UCLA Institute for Quantitative and Computational Biology

Workshop W5a, March 2020

Introduction to RNAseq I
Day 3

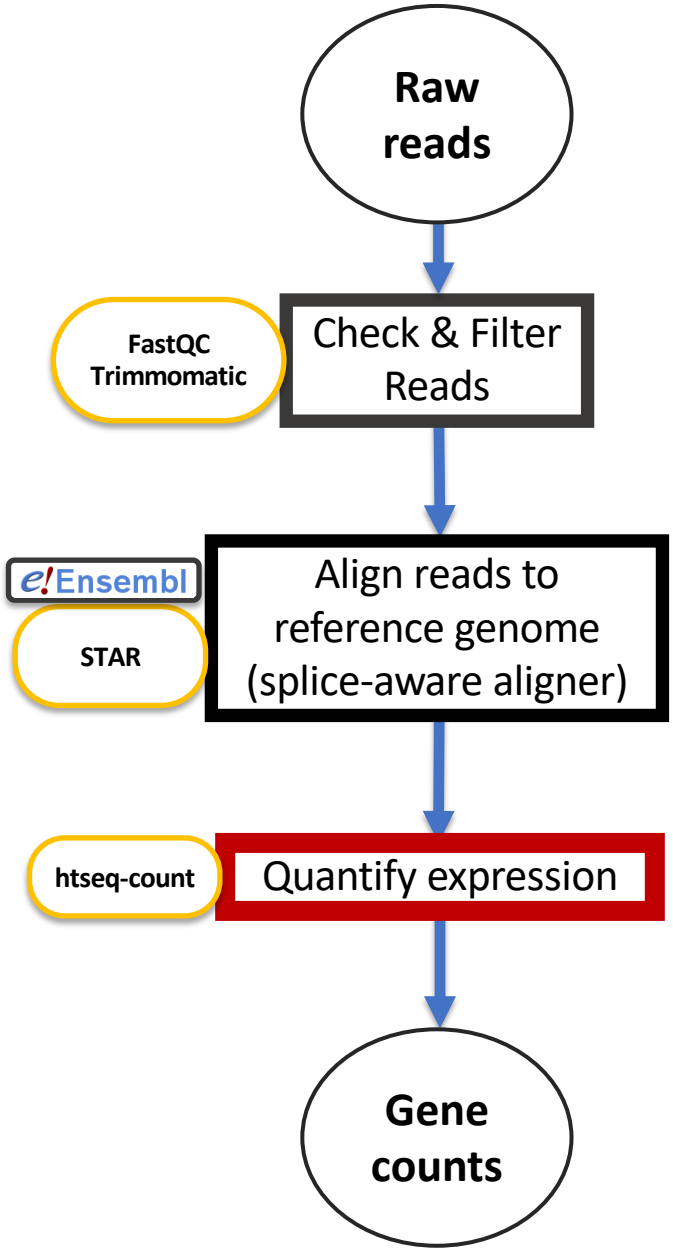
Nicolas Rochette

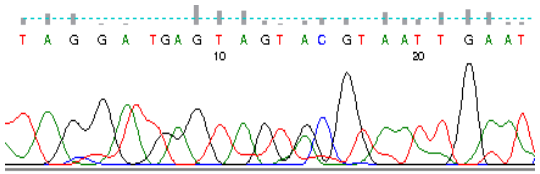
(EEB/ISG, UCLA)

Karolina Kaczor-Urbanowicz

(Oral Biology & Medicine, UCLA)

RNA-seq analysis: Overview

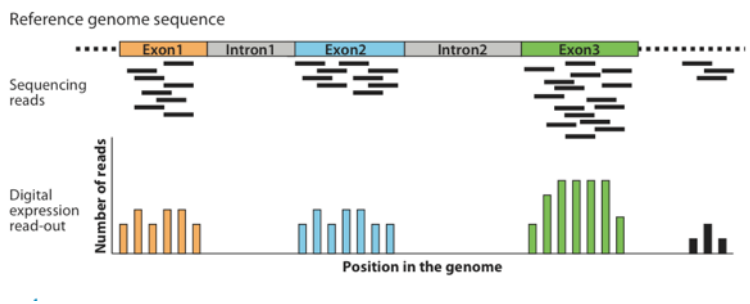




	sample1	sample2	sample3	...
gene1	999	701	616	
gene2	532	520	41	
gene3	14	36	305	
...				

