Introduction to RNA-seq for Differential Expression Analysis – 2

by Jiajia Li

Biological Data Science Institute

6 May 2025

Australian National University

Learning Objectives of Today

- Align RNA-seq data to reference using HISAT2
- Learn about SAM/BAM file format
- Convert SAM to Bam
- Sort alignment
- Expression estimates with StringTie
- Learn about FPKM and TPM
- Generate raw counts with htseq-count

Run HISAT2 Alignment

Let's review what's HISAT2 first.

HISAT2 is a widely used software tool for aligning sequencing reads (especially RNA-seq) to a reference **genome**.

Splice-aware alignment:

- HISAT2 is designed to align RNA-seq reads, which often span exon-exon junctions.

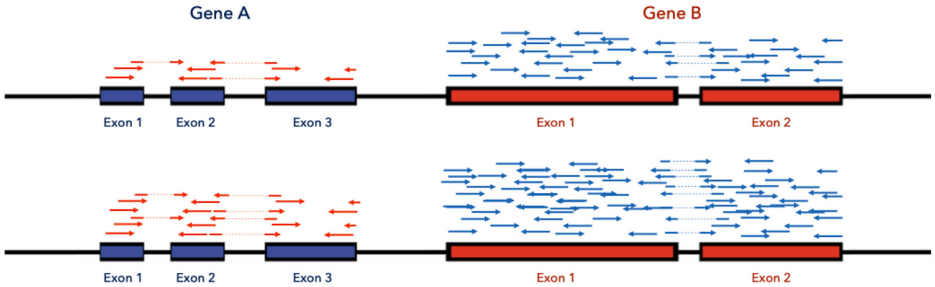


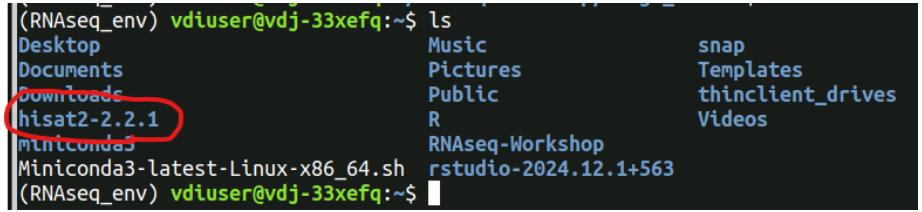
Figure 1: Illustration of paired-end sequencing reads from an RNA-Seq experiment aligning to the reference genome. The dotted line represents individual reads that have been split to enable mapping on the discrete exons that it originated from.

Add HISAT2 to PATH

Previously we have downloaded HISAT2 and extracted to our HOME folder.

To ensure easier access, we can add the HISAT2 to PATH so we can call it in command at any location of the machine.

Double check if you have put HISAT2 in your HOME folder.
`cd ~` then `ls`



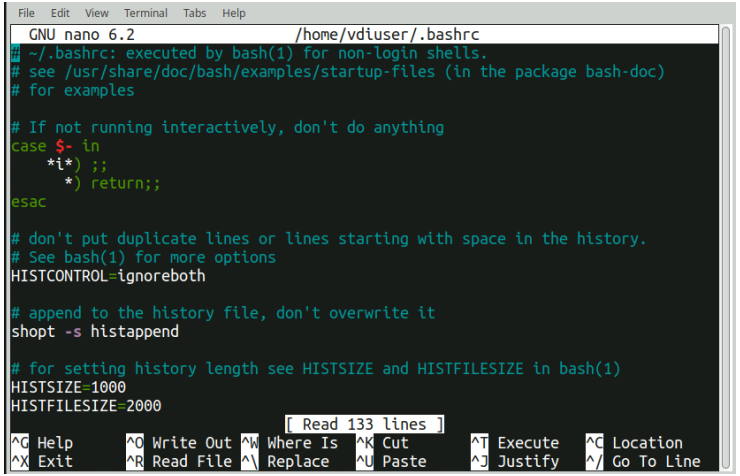
The folder is here.

Add HISAT2 to PATH

Then we need to edit a file called ".bashrc" under our home directory. To edit a file in Linux, we can use the **nano editor**.

