

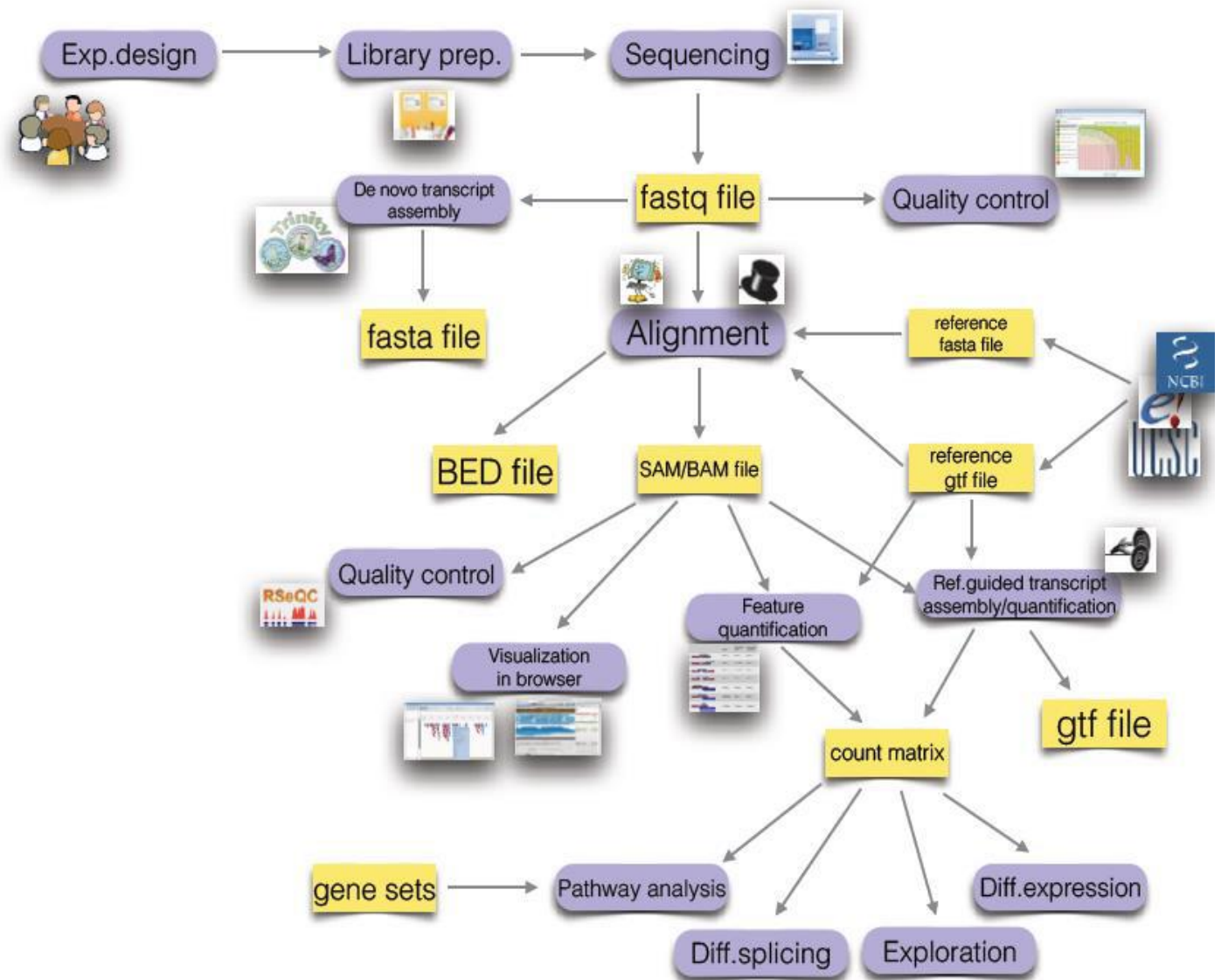
E5444 Analysis of sequencing data

Feature quantification

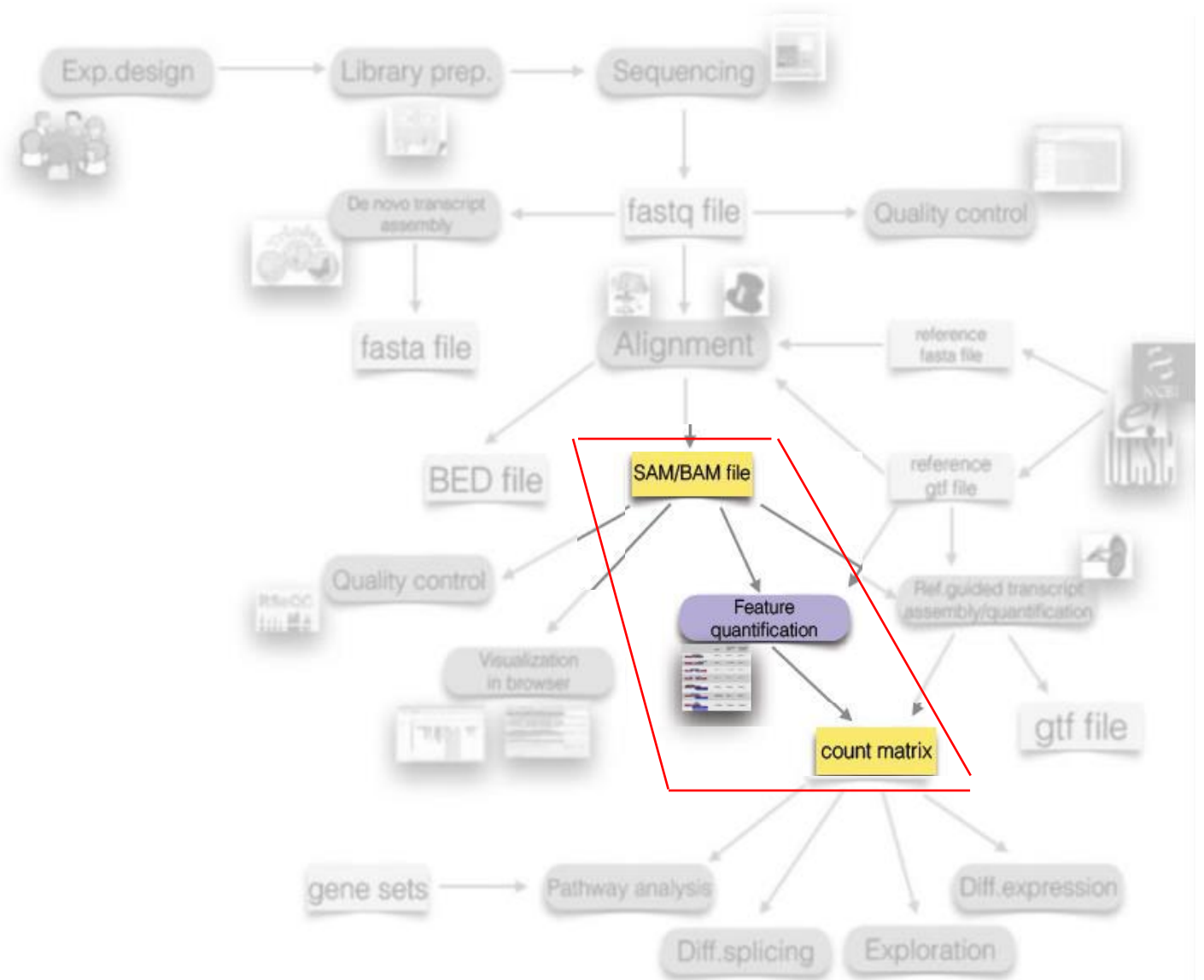
Autumn 2023

Eva Budinska

The NGS analysis pipeline



Step 4: Feature detection (quantification)



Once we finish the alignment, we can continue with the **quantification** of the **features**

Important – there are important differences within this and previous steps (alignment) in case of targeted gene sequencing experiments or metagenomics. These will be discussed more in detail later in separate (focused) lectures.

Step 4: Feature detection (quantification)

- Creates the **final table** with **read counts** for further statistical analyses

- A feature of interest differs based on the experiment:
 - gene, exon, intron... (WGS, WES)
 - transcript, isoform (RNA-seq)
 - variant - SNP, insertion, deletion, CNV - (WGS, WES, targeted sequencing)
 - promotor sequence (ChIP-Seq)
- In **transcriptomics** NGS experiments, the emphasis is on **quantification** of known transcripts (unless the aim is to get new isoforms) – we quantify the abundance of the RNA.
- In **genomic** NGS experiments, the emphasis is more on the **detection** of structural changes (the quantification is the % of alternative alleles found).

Step 4: Feature detection (quantification)

- Creates the final table with read counts for further statistical analyses

- The final output of this step is always a matrix with:
 - **Information** about the feature (ID, name, variant...) - annotation
 - **Quantification** of this feature in each of the samples

Feature annotation

- Gathering all the information about the feature

- Based on the feature type, we are using different information in the annotation files
 - RNAseq – ID and name of the gene/transcript, position on the chromosome...
 - Variants - specific format .vcf - includes reference allele, variant, annotation,
 - Metagenomics – taxonomical assignment of the OTUs (ASVs)
- We are using GTF/GFF files (remember from the previous alignment lecture)
 - List of genes, transcripts, exons, introns, CDS, ...
- We are using known databases for feature annotation
- Important! **Feature annotation can change using different versions of dbs!**

GTF/GFF files

- General feature format (.gff) or Gene transfer format (.gtf)

Extensions:

- Gtf/.gff2/.gff3
- GTF files can describe a variety of genomic features, such as genes, transcripts, exons, introns, and more. Each feature is represented as a separate line in the GTF file, with the relevant attributes specified.

