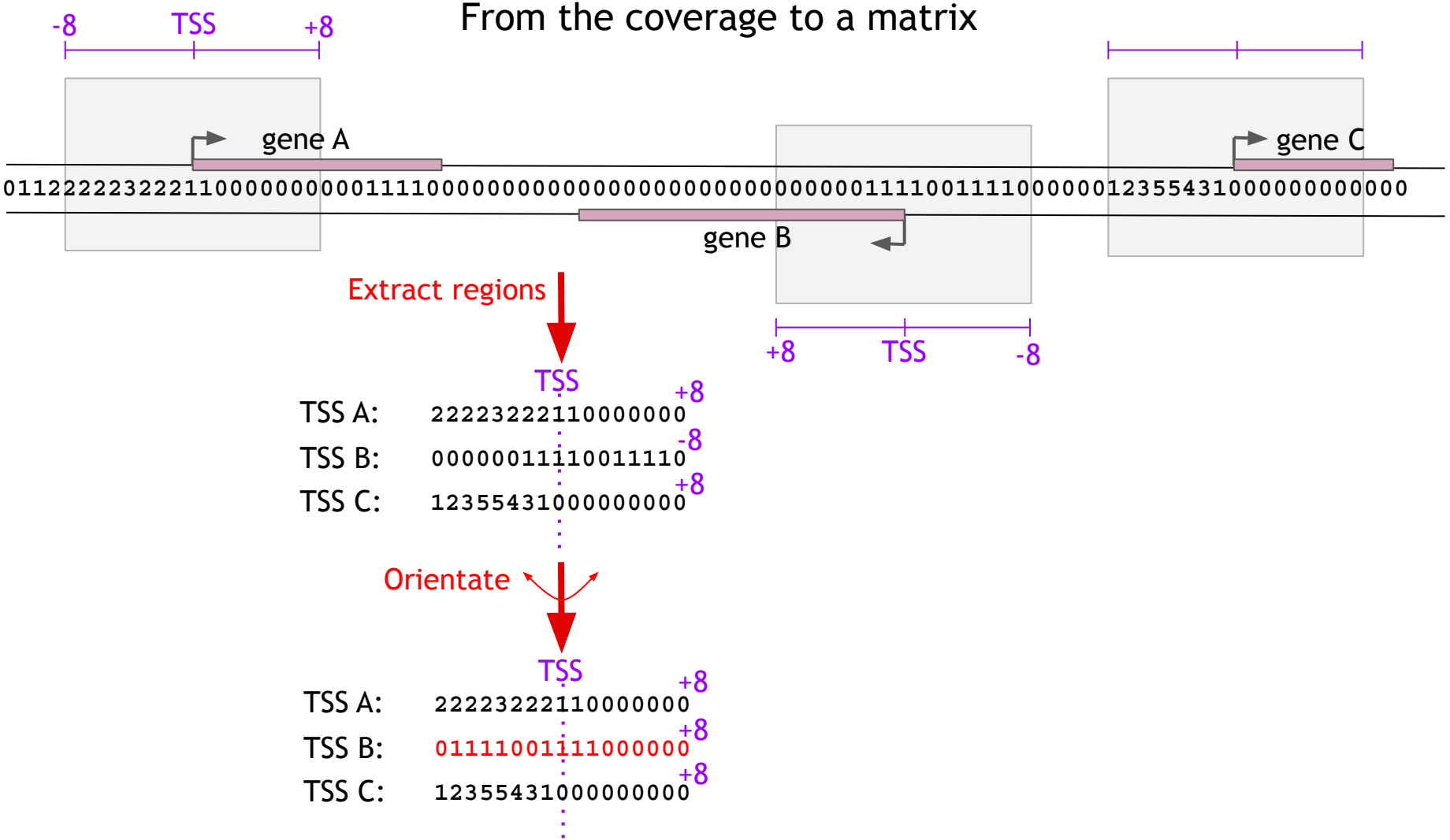

5’
gene A
gene C
3’
3’
gene B
5’
Strand-specific coverage
reads
gene A
gene C
0112211011110000000000001111000000000000000000000000000000001111000000000000122321100000000000000
0000011221111100000000000000000000000000000000000000000000000000000000000011110000000012332100000000000000
gene B
Unstranded coverage
0112211011110000000000001111000000000000000000000000000000001111000000000000122321100000000000000
+
0000011221111100000000000000000000000000000000000000000000000000000000000011110000000012332100000000000000
=
0112222232221100000000001111000000000000000000000000000000001111001111000000123554310000000000000