`nano ~/.bashrc`

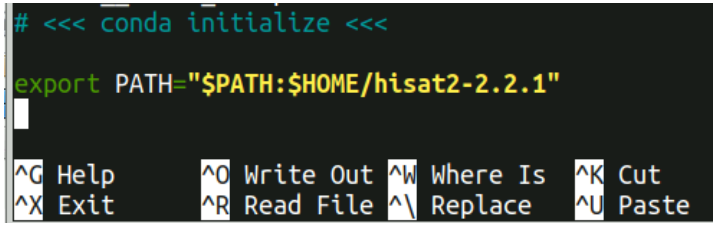
Then you will be guided to the editor interface. Looking like this:



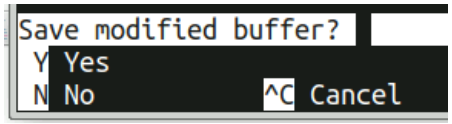
Add HISAT2 to PATH

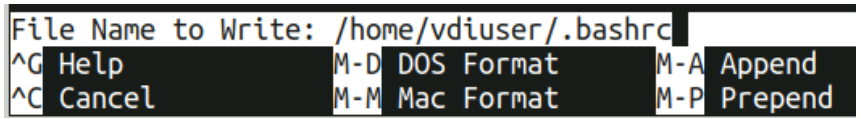
Inside the editor, you can use arrow keys to navigate and type text.

Use the “down” arrow key to navigate to the end of the file. Then add this line:
`export PATH="$PATH:$HOME/hisat2-2.2.1"`



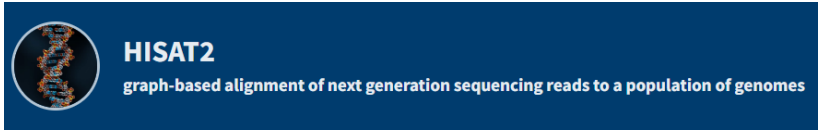
To save and exit the file, press “Ctrl + X”. You will be asked:





Press “Y” for first question, then press “ENTER” for second question.

Add HISAT2 to PATH



After saving the changes, run:

`source ~/.bashrc`

```
(RNAseq_env) vdiuser@vdj-33xefq:~$ source ~/.bashrc
(base) vdiuser@vdj-33xefq:~$
```

Let's test if it's working:

`hisat --version`

```
(base) vdiuser@vdj-33xefq:~$ hisat2 --version
/home/vdiuser/hisat2-2.2.1/hisat2-align-s version 2.2.1
64-bit
Built on Nucleus005
Wed Dec  2 16:48:17 CST 2020
Compiler: gcc version 5.4.0 (GCC)
Options: -O3 -m64 -msse2 -funroll-loops -g3 -DPOPCNT_CAPABILITY -std=c++11
Sizeof {int, long, long long, void*, size_t, off_t}: {4, 8, 8, 8, 8, 8}
(base) vdiuser@vdj-33xefq:~$
```

It's working!!!

Run HISAT2 Alignment

Let’s create a folder to store the alignment result first.

`cd ~/RNAseq-Workshop`
`mkdir align_result`
`cd align_result`

```
(RNAseq_env) vdiuser@vdj-33xefq:~/RNAseq-Workshop$ mkdir align_result
(RNAseq_env) vdiuser@vdj-33xefq:~/RNAseq-Workshop$ cd align_result
(RNAseq_env) vdiuser@vdj-33xefq:~/RNAseq-Workshop/align_result$
```

Run HISAT2 Alignment

The basic usage of HISAT2 is:
`hisat2 -x <index_base> -1 <reads_1.fq> -2 <reads_2.fq> -S <output.sam>`

- `-x`: the location of your reference genome and index files, but you only need to put the index base which is the part before ".1.ht2" and ".2.ht2".
- `-1`: input FASTQ file, read1
- `-2`: input FASTQ file, read2
- `-S`: output SAM file, specify file name and location

Let's try this on our first sample, **UHR_Rep1.**

First let's go to `~/RNAseq-Workshop`, it's easier for us to type the directory and file locations if we stay in this folder.
`cd ~/RNAseq-Workshop`

Run HISAT2 Alignment


```
hisat2 -x reference/chr22_with_ERCC92_index \
       -1 data/UHR_Rep1_ERCC-Mix1_Build37-ErccTranscripts-chr22.read1.fastq.gz \
       -2 data/UHR_Rep1_ERCC-Mix1_Build37-ErccTranscripts-chr22.read2.fastq.gz \
       -S align_result/UHR_Rep1.sam
```

Run this, and we will get an output SAM file in our folder “align_result”. It also generates an alignment summary and printed on screen.

```
227392 reads; of these:
  227392 (100.00%) were paired; of these:
    957 (0.42%) aligned concordantly 0 times
    223659 (98.36%) aligned concordantly exactly 1 time
    2776 (1.22%) aligned concordantly >1 times
    ----
    957 pairs aligned concordantly 0 times; of these:
      439 (45.87%) aligned discordantly 1 time
    ----
    518 pairs aligned 0 times concordantly or discordantly; of these:
      1036 mates make up the pairs; of these:
        257 (24.81%) aligned 0 times
        600 (57.92%) aligned exactly 1 time
        179 (17.28%) aligned >1 times
99.94% overall alignment rate
(base) vdiuser@vdj-33xefq:~/RNAseq-Workshop$
```

99.94% overall alignment rate, very high!! Good quality data.