Counting per-gene aligned reads

Counting per-gene alignments



	sample1	sample2	sample3	sample4	...
gene1	999	701	616	595	
gene2	532	520	41	26	
gene3	14	36	305	322	
...					

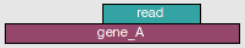
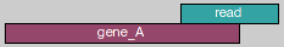
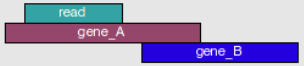
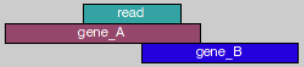
- **HTSeq** package
  - Anders, Pyl & Huber, 2015, *Bioinformatics 31:2*
  - Hompage at https://htseq.readthedocs.io/
  - Allows *per-gene* or *per-exon* counts (not per-transcript)
  - Designed for *differential gene expression testing* (vs. absolute gene expression quantification)
  - Includes the **`htseq-count`** command
    - Given GTF annotations, per-gene alignments are counted for every sample individually (ie. for each sample BAM file)

Choices for `htseq-count`

- What to count?
  ⇒ **`id_attr=gene_id`**
  for **per-gene** counts
- Where to count?
  ⇒ **`type=exon`**
  to count alignments overlapping exons
- How to count?
  ⇒ **`mode=union`**
  for differential expression at the gene level

Each read may be flagged as:

- On one gene
- Ambiguous (several genes)
- On unknown features

	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

Installing HTSeq

- Easiest if every user installs their own HTSeq (via PIP, the Package Installer for Python)

```
module load python/3.7.2
python3 -m pip install --user HTSeq
```

- This will install HTseq at `~/.local/bin/htseq-count`

Running htseq-count (interactively)

- eg. for the alignments (BAM file) of the **`P10_rep1`** sample:

```
bam_file=./P10_rep1.Aligned.sortedByCoord.out.bam
gtf_file=/PATH/TO/Mus_musculus.GRCm38.98.gtf

~/.local/bin/htseq-count \
    --stranded=yes \
    --idattr=gene_id \
    --type=exon --mode=union \
    --format=bam \
    $bam_file \
    $gtf_file \
    > $bam_file.pergene_counts
```

Merging all samples' counts (into one big table)

- Many ways to do it (awk, R, python…) — whichever works for you is good, just be sure not to mix up sample labels!
- One way with the R package EdgeR:

```
# install.packages("BiocManager")
# BiocManager::install("edgeR")
library(edgeR)

samples <- c('P10KO_rep1', 'P10_rep1')
files <- paste0(samples,'.Aligned.sortedByCoord.out.bam.pergene_counts')
dge_counts <- readDGE(files, labels=samples, header=FALSE)

write.csv(dge_counts$counts, 'counts.csv')
```

Additional Methods & considerations

Important (if obvious) properties of the Counts Table

	sample1	sample2	sample3	sample4	...
gene1	999	701	616	595	
gene2	532	520	41	26	
gene3	14	36	305	322	
...					

- Not all samples have the same total number of counts (∽aligned reads) *Scaling/normalization is necessary!*
  → 'Counts per gene per million reads' (**CPM**)
- Some genes are longer than others!
  → 'Transcripts per million' (**TPM**), **RPKM**/FPKM
  (But note that for differential expression analysis it doesn't matter — actually **raw counts *must* be used**!)