Run-length encoding (Rle)



Value	0	1	2	3	2	1	0	1	0	1	0	1	0	1	2	3	5	4	3	1	0
Length	1	2	5	1	3	2	10	4	29	4	2	4	6	1	1	1	2	1	1	1	12

From the coverage to a matrix



From the matrix to the heatmap

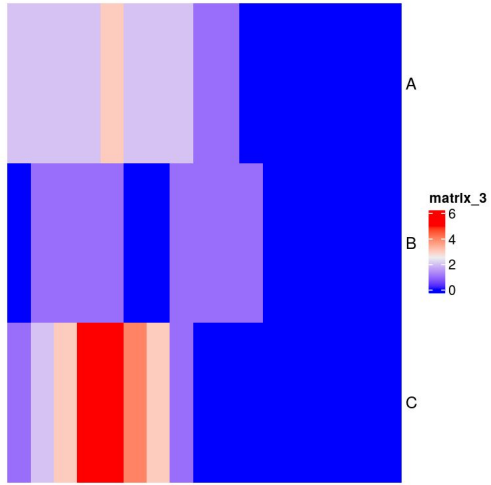
-8 TSS +8

TSS A: 22223222110000000

TSS B: 01111001111000000

TSS C: 12355431000000000

```
Heatmap(mymatrix,
        cluster_rows = FALSE,
        cluster_columns = FALSE)
```



Scale rows

```
t(scale(t(mymatrix)))
```

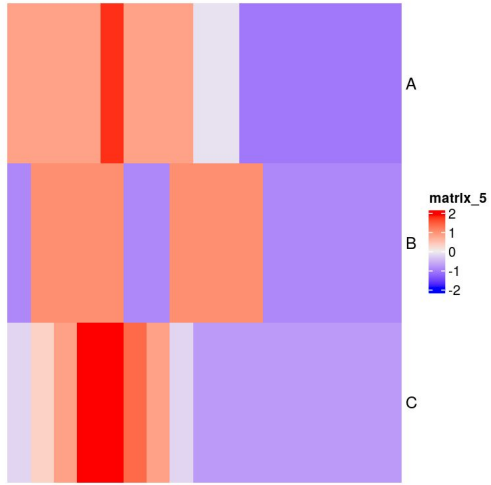
-8 TSS +8

TSS A:

TSS B:

TSS C:

```
Heatmap(myscaledmatrix,
        cluster_rows = FALSE,
        cluster_columns = FALSE)
```