HISAT2 Output SAM/BAM

HISAT2 produces alignments in **SAM/BAM** format, compatible with tools like SAMtools, StringTie, and featureCounts for downstream analysis.

SAM and BAM are file formats used to store **sequence alignment data** - that is, how sequencing reads align to a reference genome.

A SAM file has two main parts:

- **Header** section (starts with @)
- **Alignment** section (one line per read)

Common Headers:

- `@HD`: Header version and sort order
- `@SQ`: Reference sequence (chromosome) name and length
- `@RG`: Read group (sample, library, platform info)
- `@PG`: Software and command used to generate the alignment

HISAT2 Output SAM/BAM

Let's use our UHR_Rep1 as an example to show you how does a SAM file looks like.
`less -S UHR_Rep1.sam`

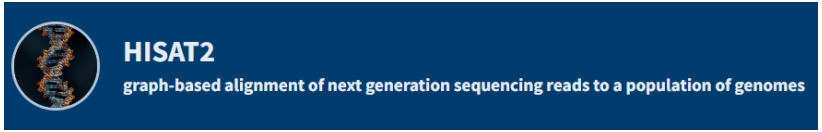
```
@HD     VN:1.0  SO:unsorted
@SQ     SN:22   LN:50818468
@SQ     SN:ERCC-00002   LN:1061
@SQ     SN:ERCC-00003   LN:1023
@SQ     SN:ERCC-00004   LN:523
@SQ     SN:ERCC-00009   LN:984
@SQ     SN:ERCC-00012   LN:994
```

These are the first few header lines in the file. We can see that:

- It is version 1.0 of this SAM file, and it is unsorted.
- There is a reference sequence called "22" and its length is 50818468 base pairs. This should be chromosome 22.
- There is another reference sequence called "ERCC-00002", and its length is 1061 base pairs.
- And then all our 92 sequences of the spike-in control.

Now, let's keep scrolling by pressing "space" bar, until where you find the end of header section.

HISAT2 Output SAM/BAM



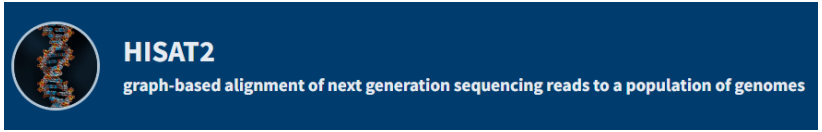

```
@SQ     SN:ERCC-00165   LN:872
@SQ     SN:ERCC-00168   LN:1024
@SQ     SN:ERCC-00170   LN:1023
@SQ     SN:ERCC-00171   LN:505
@PG     ID:hisat2       PN:hisat2       VN:2.2.1        CL:"/home/vdiuser/hisat2-2.2.1/hisat2-align-s --wrapper basic-0 -x reference/chr22_with_ERCC92_index
HWI-ST718_146963544:6:1213:8996:10047   99      22      15907711        60      100M    =       15907796        185     CTTTTTTATTTTTGTCTGACTGGGTTGATTCAAAGGT
HWI-ST718_146963544:6:1213:8996:10047   147     22      15907796        60      100M    =       15907711        -185    GCTAAGGCTGCCAACTCTTTTTGAAATTCCTATAGTA
HWI-ST718_146963544:5:2303:11793:37095  99      22      15905213        60      100M    =       15905307        194     ATGAATTATAGGGCTGTATTTTAATTTTGCATTTTAA
```

The @PG line gives you information about the software we have used to generate the SAM file, which is HISAT2 here. It also gives you the command ran if you scroll to right, you'll see the complete command.

We missed the @RG header line. It specifies read group metadata, which is crucial for **distinguishing between samples or technical replicates**. These information need to be manually added when running the aligner.

Previously we only used the most basic setting to run HISAT2, later we are going to run it with a more sophisticated setting.

HISAT2 Output SAM/BAM



Now, let's look at the Alignment Section in our SAM file.