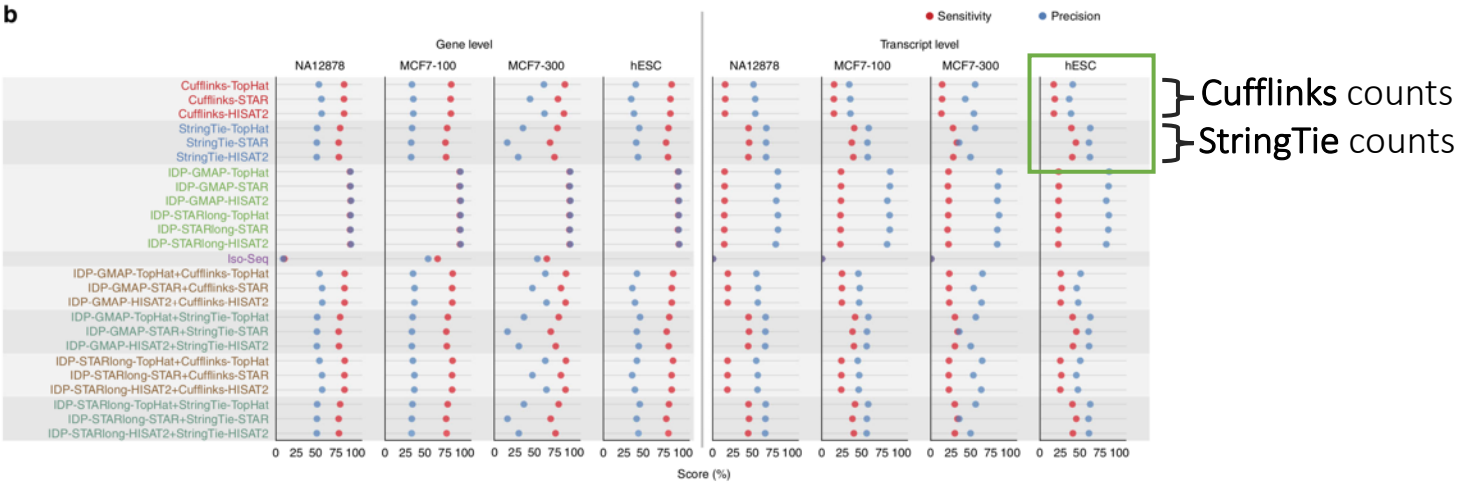
Aligners

They all produce SAM/BAM files!

- **STAR** — Used & recommended by many pipelines (eg. GATK best practices). Has large memory requirements.
- **Tophat 1 & 2** — legacy mainstream aligners that have been superseded by HISAT2 from the same research group. Slow!
- **HISAT2** — Successor to Tophat2. Similar to STAR in overall performance, for most purposes.

Aligners (ctd.)

- Most computationally expensive step of the pipeline
- But most of the variation in alignment results comes from the **quality of input read libraries** & genome annotations.
- The largest differences in outcome more often come from method choices at **other steps** of the RNAseq pipeline



Sahraeian et al. 2017, Nat Comm 8:1

- The mapping method should be adjusted to the ***type of data*** ("standard" RNAseq vs. eg. short RNAs, long reads—STAR/Minimap2) and ***goals*** (transcript discovery…)

Gene counting approaches

- **HTSeq** (Anders et al.2015, Bioinformatics 31:2)
- **Cufflinks** (Trapnell et al, 2010, Nat Biotech 28:5)
- **StringTie** (Pertea et al. 2015, Nat Biotech 33:3)
- LeafCutter (Li+Knowles2018), intron-centric
- Alignment-free methods
  - **Kallisto** (Bray et al. 2016, Nat Biotech 34:5)
  - **Sailfish** (Patro et al. 2015, Nat Biotech 32:5)