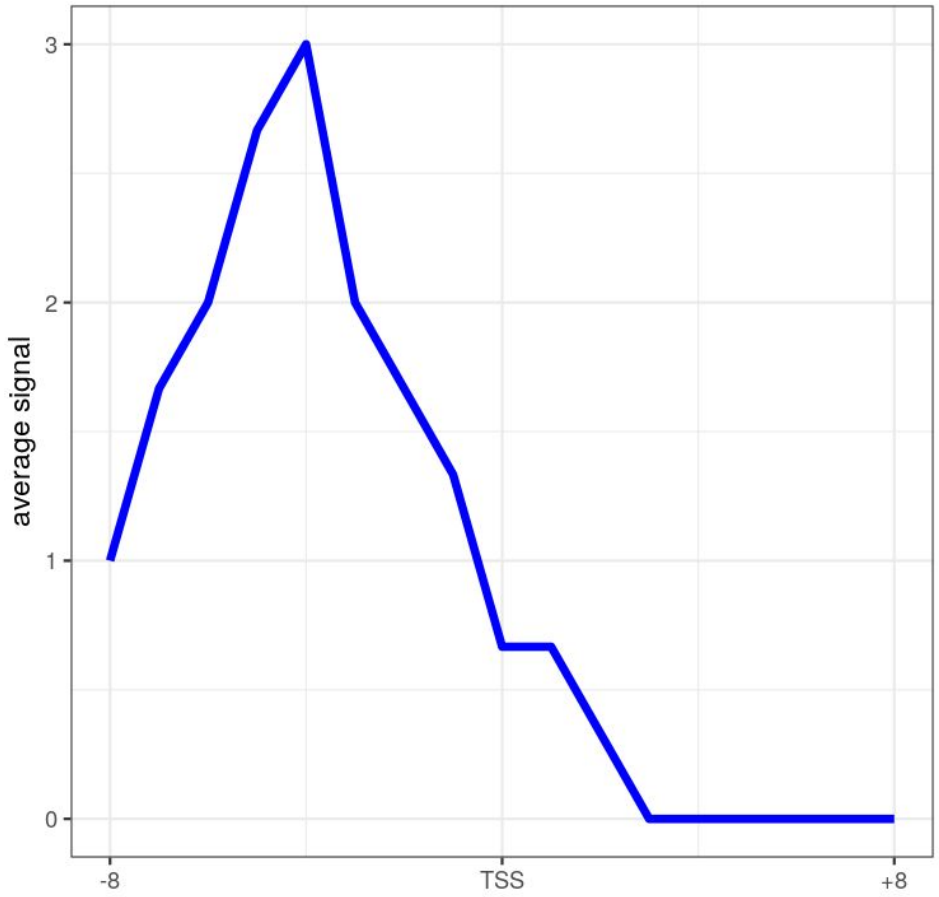


From the matrix to the average profile

```
          -8        TSS        +8
TSS A:    22223222110000000
TSS B:    01111001111000000
TSS C:    12355431000000000
```

`colMeans(mymatrix)`

```
position: -8        TSS        +8
    avg:  12233221.7.7.3000000
```



Useful to compare groups/clusters of genes

Workshop: Analyzing and visualizing CHiP-seq data

Enrichment of functional annotations

ClusterProfiler

Statistical analysis and visualization of functional profiles for genes and gene clusters

Guangchuang Yu
School of Public Health, The University of Hong Kong

2017-10-30

http://yulab-smu.top/clusterProfiler-book/

2 types of analyses

- Over-representation analysis (hypergeometric test = Fisher’s exact test)

- Gene Set Enrichment Analysis

2 types of analyses

- Over-representation analysis (hypergeometric test = Fisher’s exact test)

- Gene Set Enrichment Analysis

Hypergeometric test

How likely am I to observe the proportion that I observed of genes annotated with this specific annotation/pathway?

Think about poker:

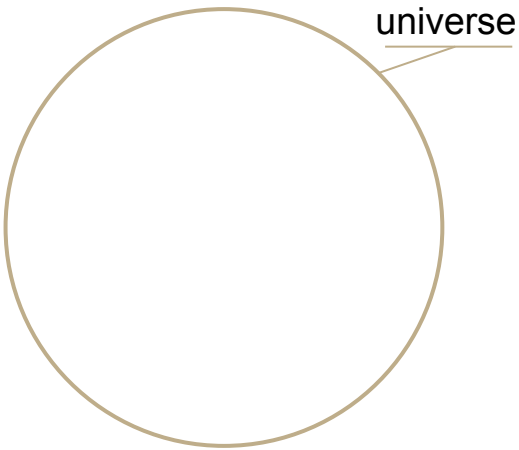
- You start with 52 cards (= your gene universe)
- There are 4 aces among the 52 cards (= genes involved in “apoptosis”)
- You draw 5 cards (= your “differentially expressed genes”)
- You get 3 aces (= your “apoptosis” genes among the DE genes)

What was the chance to get this by chance (the answer is low ;))?