- We need information about **what** and **where** it is located in a reference genome/transcriptome
- This information is usually stored in an annotation file
- Contains information about **each feature and its location**
- They are very similar but differ in the “strictness” and/or syntax
- Is often used for gene expression counting, localization of genes, etc.
- There is also .gff1 but it is very rare

GTF/GFF files

- General feature format (.gff) or Gene transfer format (.gtf)

Extensions:

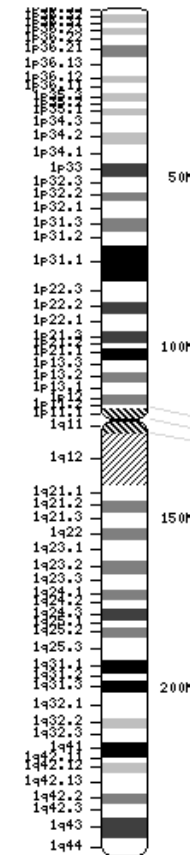
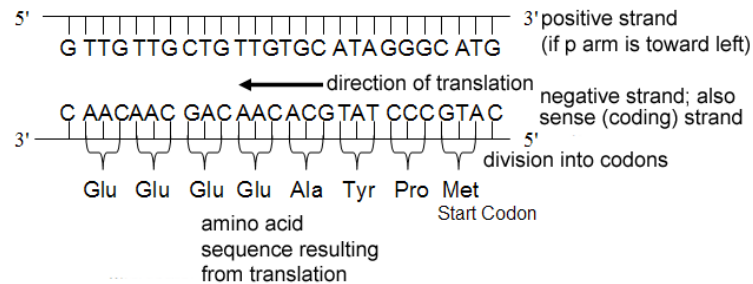
- gtf/.gff2/.gff3

- **Structure:** A GTF file is a tab-delimited text file with various columns that provide information about genomic features. The minimum required columns typically include the following:
 - Chromosome/contig name
 - Source (the program or database that generated the annotation)
 - Feature type (e.g., gene, transcript, exon, etc.)
 - Start position (the beginning of the feature)
 - End position (the end of the feature)
 - Strand (either "+" for the positive strand or "-" for the negative strand)
 - Attributes (a set of key-value pairs describing additional information about the feature, including gene and transcript identifiers)

Positive vs negative strand

- There are several conventions for labeling the two strands of a piece of DNA depending on the frame of reference.

- In the human genome, the National Center for Biotechnology Information (NCBI) website uses the term "positive strand" to refer specifically to **the strand whose 5' end begins at the end of the p arm**. Basepair numbering starts at the 5' end of this strand.



GTF/GFF files - example

- General feature format (.gff) or Gene transfer format (.gtf)

Extensions:

- gtf/.gff2/.gff3

- The files are usually TAB delimited and 1 numbering based
- .gff2–Sanger Institute http://www.sanger.ac.uk/resources/software/gff/spec.html#t_2
- .gtf – Modification of .gff2, sometimes called gff2.5 <http://mblab.wustl.edu/GTF22.html>
- .gff3 – Sequence Ontology Project <http://www.sequenceontology.org/gff3.shtml>

GTF/GFF files – comparison of different formats

- General feature format (.gff) or Gene transfer format (.gtf)

Extensions:

- gtf/.gff2/.gff3

.gff2

```

fosmid_2748I18 GEP CDS 6410 7639 . + 2 gene_id "CS-2"; transcript_id "CS-2-PC";
fosmid_2748I18 GEP CDS 7701 8899 . + 2 gene_id "CS-2"; transcript_id "CS-2-PC";
fosmid_2748I18 GEP stop_codon 8900 8902 . + . gene_id "CS-2"; transcript_id "CS-2-PC";
fosmid_2748I18 GEP exon 951 1192 . + . gene_id "CS-2"; transcript_id "CS-2-PC";
fosmid_2748I18 GEP exon 1262 1431 . + . gene_id "CS-2"; transcript_id "CS-2-PC";
fosmid_2748I18 GEP exon 2621 2747 . + . gene_id "CS-2"; transcript_id "CS-2-PC";

```

.gtf

```

140 Twinscan 3UTR 66823 66992 . - . gene_id "140.000"; transcript_id "140.000.1";
140 Twinscan stop_codon 66993 66995 . - 0 gene_id "140.000"; transcript_id "140.000.1";
140 Twinscan CDS 66996 66999 . - 1 gene_id "140.000"; transcript_id "140.000.1";
140 Twinscan intron_CNS 70103 70151 . - . gene_id "140.000"; transcript_id "140.000.1";
140 Twinscan CDS 70207 70294 . - 2 gene_id "140.000"; transcript_id "140.000.1";
140 Twinscan CDS 71696 71807 . - 0 gene_id "140.000"; transcript_id "140.000.1";
140 Twinscan start_codon 71805 71806 . - 0 gene_id "140.000"; transcript_id "140.000.1";