HWI-ST718_146963544:6:1213:8996:10047	99	22	15907711	60	100M	=	15907796	185	CTTTTTTATTTTTGTCTGACTGGGTTGATTCAAAGGTCTG
HWI-ST718_146963544:6:1213:8996:10047	147	22	15907796	60	100M	=	15907711	-185	GCTAAGGCTGCCAACTCTTTTTGAAATTCCTATAGTAAAT
HWI-ST718_146963544:5:2303:11793:37095	99	22	15905213	60	100M	=	15905307	194	ATGAATTATAGGGCTGTATTTTAATTTTGCATTTTAAATT
HWI-ST718_146963544:5:2303:11793:37095	147	22	15905307	60	100M	=	15905213	-194	TTTGTGGCTTCTTGATCTTCTTTACTTGTATGTTATTGAT
HWI-ST718_146963544:6:2112:14109:7701	99	22	15903564	1	1S99M	=	15903682	219	GCCCTGATGTGATTATTACACATTGCATGCCTGTGTCAAA
HWI-ST718_146963544:6:2112:14109:7701	147	22	15903682	1	100M	=	15903564	-219	AGTGGGAGCTTTTAAAGGTGAGGTTTGCCCTCCAGCACTG
HWI-ST718_146963544:6:2112:14109:7701	355	22	10874150	1	1S99M	=	10874268	219	GCCCTGATGTGATTATTACACATTGCATGCCTGTGTCAAA
HWI-ST718_146963544:6:2112:14109:7701	403	22	10874268	1	100M	=	10874150	-219	AGTGGGAGCTTTTAAAGGTGAGGTTTGCCCTCCAGCACTG

- One TAB-separated line per read.
- Contains **11 mandatory fields** (columns) + optional tags.

The mandatory fields are:

1. Query Name (read ID)
2. Bitwise flag (info on read properties)
3. Reference name (e.g., 22)
4. Alignment start position (e.g., 15907711)
5. Mapping quality (0-255)
6. Alignment string ??
7. Mate reference name ("=" if same as reference name)
8. Mate alignment start position
9. Template length (insert size) ??
10. Read sequence
11. Quality score of read
12. Additional information

Bitwise Flag

The Bitwise Flag in a SAM file is a single integer value that **encodes multiple properties of a read** using binary bits. Each bit in this integer represents a specific attribute - like whether the read is mapped, is the first or second read in a pair, is on the reverse strand, etc.

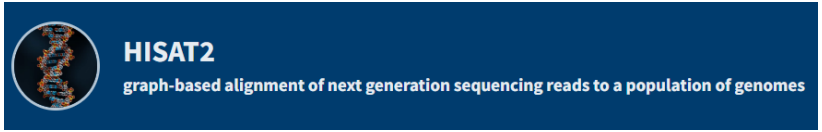
HWI-ST718_146963544:6:1213:8996:10047	99	22	15907711	60
HWI-ST718_146963544:6:1213:8996:10047	147	22	15907796	60
HWI-ST718_146963544:5:2303:11793:37095	99	22	15905213	60
HWI-ST718_146963544:5:2303:11793:37095	147	22	15905307	60
HWI-ST718_146963544:6:2112:14109:7701	99	22	15903564	1
HWI-ST718_146963544:6:2112:14109:7701	147	22	15903682	1
HWI-ST718_146963544:6:2112:14109:7701	355	22	10874150	1
HWI-ST718_146963544:6:2112:14109:7701	403	22	10874268	1

A bitwise flag of “99” means:

- Read is paired.
- Read is properly aligned.
- It’s the first read in a pair.
- It’s mapped on the forward strand.
- Its mate is mapped on the reverse strand.

Here is an online decoder for bitwise flag: https://broadinstitute.github.io/picard/explain-flags.html

Run HISAT2 Alignment



Now, let's try adding information about our sample when running the alignment.

Sometimes we might **combine SAM/BAM files** from multiple samples into one file for analyses. In that situation, a header line with some information about our sample would be great to distinguish them.

```
hisat2 -p 4 \
       --rg-id=UHR_Rep1 \
       --rg SM:UHR \
       --rg LB:UHR_Rep1_ERCC-Mix1 \
       --rg PL:ILLUMINA \
       --rg PU:HWI-ST718_146963544.5-6 \
       -x reference/chr22_with_ERCC92_index \
       --dta \
       --rna-strandness RF \
       -1 data/UHR_Rep1_ERCC-Mix1_Build37-ErccTranscripts-chr22.read1.fastq.gz \
       -2 data/UHR_Rep1_ERCC-Mix1_Build37-ErccTranscripts-chr22.read2.fastq.gz \
       -S align_result/UHR_Rep1.sam
```


Run HISAT2 Alignment

flag	meaning
-p 4	Use 4 CPU threads
--rg-id=UHR_Rep1	Read group ID
--rg SM:UHR	Sample name
--rg LB:UHR_Rep1_ERCC-Mix1	Library name
--rg PL:ILLUMINA	Platform used for sequencing
--rg PU:HWI-ST718_146963544.5-6	Platform unit
--dta	Enables output tailored for transcript assemblers like StringTie
--rna-strandness RF	Specify the strandness of the RNA-seq library

Platform Unit

The PU (platform unit) field is often used to uniquely identify the sequencing run, especially for downstream tools like **GATK** that need to **distinguish between lanes or barcodes**.