=> hypergeometric distribution

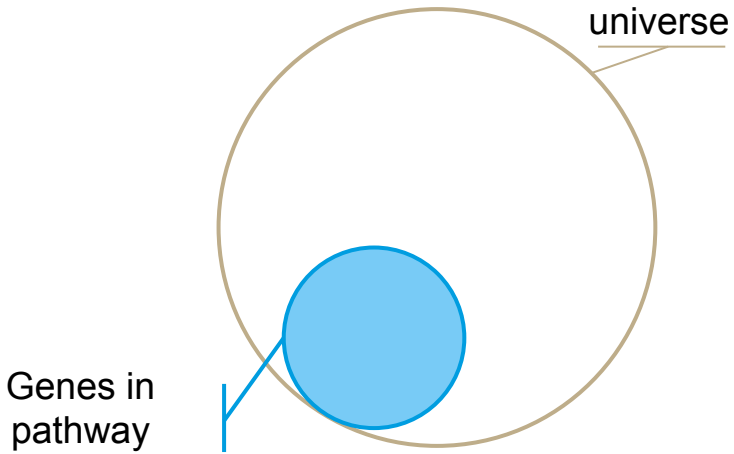
Hypergeometric test

How likely am I to observe the proportion that I observed of genes annotated with this specific annotation/pathway?



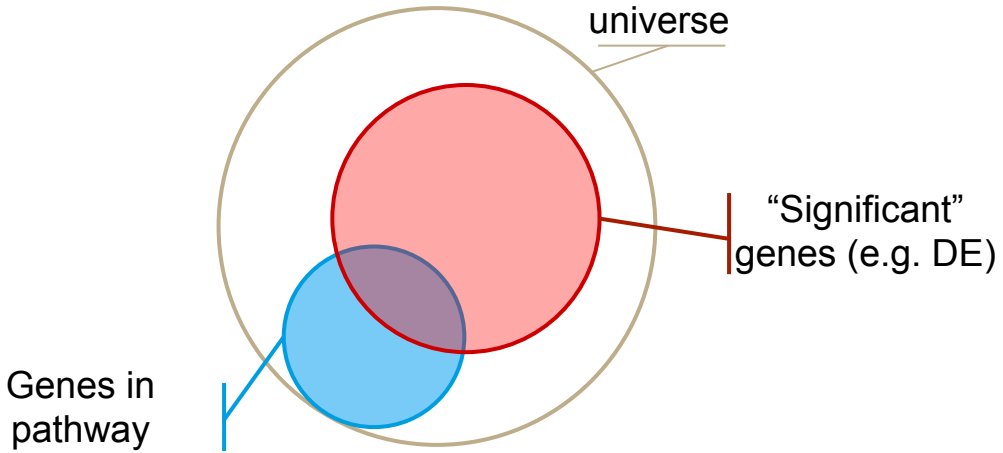
Hypergeometric test

How likely am I to observe the proportion that I observed of genes annotated with this specific annotation/pathway?



Hypergeometric test

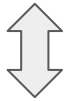
How likely am I to observe the proportion that I observed of genes annotated with this specific annotation/pathway?



Hypergeometric test

How likely am I to observe the proportion that I observed of genes annotated with this specific annotation/pathway?

	Genes in pathway	Genes Not in Pathway
Significant genes		
Non significant genes		



	Aces	Other cards (not aces)
In my hand		
Not in my hand		

Ressources

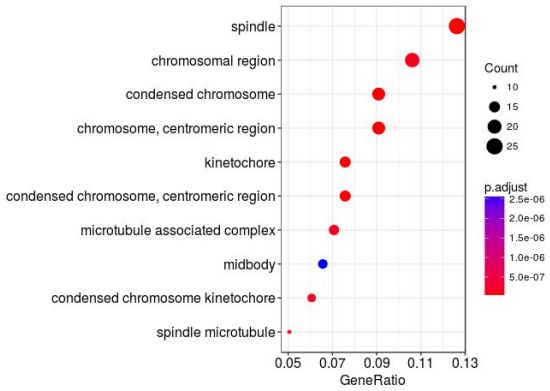
- Disease Ontology
- Network of Cancer Gene et DisGeNET (via **DOSE**)
- Gene Ontology (**GO** via AnnotationHub)
- **KEGG** (plus de 4000 espèces disponibles)
- Reactome Pathway (via **ReactomePA**)
- **DAVID** (via RDAVIDWebService)
- Molecular Signatures Database (**MSigDB**)
- autres (annotations, ontologies personnalisées etc)