```

.gff3

```

##gff-version 3.2.1
##sequence-region ctg123 1 1497228
ctg123 . gene 1000 9000 . + . ID=gene00001;Name=EDEN
ctg123 . TF_binding_site 1000 1012 . + . Parent=gene00001
ctg123 . mRNA 1050 9000 . + . ID=mRNA00001;Parent=gene00001
ctg123 . mRNA 1050 9000 . + . ID=mRNA00002;Parent=gene00001
ctg123 . mRNA 1300 9000 . + . ID=mRNA00003;Parent=gene00001
ctg123 . exon 1300 1500 . + . Parent=mRNA00003
ctg123 . exon 1050 1500 . + . Parent=mRNA00001,mRNA00002
ctg123 . exon 3000 3902 . + . Parent=mRNA00001,mRNA00003
ctg123 . exon 5000 5500 . + . Parent=mRNA00001,mRNA00002,mRNA00003
ctg123 . exon 7000 9000 . + . Parent=mRNA00001,mRNA00002,mRNA00003
ctg123 . CDS 1201 1500 . + 0 ID=cds00001;Parent=mRNA00001
ctg123 . CDS 3000 3902 . + 0 ID=cds00001;Parent=mRNA00001

```

GTF/GFF files – comparison of different formats

- General feature format (.gff) or Gene transfer format (.gtf)

Extensions:

- gtf/.gff2/.gff3

Feature	GTF	GFF2	GFF3
Version	GTF is a specific format	GFF2 is an older format	GFF3 is the latest format
File Extension	.gtf	.gff2	.gff3
Column Count	Typically 9 columns:	9 columns:	9 columns:
	- Sequence name	- Sequence name	- Sequence name
	- Source	- Source	- Source
	- Feature	- Feature	- Feature
	- Start	- Start	- Start
	- End	- End	- End
	- Score	- Score	- Score
	- Strand	- Strand	- Strand
	- Frame	- Frame	- Frame
	- Attributes	- Attributes	- Attributes
Comment Lines	None	Supported but not required	Supported but not required
Parent-Child Relationships	Not explicitly defined	Not explicitly defined	Defined using Parent and ID attributes
Attribute Syntax	Key-Value Pairs	Key-Value Pairs	Key-Value Pairs (structured)
Multiple Features	No direct support	No direct support	Supports nested features
Hierarchical Structure	Typically used for genes	Not well-defined	Supports multi-level features
Ontology Annotations	Not supported	Not supported	Supports ontology-based annotations
Compatibility	Commonly used and well-supported	Less common, legacy format	Commonly used, current standard

- GFF3 is preferred to GFF2, its newer and comprises hierarchical structure

GTF/GFF files – the hierarchy

- Suppose you have a **gene**, which is a **parent feature**.
- This gene may have **multiple child features**, such as exons and introns.
- Exons can further contain subfeatures, like coding sequences (CDS) and untranslated regions (UTRs).
- In GFF3, this hierarchical relationship is represented using the "Parent" and "ID" attributes in the attributes column.

Feature	GTF	GFF2	GFF3
Version	GTF is a specific format	GFF2 is an older format	GFF3 is the latest format
File Extension	.gtf	.gff2	.gff3
Column Count	Typically 9 columns:	9 columns:	9 columns:
	- Sequence name	- Sequence name	- Sequence name
	- Source	- Source	- Source
	- Feature	- Feature	- Feature
	- Start	- Start	- Start
	- End	- End	- End
	- Score	- Score	- Score
	- Strand	- Strand	- Strand
	- Frame	- Frame	- Frame
	- Attributes	- Attributes	- Attributes
Comment Lines	None	Supported but not required	Supported but not required
Parent-Child Relationships	Not explicitly defined	Not explicitly defined	Defined using Parent and ID attributes
Attribute Syntax	Key-Value Pairs	Key-Value Pairs	Key-Value Pairs (structured)
Multiple Features	No direct support	No direct support	Supports nested features
Hierarchical Structure	Typically used for genes	Not well-defined	Supports multi-level features
Ontology Annotations	Not supported	Not supported	Supports ontology-based annotations
Compatibility	Commonly used and well-supported	Less common, legacy format	Commonly used, current standard

```
##gff-version 3.2.1
##sequence-region ctg123 1 1497228
ctg123 . gene 1000 9000 . + . ID=gene00001;Name=EDEN
ctg123 . TF_binding_site 1000 1012 . + . Parent=gene00001
ctg123 . mRNA 1050 9000 . + . ID=mRNA00001;Parent=gene00001
ctg123 . mRNA 1050 9000 . + . ID=mRNA00002;Parent=gene00001
ctg123 . mRNA 1300 9000 . + . ID=mRNA00003;Parent=gene00001
ctg123 . exon 1300 1500 . + . Parent=mRNA00003
ctg123 . exon 1050 1500 . + . Parent=mRNA00001,mRNA00002
ctg123 . exon 3000 3902 . + . Parent=mRNA00001,mRNA00003
ctg123 . exon 5000 5500 . + . Parent=mRNA00001,mRNA00002,mRNA00003
ctg123 . exon 7000 9000 . + . Parent=mRNA00001,mRNA00002,mRNA00003
ctg123 . CDS 1201 1500 . + 0 ID=cds00001;Parent=mRNA00001
ctg123 . CDS 3000 3902 . + 0 ID=cds00001;Parent=mRNA00001
```

This hierarchical structure is useful for **accurately representing the organization of genes and their constituent elements in a genome**, which is **crucial for various bioinformatics analyses**, including gene prediction, functional annotation, and visualization of genomic data.