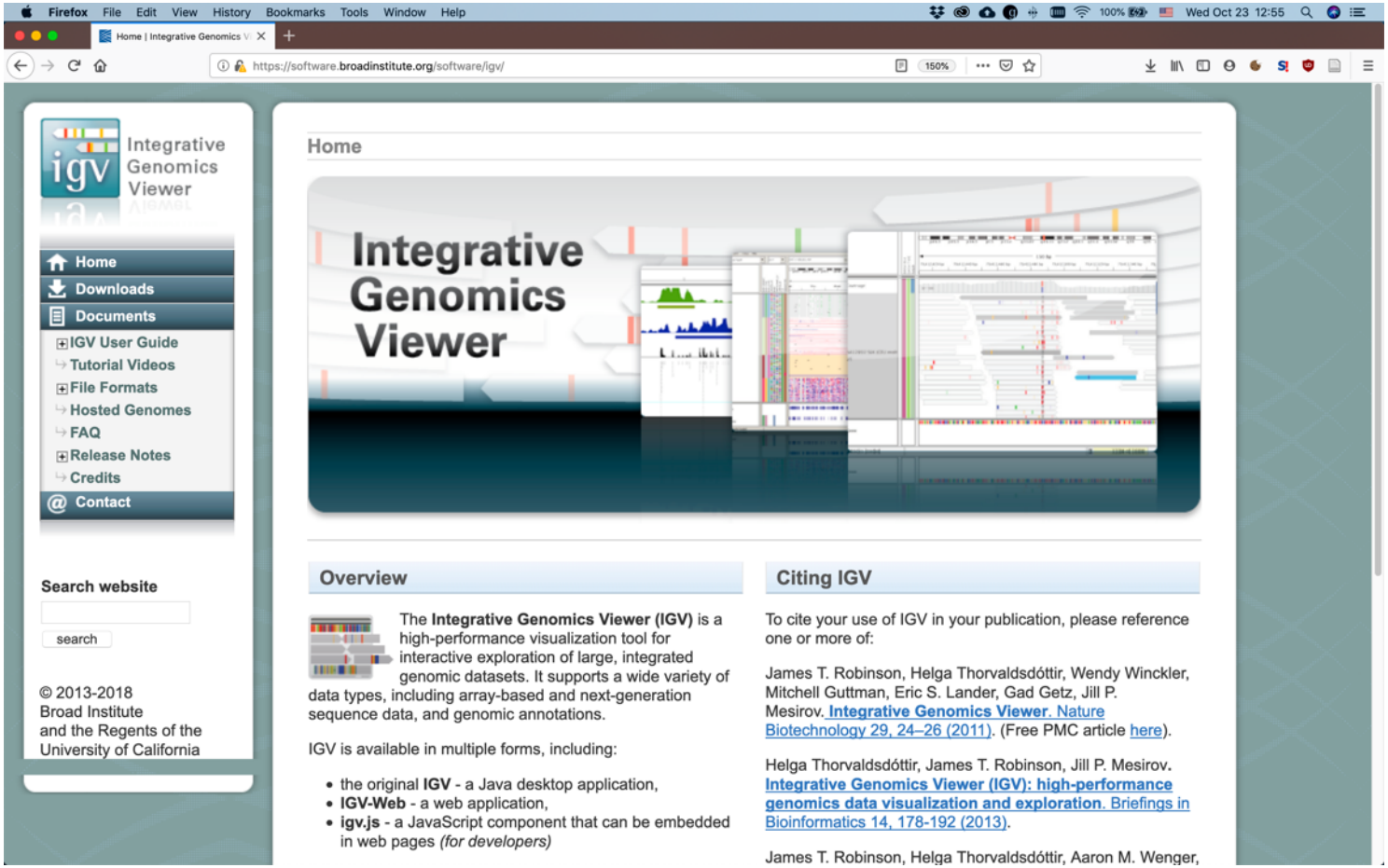
⇒ Which one to use depends on the **purpose** the counts will be used for

- differential gene expression vs. absolute estimates
- analyses at the gene or transcript level; types of RNA that are of interest