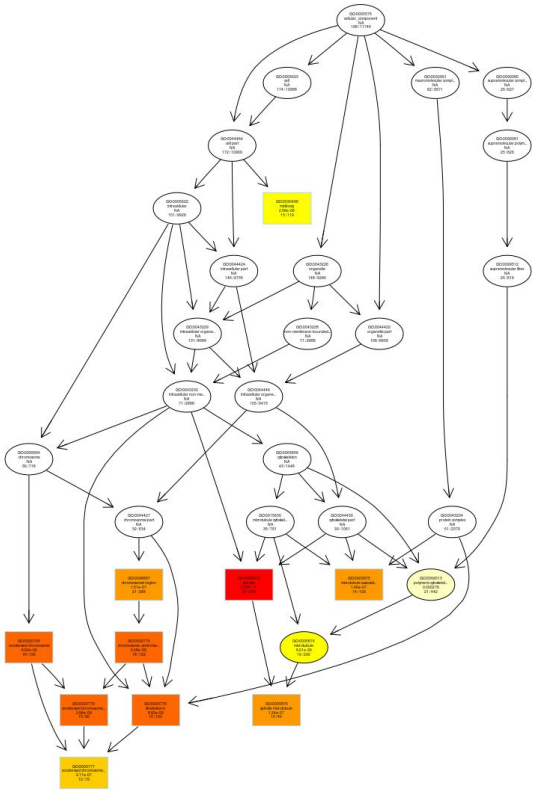
Fonctions d’enrichissement

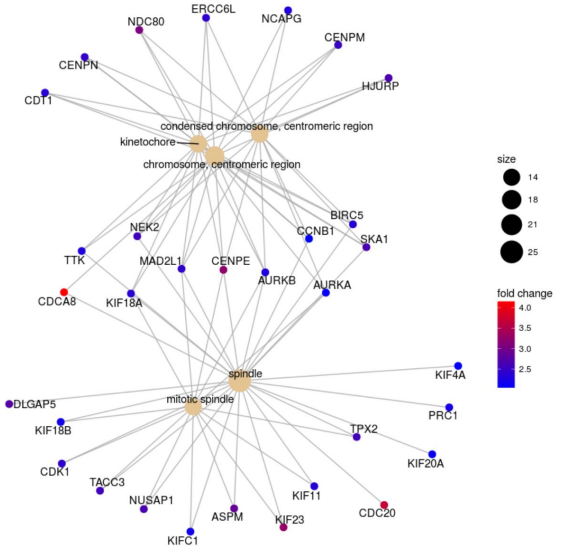
- enrichGO(), enrichKEGG(), enrichPathway(), enrichDAVID(), enrichDO()

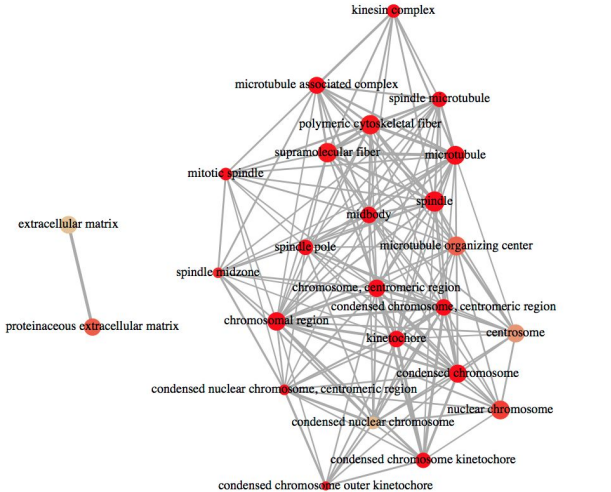
- enricher(x, TERM2GENE)
  - TERM2GENE est une data.frame avec une colonne contenant le nom du pathway et la seconde colonne contenant les gènes
  - Nous pouvons utiliser les groupes de gènes provenant de MSigDB en les téléchargeant depuis le site du Broad Institute et en utilisant la fonction read.gmt()