Where to get GTF files

!IMPORTANT - GTF files downloaded from the UCSC Table Browser **have the same entries for gene id and transcript id.**

(not suitable for analyses of different isoforms)

To get the correct formatting, select in the output format field “all fields from selected table” and apply UCSC tool genePredToGtf on the resulting file.

The screenshot shows the UCSC Table Browser interface. At the top is a navigation bar with links: Genomes, Genome Browser, Tools, Mirrors, Downloads, My Data, Projects, Help, and About Us. Below this is the 'Table Browser' section. It contains a detailed instruction paragraph about using the program. Below the instructions are several dropdown menus and buttons for configuring the query: 'clade' (Mammal), 'genome' (Human), 'assembly' (Dec. 2013 (GRCh38/hg38)), 'group' (Genes and Gene Predictions), 'track' (GENCODE v32), 'table' (knownGene), 'region' (genome), 'identifiers' (paste list), 'filter' (create), 'intersection' (create), 'correlation' (create), 'output format' (GTF - gene transfer format (limited)), 'output file' (GTF_h38_knownGene_gencodev32.gtf), and 'file type returned' (plain text). At the bottom left are buttons for 'get output' and 'summary/statistics'. At the bottom right is a link to download from UCSC Genome Table Browser.

Use this program to retrieve the data associated with a track in text format, to calculate intersections between tracks, and to retrieve DNA sequence covered by a track. For help in using this application see [Using the Table Browser](#) for a description of the controls in this form, and the [User's Guide](#) for general information and sample queries. For more complex queries, you may want to use [Galaxy](#) or our [public MySQL server](#). To examine the biological function of your set through annotation enrichments, send the data to [GREAT](#). Refer to the [Credits](#) page for the list of contributors and usage restrictions associated with these data. All tables can be downloaded in their entirety from the [Sequence and Annotation Downloads](#) page.

clade: Mammal **genome:** Human **assembly:** Dec. 2013 (GRCh38/hg38)
group: Genes and Gene Predictions **track:** GENCODE v32
table: knownGene
region: genome position chrX:15,560,138-15,602,945
identifiers (names/accessions):
filter:
intersection:
correlation:
output format: GTF - gene transfer format (limited) Send output to Galaxy GREAT
output file: GTF_h38_knownGene_gencodev32.gtf (leave blank to keep output in browser)
file type returned: plain text gzip compressed

To reset **all** user cart settings (including custom tracks), [click here](#).

Downloading from UCSC Genome Table Browser
(<https://genome.ucsc.edu/cgi-bin/hgTables>)

BED files

- Browser Extensible Data (.bed)

- Could be considered as another type of annotation file but is much simpler
- Most often used for visualization in genomic browsers (UCSC Genome Browser: <https://genome.ucsc.edu/>, IGV <https://www.broadinstitute.org/igv/>, Tablet <https://ics.hutton.ac.uk/tablet/>, ...)
- Very often as a results of ChIP-Seq or similar experiments
- Used for targeted experiments to import target regions

IMPORTANT – in comparison with .gff/.gff2/.gff3 is zero-based

- Has only three mandatory columns – chromosome, start, end
 - Everything else is optional
- (gff/.gff2/.gff3 has 8/9 mandatory columns)

BED file example

- Browser Extensible Data (.bed)

<https://genome.ucsc.edu/FAQ/FAQformat.html#format1>

```
chr7 127471196 127472363 Pos1 0 + 127471196 127472363 255,0,0
chr7 127472363 127473530 Pos2 0 + 127472363 127473530 255,0,0
chr7 127473530 127474697 Pos3 0 + 127473530 127474697 255,0,0
chr7 127474697 127475864 Pos4 0 + 127474697 127475864 255,0,0
chr7 127475864 127477031 Neg1 0 - 127475864 127477031 0,0,255
chr7 127477031 127478198 Neg2 0 - 127477031 127478198 0,0,255
chr7 127478198 127479365 Neg3 0 - 127478198 127479365 0,0,255
chr7 127479365 127480532 Pos5 0 + 127479365 127480532 255,0,0
chr7 127480532 127481699 Neg4 0 - 127480532 127481699 0,0,255
```

VCF/BCF files

- Variant Call Format (.vcf)
- Binary Call Format (.bcf) is compressed version of .vcf

- Used in **targeted sequencing** while calling **SNPs/Indels/...**
- Contains information about “genetic variations”
- Officially: It contains meta-information lines, a header line, and then data lines each containing information about a position in the genome. The format also has the ability to contain genotype information on samples for each position.
- Defined format but again not every tool provide all “necessary” fields in the output
- Several versions, current at 4.3
- <http://www.1000genomes.org/node/101>
- <http://samtools.github.io/hts-specs/VCFv4.3.pdf>

Feature quantification

- Counting the numbers of reads that aligned to the feature

- The concept seems simple – lets have a look where the read mapped and then count all the reads within the feature
- For **gene-level** quantifications, 2 possibilities:
 1. Directly **count reads overlapping with the gene loci**
 2. In the case of transcriptomics, we can **quantify on the level of transcripts** and then **aggregate the values per gene**
- But, what about multimapped reads – this is a problem especially for isoforms in **transcriptome sequencing!**

Feature quantification – the most used SW

Usually count uniquely mapped reads and relies on counting scheme

Available in different languages (R, Python, C++, ...)