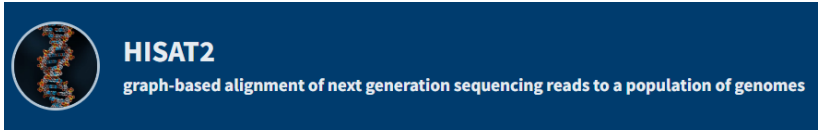
Normally we construct the PU like this:

- PU = Instrument:Flowcell:Lane:Barcode

Where to find the Platform Unit?

- FASTQ file:
  - if your FASTQ files come directly from a sequencer (like Illumina), the read headers contain PU-relevant information.
- The FASTQ header for UHR_Rep1:
  - `@HWI-ST718_146963544:6:1213:8996:10047/1`
  - `@HWI-ST718`: is the Instrument ID
  - `146963544`: is the unique ID for the run
  - `6`: is the lane number
  - `1213`: imaging tile on the flow cell
  - `8996:10047`: spatial coordinates of the cluster
  - `/1` : read 1 of a paired-end read

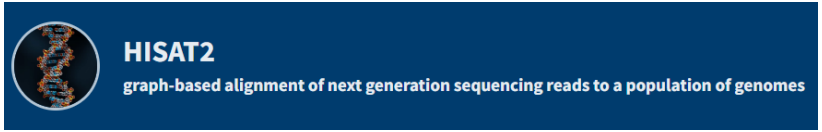
Run HISAT2 Alignment




```
227392 reads; of these:
  227392 (100.00%) were paired; of these:
    1155 (0.51%) aligned concordantly 0 times
    222491 (97.84%) aligned concordantly exactly 1 time
    3746 (1.65%) aligned concordantly >1 times
    ----
    1155 pairs aligned concordantly 0 times; of these:
      526 (45.54%) aligned discordantly 1 time
    ----
    629 pairs aligned 0 times concordantly or discordantly; of these:
      1258 mates make up the pairs; of these:
        510 (40.54%) aligned 0 times
        646 (51.35%) aligned exactly 1 time
        102 (8.11%) aligned >1 times
99.89% overall alignment rate
```

This time we have 99.89% overall alignment rate. Not much change from the previous alignment.

Run HISAT2 Alignment



For the 3 samples of UHR, the PU is the same since they are pooled together and sequenced in a single run. So, we can write a for loop and run 3 samples together.

```
for i in UHR_Rep1 UHR_Rep2 UHR_Rep3
do
hisat2 -p 4 --rg-id=${i} --rg SM:UHR --rg LB:${i}_ERCC-Mix1 --rg PL:ILLUMINA \
       --rg PU:HWI-ST718_146963544.5-6 \
       -x reference/chr22_with_ERCC92_index --dta --rna-strandness RF \
       -1 data/${i}_ERCC-Mix1_Build37-ErccTranscripts-chr22.read1.fastq.gz \
       -2 data/${i}_ERCC-Mix1_Build37-ErccTranscripts-chr22.read1.fastq.gz \
       -S align_result/${i}.sam
done
```

Exercise: try to write a similar loop to run alignment on the 3 samples of HBR.

The PU of HBR is `HWI-ST718_146963544.7-8`.

Convert SAM to BAM


```
(base) vdiuser@vdj-33xefq:~/RNAseq-Workshop$ ls -lh align_result/
total 703M
-rw-rw-r-- 1 vdiuser vdiuser  86M May  2 10:32 HBR_Rep1.sam
-rw-rw-r-- 1 vdiuser vdiuser 105M May  2 10:32 HBR_Rep2.sam
-rw-rw-r-- 1 vdiuser vdiuser  94M May  2 10:32 HBR_Rep3.sam
-rw-rw-r-- 1 vdiuser vdiuser 166M May  2 10:29 UHR_Rep1.sam
-rw-rw-r-- 1 vdiuser vdiuser 119M May  2 10:29 UHR_Rep2.sam
-rw-rw-r-- 1 vdiuser vdiuser 136M May  2 10:29 UHR_Rep3.sam
(base) vdiuser@vdj-33xefq:~/RNAseq-Workshop$
```

You can check if your result looks the same as mine.

BAM is the compressed and binary version of SAM. Normally BAM files are used to store alignment information instead of SAM to save space.

Except to convert SAM to BAM, we want to **sort the alignment** file as well.

Now, the alignment information is stored **in the order of the raw sequencing**, e.g., the first read in the FASTQ file would also be the first read in the SAM file.

Sort the Alignment

Many downstream tools require the input SAM/BAM file to be sorted by coordinates (the location it mapped to reference). So, in this step, we will sort the SAM and convert it to BAM by using SAMtools.