Visualisation des résultats

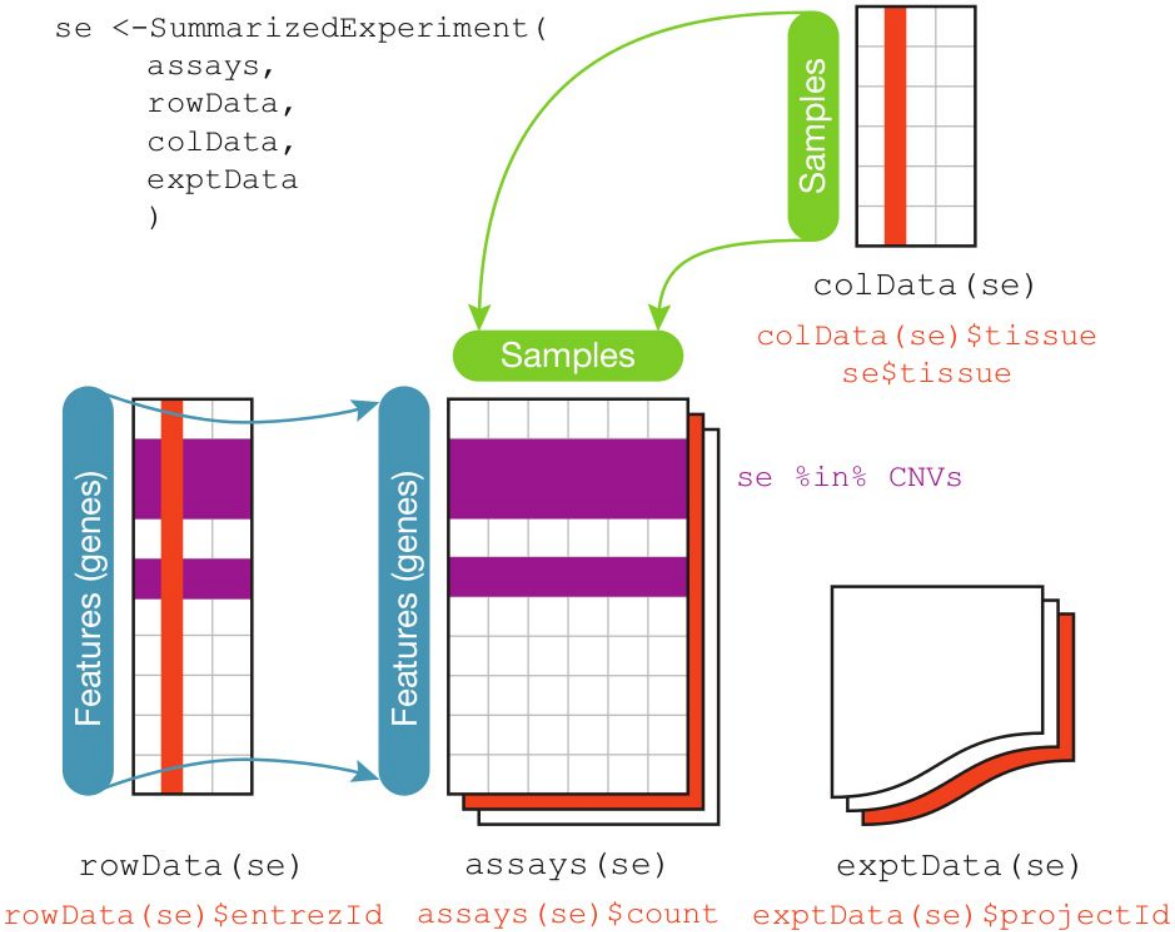








Storage of experimental data in R: *SummarizedExperiment*



Huber et al., Nat Methods, 2015

Other contributors:

- Stéphanie Le Gras
- Delphine Potier
- Morgane Thomas-Chollier
- Tao Ye
- Denis Putier
- Rachel Legendre
- Lucie Khamvongsa-Charbonnier