Important / The SW (usually) evolves over time and gain more functions and get faster

- `GenomicRanges`, `IRanges` – packages developed by core team of the Bioconductor project (R language) - include functions for counting reads that overlap genomic features
- `HTSeq-count` - function of the HT-Seq Python framework for processing RNA-seq or DNA-seq data
- `BEDTools` - a popular tool for finding overlaps between genomic features that can be used to count overlaps between reads and features, in C++, much faster, but not specifically designed for RNA-seq data, so can count reads for exons or interval features only, similar to `countOverlaps`.
- `featureCounts` – optimized read count program, fast and flexible, used to quantify reads generated from either RNA or DNA sequencing technologies in terms of any type of genomic feature
 - available either as a Unix command or as a function in the R package *Rsubread* (the core coded in C)

Feature quantification – selecting the SW

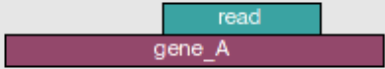

- When counting reads, make sure you know how the program handles the following:
 - overlap size (full read vs. partial overlap);
 - multi-mapping reads, i.e. reads with multiple hits in the genome;
 - reads overlapping multiple genomic features of the same kind;
 - reads overlapping introns
- The gene quantification will be strongly affected by the underlying gene models that are usually supplied to the quantification programs via GTF or BED(-like)

HTSeq gene-based counting schemes

Three different modes to tune its behavior with respect to:

- the multimapping
- the gaps

<http://www-huber.embl.de/users/anders/HTSeq/doc/count.html>

	union	intersection_strict	intersection_nonempty
	gene_A	gene_A	gene_A
	gene_A	no_feature	gene_A
	gene_A	no_feature	gene_A
	gene_A	gene_A	gene_A
	gene_A	gene_A	gene_A
	ambiguous	gene_A	gene_A
	ambiguous	ambiguous	ambiguous

featureCounts

Also allows to count reads overlapping with individual exons.

If an exon is part of more than one isoform in the annotation file, `featureCounts` will return the read counts for the same exon multiple times (n = number of transcripts with that exon).

If we want to assess differential expression of exons, it is highly recommended to create an annotation file where overlapping exons of different isoforms are split into artificially disjoint bins before applying `featureCounts` (e.g. using `dexseq_prepare_annotation.py` script of the DEXSeq package)