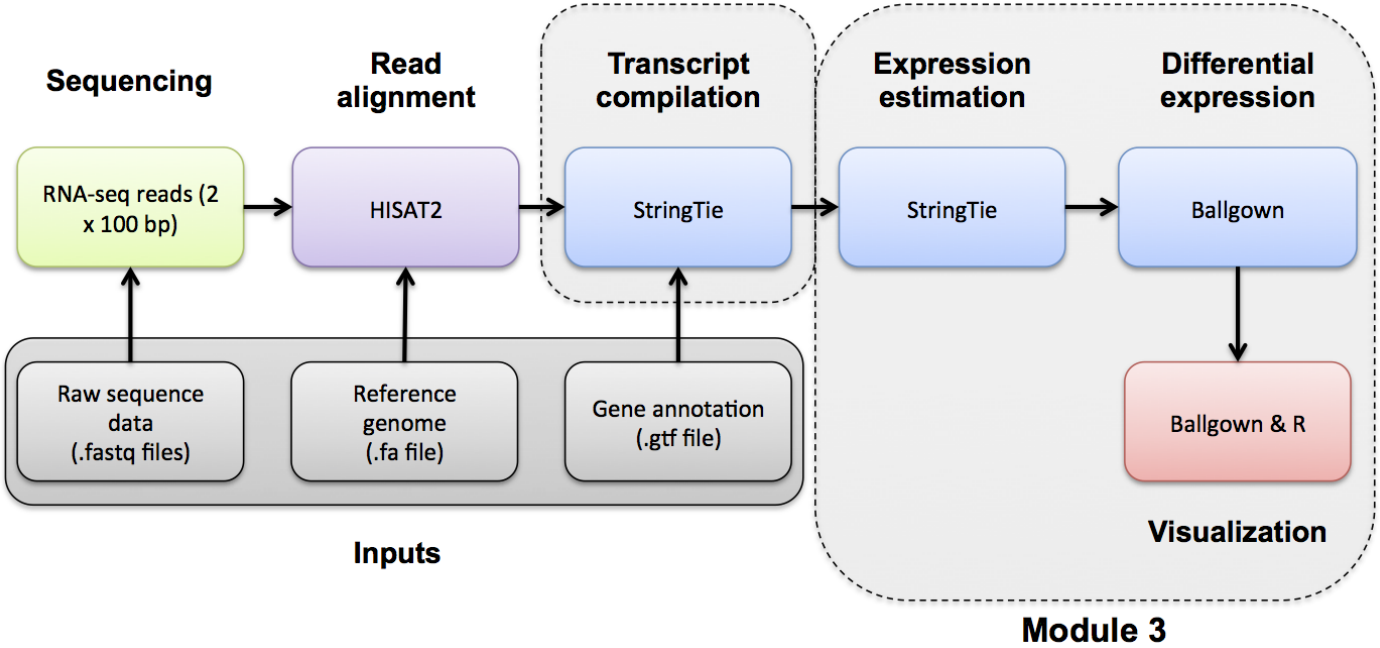
First, let's activate our conda environment: `conda activate RNAseq_env`.
Then, let's go to the align_result folder: `cd align_result`.

```
for i in UHR_Rep1 UHR_Rep2 UHR_Rep3 HBR_Rep1 HBR_Rep2 HBR_Rep3
do
samtools sort -@ 4 -o ${i}.bam ${i}.sam
done
```

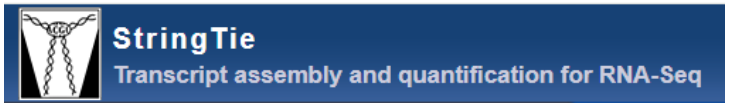
The BAM files are much smaller than SAM!

```
(RNAseq_env) vdiuser@vdj-33xefq:~/RNAseq-Workshop/align_result$ ls -lh
total 843M
-rw-rw-r-- 1 vdiuser vdiuser  17M May  2 10:52 HBR_Rep1.bam
-rw-rw-r-- 1 vdiuser vdiuser  86M May  2 10:32 HBR_Rep1.sam
-rw-rw-r-- 1 vdiuser vdiuser  20M May  2 10:52 HBR_Rep2.bam
-rw-rw-r-- 1 vdiuser vdiuser 105M May  2 10:32 HBR_Rep2.sam
-rw-rw-r-- 1 vdiuser vdiuser  18M May  2 10:52 HBR_Rep3.bam
-rw-rw-r-- 1 vdiuser vdiuser  94M May  2 10:32 HBR_Rep3.sam
-rw-rw-r-- 1 vdiuser vdiuser  34M May  2 10:52 UHR_Rep1.bam
-rw-rw-r-- 1 vdiuser vdiuser 166M May  2 10:29 UHR_Rep1.sam
-rw-rw-r-- 1 vdiuser vdiuser  25M May  2 10:52 UHR_Rep2.bam
-rw-rw-r-- 1 vdiuser vdiuser 119M May  2 10:29 UHR_Rep2.sam
-rw-rw-r-- 1 vdiuser vdiuser  28M May  2 10:52 UHR_Rep3.bam
-rw-rw-r-- 1 vdiuser vdiuser 136M May  2 10:29 UHR_Rep3.sam
(RNAseq_env) vdiuser@vdj-33xefq:~/RNAseq-Workshop/align_result$
```