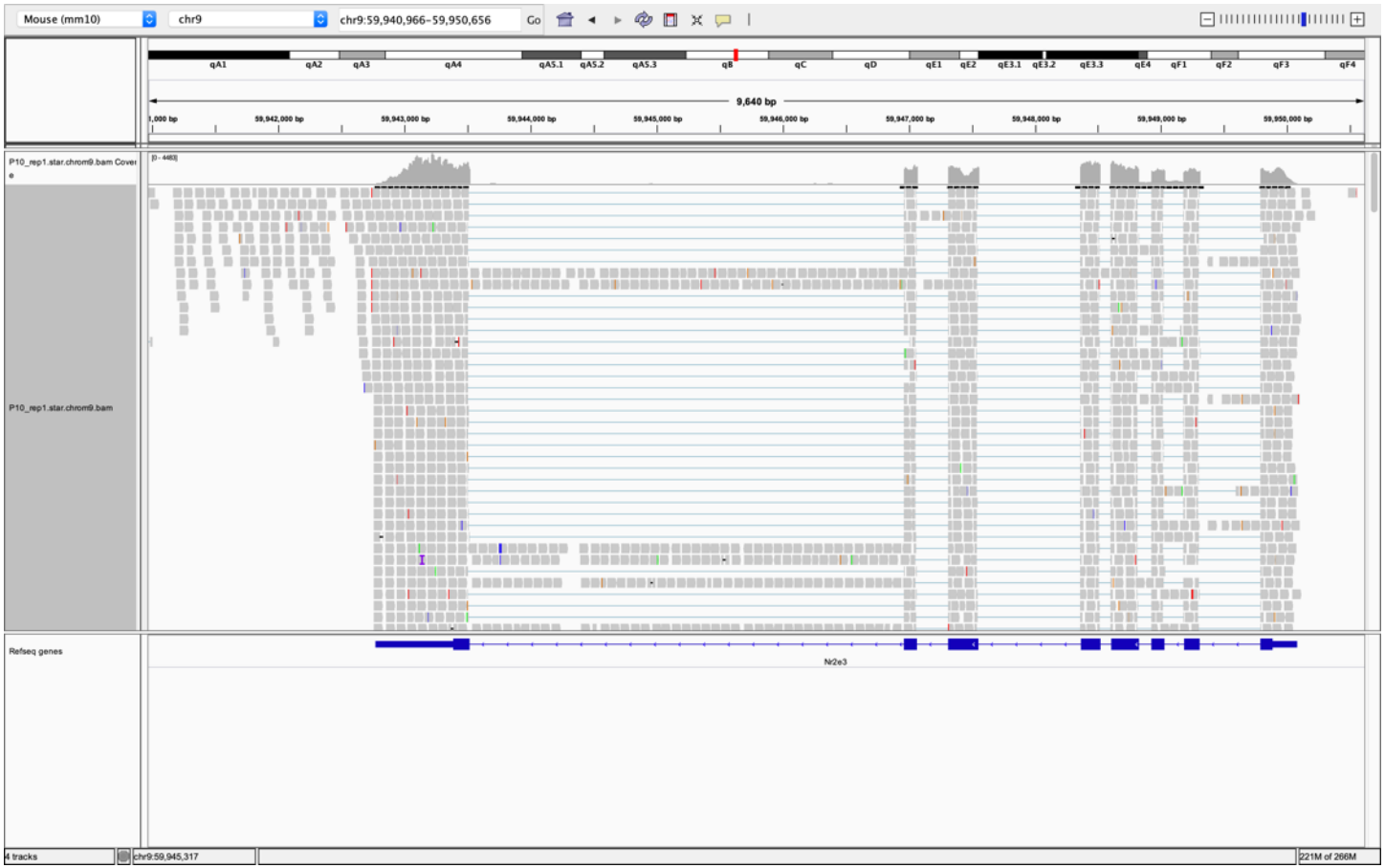
Visualizing alignments

IGV

https://software.broadinstitute.org/software/igv/



- Lets you run a genome browser on your laptop
- Similar to IGB (‘Integrated Genome Browser’)



(Read alignments to the NR2E3 region, for sample P10_rep1)

- **Visualization/manual inspection** of the "alignment data" that is the basis for gene expression counts

Installing & running IGV

- **Download** the version of IGV corresponding to your operating system from the website's downloads page
- Open IGV and select the mouse genome
- **jump to the NR2E3 gene** region (chr9:59,942,771-59,950,079)
- *On the cluster*, **create an index** for the `P10_rep1.Aligned.sortedByCoord.out.bam` alignments file (using `samtools index`)
- Download the BAM file and its index **to your laptop**
- **Load the BAM file** in IGV (File→Load from File)
- Observe the read alignments; notice the splice events, the mismatches to the references. Roughly how many are there?
- Repeat the procedure for `P10KO_rep1`; notice how very few reads align to NR2E3 for that sample.

Running batch jobs

Single-sample batch job

- See the **`batchjob_single.bash`** script in the workshop's directory for an example
- Submit it using the qsub grid command:
  `qsub -N some_jobname batchjob_single.bash`
  - command lines options to qsub may be specified in the script itself (**`#$`** lines) depending on the needs of the commands
- Monitoring your jobs: `qstat -u USER`

Array jobs

- Typical context:. “do something for each sample”
- Use the **`-t`** option of **`qsub`** (eg. `-t 1-30` for 30 samples)
- See the **`batchjob_array.bash`** script in the workshop’s SCRATCH directory for an example
- Submit it using the **`qsub`** grid command:
  `qsub -N my_jobname -t 1-30 batchjob.bash`