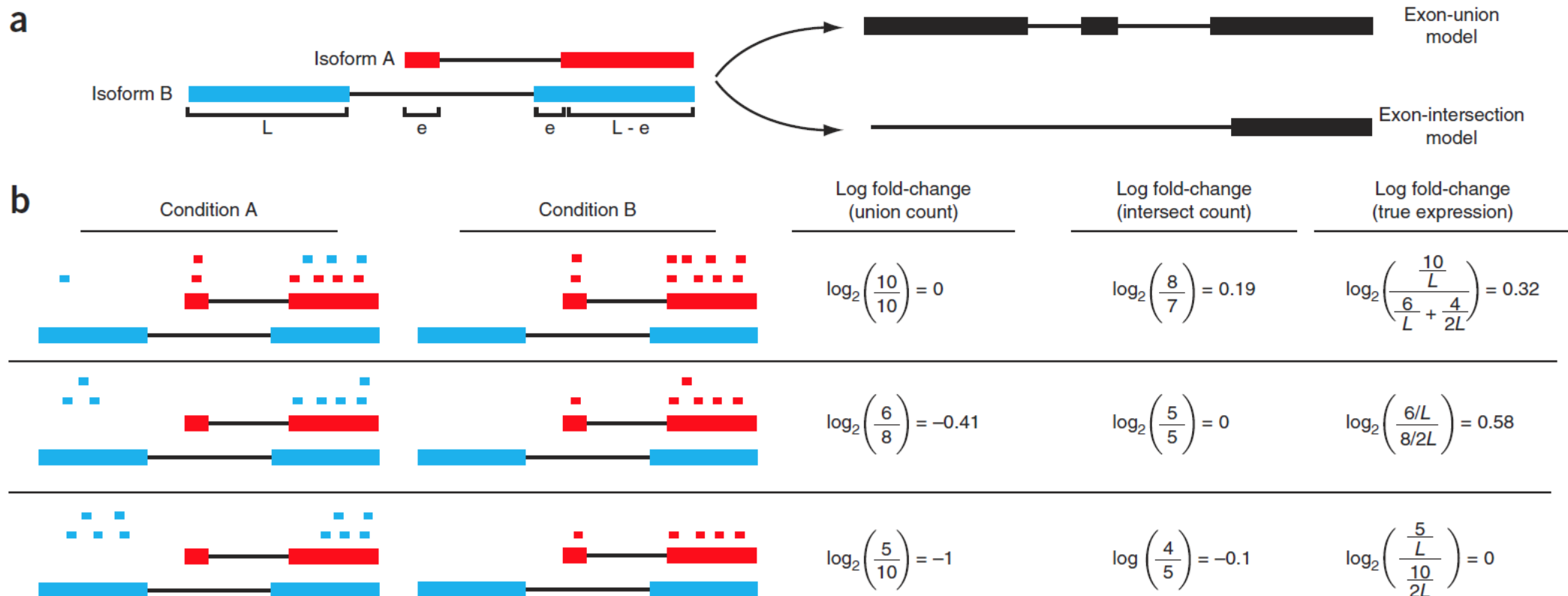
Isoform mapping

- Simple count-based approaches tend to ignore reads that overlap with more than one feature
- This is a problem if the aim is to quantify different isoforms (multiple isoform transcripts of the same gene may overlap)
- **Special SW:** Cufflinks, eXpress, DEXseq, RSEM
- RSEM is the one that tends to perform best in most comparisons and the statistical interpretations and assumptions to handle transcript structures have been widely adopted.

Quantifying gene vs transcript abundance

<https://www.semanticscholar.org/paper/Differential-analysis-of-gene-regulation-at-with-Trappell-Hendrickson/55507aba54a92b31a200e8112f088f5c356ed7bc>

Cuffdiff SW



The *alignment free* methods for transcript quantification!

Alignment is the most computationally expensive step in the whole pipeline.

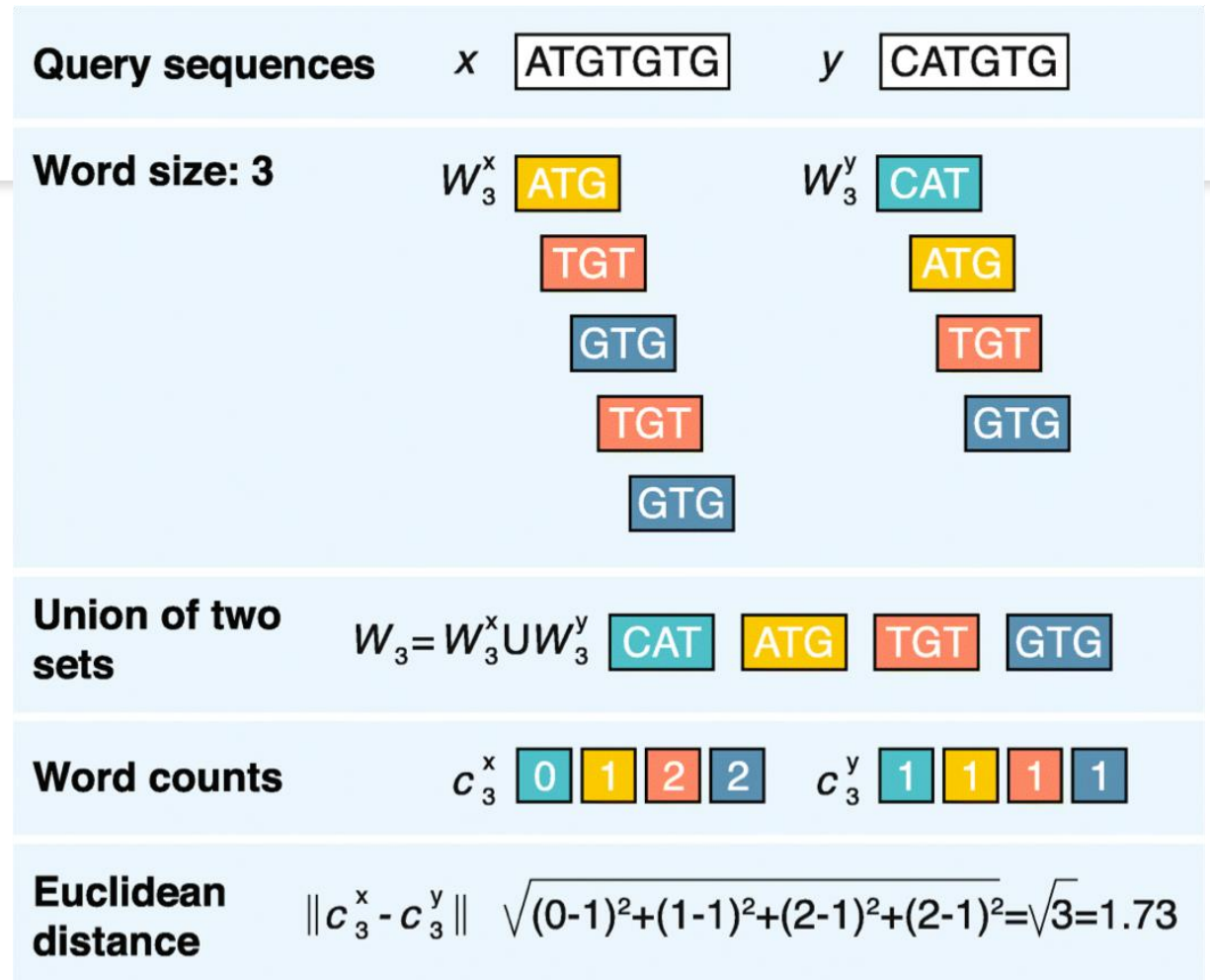
But what if we skipped this step??