Expression Analysis with StringTie



Expression Estimates with StringTie



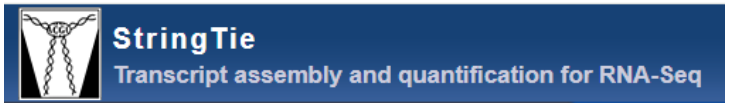
StringTie is a fast and highly efficient **RNA-Seq transcript assembler and quantifier**.

It takes **aligned RNA-seq reads** (in BAM format) and **reconstructs transcripts**, **estimate their abundance**, and can perform differential expression analysis when combined with other tools.

StringTie can assemble transcripts both **with and without** a reference annotation.

Here, we will assemble the transcripts by using our **chr22_ERCC** reference.

Expression Estimates with StringTie



First, let's create a new folder to store the result of StringTie.

```
`mkdir -p
~/RNAseq_Workshop/expression/stringtie/{UHR_Rep1,UHR_Rep2,UHR_Rep2,HBR_Rep1,HBR
_Rep2,HBR_Rep3}`
```

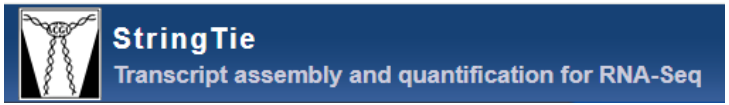
Then, activate our conda environment.

```
`conda activate RNAseq_env`
```

Run:

```
`stringtie --rf -p 4 -G reference/chr22_with_ERCC92.gtf -e -B \
          -o expression/stringtie/UHR_Rep1/transcripts.gtf \
          -A expression/stringtie/UHR_Rep1/gene_abundances.tsv \
          align_result/UHR_Rep1.bam`
```


Expression Estimates with StringTie



Exercise: write a for loop to run StringTie on all 6 samples.

Expression Estimates with StringTie

Flag	Meaning
--rf	strandness (1-reverse, 2-forward)
-p	threads
-G	reference annotation
-e	only estimate expression of known transcripts (from -G); do not assemble new ones
-B	output Ballgown-ready files in a subdirectory for downstream differential expression analysis
-o	output the sample-specific transcript-level expression in GTF format
-A	output gene-level abundance estimates (e.g. TPM, FPKM, coverage)

StringTie Outputs

Let's have a look of the StringTie outputs.

Go to folder `cd expression/stringtie/UHR_Rep1`

```
(RNAseq_env) vdiuser@vdj-33xefq:~/RNAseq-Workshop/expression/stringtie/UHR_Rep1$ ls
e2t.ctab      gene_abundances.tsv  i_data.ctab  transcripts.gtf
e_data.ctab   i2t.ctab             t_data.ctab
(RNAseq_env) vdiuser@vdj-33xefq:~/RNAseq-Workshop/expression/stringtie/UHR_Rep1$
```

The primary output of StringTie is a GTF file that contains details of the transcripts that StringTie assembles from RNA-Seq data.

Use `less -S transcripts.gtf` to view the output.

StringTie Outputs - transcripts

```
# stringtie --rf -p 4 -G reference/chr22_with_ERCC92.gtf -e -B -o expression/stringtie/UH>
# StringTie version 2.1.7
22      havana  transcript      15622601        15632051        .       -       .       g>
22      havana  exon    15622601        15622712        .       -       .       gene_id "
22      havana  exon    15623987        15624071        .       -       .       gene_id "
22      havana  exon    15624539        15624674        .       -       .       gene_id "
22      havana  exon    15624956        15625094        .       -       .       gene_id "
22      havana  exon    15625472        15625901        .       -       .       gene_id "
22      havana  exon    15627332        15627410        .       -       .       gene_id "
22      havana  exon    15627650        15627716        .       -       .       gene_id "
22      havana  exon    15628382        15628586        .       -       .       gene_id "
22      havana  exon    15629738        15629891        .       -       .       gene_id "
22      havana  exon    15631430        15631538        .       -       .       gene_id "
22      havana  exon    15631919        15632051        .       -       .       gene_id "
```

The first 2 lines start with a “#” are **header lines** with information about the command and the software version.