- “Special” counting – RSEM; (Salmon, Kallisto)
- Mapping to transcriptome and dividing multi mapped reads between all isoforms
 - Expression by transcripts summarized to gene expression
 - <https://f1000research.com/articles/4-1521/v2>

Three main steps in the *alignment free* transcript quantification!

Zielezinski A, Vinga S, Almeida J, and Karlowski WM. Alignment-free sequence comparison: Benefits, applications, and tools. *Genome Biology*, 2017. doi:10.1186/s13059-017-1319-7.

1. The sequences for comparison (reads, reference) are sliced up into collections of unique (!) *k*-mers of a given length *k*.
2. For each pairwise comparison, we count the number of times a specific *k-mer* appears in both sequence strings that are being compared.
3. To assess the similarity between the two strings, some sort of **distance** function is employed (Euclidean distance; identical sequences have a distance of zero)



The *alignment-free*
transcript
quantification!

- In practice, Salmon and Kallisto will first generate an index of k-mers from **all known transcript sequences**.
- These transcript k-mers will then be compared with the k-mers of the sequenced reads, yielding a ***pseudoalignment*** that describes how many k-mers a read shares with a set of compatible transcripts (based on the distances)
- By grouping all pseudoalignments that belong to the same set of transcripts, they can then estimate the expression level of each transcript model.

The *alignment-free* transcript quantification!

The speed is increased but to a cost!

- The pseudoaligners **rely** absolutely on a **precise and comprehensive transcript annotation**.
- If a sequenced fragment originates from an intron or an unannotated transcript, it can be falsely mapped to a transcript since the relevant genomic sequence is not available.
- Alignment-based tools will discard reads if their edit distance becomes too large, pseudoalignment currently does not entail a comparable scoring system to validate the compatibility; therefore **there is no safeguard against spurious alignments**.
- For example, that a 100-bp-read can pseudoalign with a transcript with which it shares only a single k-mer { if no better match can be found within the universe of the pre-generated cDNA index.)

Comparison of different algorithms for counting features in RNAseq data

Feature	featureCounts	HTSeq-count	Cufflinks	eXpress	DEXSeq	RSEM
Developed by	Bioconductor team	Simon Anders	Cole Trapnell's Lab	Lior Pachter's Lab	Simon Anders' Lab	Bo Li's Lab
Compatibility	SAM, BAM, CRAM formats	Primarily works with SAM	SAM, BAM, GTF formats	SAM, BAM formats	BAM, GFF formats	SAM, BAM formats
Parallelization	Supports parallel processing	Does not support parallelization	Supports parallel processing	Supports parallel processing	Supports parallel processing	Supports parallel processing
Annotation Formats	GTF, GFF, SAF formats	Requires GFF feature type format	GTF format	GTF format	GFF format	GFF format
Output Format	Tab-delimited text file with counts	Tab-delimited text file with counts	Various files and tables	Various files	Tab-delimited text files and tables	Various files and tables
Speed	Known for speed and efficiency	Efficient but may be slower for large datasets	Slightly slower for large datasets	Known for speed and efficiency	Slightly slower for large datasets	Known for speed and efficiency
Handling Options	Various options for read counting, handling multi-mapped reads, and strandedness	Options for secondary alignments, ambiguous reads, and stranded data	Performs transcript assembly and quantification	Efficient and accurate quantification	Differential exon usage analysis	Accurate quantification, support for alternative isoforms
Statistical Methods	Employs a counting algorithm that considers multi-mapping reads	Employs a counting algorithm for read assignment	Utilizes a likelihood-based method for transcript quantification	Utilizes a Bayesian framework for quantification	Employs statistical models for identifying differentially expressed exons	Utilizes an Expectation-Maximization (EM) algorithm for quantification
Supports Isoform Counts	No	No	Yes	Yes	No	Yes
Additional Features	Flexible annotation format compatibility	Primarily designed for SAM files and GFF feature types	Transcript assembly and differential expression analysis	Speed and memory efficiency	Differential exon usage analysis	Support for estimating isoform expression
Documentation	Well-documented and actively maintained	Well-documented and widely used	Comprehensive documentation	Documentation available	Documentation available	Documentation available

Generated with help of ChatGPT