Following lines describe:

- The first line defines a **transcript** assembled or quantified by StringTie.
- The next lines define the **exons that make up that transcript**.

StringTie Outputs - transcripts

Use `grep -v "^#" transcripts.gtf | grep -w "transcript" | column -t | less -S` to view only lines for transcript.

Scroll to the end. There are information such as FPKM and TPM which are expression matrices of this transcript.

```
FPKM  "0.000000";       TPM  "0.000000";
FPKM  "0.000000";       TPM  "0.000000";
FPKM  "0.000000";       TPM  "0.000000";
FPKM  "2.885961";       TPM  "3.761353";
FPKM  "13.342950";      TPM  "17.390240";
FPKM  "2.102139";       TPM  "2.739777";
FPKM  "7.363133";       TPM  "9.596577";
FPKM  "0.000000";       TPM  "0.000000";
FPKM  "0.000000";       TPM  "0.000000";
FPKM  "0.000000";       TPM  "0.000000";
FPKM  "2.531229";       TPM  "3.299021";
FPKM  "0.000000";       TPM  "0.000000";
FPKM  "0.000000";       TPM  "0.000000";
FPKM  "0.000000";       TPM  "0.000000";
FPKM  "0.000000";       TPM  "0.000000";
FPKM  "0.000000";       TPM  "0.000000";
FPKM  "0.000000";       TPM  "0.000000";
```


FPKM and TPM

FPKM and TPM are both units used to measure gene or transcript expression levels from RNA-seq data.

They serve the same goal - **normalising raw read counts** so you can compare expression within or between samples.

FPKM - Fragments per Kilobase of transcript per Million mapped reads
TPM - Transcripts Per Million

Why normalise? Because raw read counts are affected by:

- Gene length (longer genes get more reads)
- Sequencing depth (more total reads = more counts)

So, normalisation is essential to make comparisons.

FPKM and TPM

FPKM = (Number of fragments) / (Gene length in kb * Total fragments in million)

- Normalise for gene length and sequencing depth
- Good for comparing genes within the same sample
- Not idea across samples, because FPKM totals don't sum to a consistent number

TPM_i = (FPKM_i / sum(FPKM_all)) * 1,000,000

- Good for comparing across samples

StringTie Output - gene abundance

The `gene_abundances.tsv` file from StringTie is a **gene-level expression summary** file, especially useful for downstream analysis or visualisation.

It is a tab-delimited text file containing **one row per gene**, with columns reporting expression values and gene-level metadata.

Use `cut -f1,2,7,8,9 gene_abundances.tsv | column -s $'\t' -t | less -S` to view the file.

Gene ID	Gene Name	Coverage	FPKM	TPM
ENSG00000224435	NF1P6	0.000000	0.000000	0.000000
ENSG00000198062	POTEH	0.580394	18.331051	23.891369
ENSG00000236666	POTEH-AS1	0.000000	0.000000	0.000000
ENSG00000212216	RNU6-816P	0.000000	0.000000	0.000000
ENSG00000241838	LA16c-3G11.7	0.282418	7.363133	9.596577
ENSG00000225255	LINC01297	0.000000	0.000000	0.000000
ENSG00000235992	GRAMD4P2	0.000000	0.000000	0.000000
ENSG00000198445	CCT8L2	0.097087	2.531229	3.299021
ENSG00000239435	KCNMB3P1	0.000000	0.000000	0.000000

StringTie calculates this by **aggregating expression across all transcripts of the same gene** (based on your reference annotation).

Create a tidy expression matrix

Here, we will use a script that written by the Griffith Lab to combine expression estimates from all 6 samples and clean them up.

First, `cd expression/stringtie/`

Then download the script `wget https://raw.githubusercontent.com/griffithlab/rnabio.org/master/assets/scripts/stringtie_expression_matrix.pl`

Run `chmod +x stringtie_expression_matrix.pl` to add execution rights to the file.

```
(RNAseq_env) vdiuser@vdj-33xefq:~/RNAseq-Workshop/expression/stringtie$ chmod +x stringtie_expression_matrix.pl
(RNAseq_env) vdiuser@vdj-33xefq:~/RNAseq-Workshop/expression/stringtie$ ls -lh
total 32K
drwxrwxr-x 2 vdiuser vdiuser 4.0K May  3 11:37 HBR_Rep1
drwxrwxr-x 2 vdiuser vdiuser 4.0K May  3 11:37 HBR_Rep2
drwxrwxr-x 2 vdiuser vdiuser 4.0K May  3 11:37 HBR_Rep3
-rwxrwxr-x 1 vdiuser vdiuser 6.6K May  3 11:40 stringtie_expression_matrix.pl
drwxrwxr-x 2 vdiuser vdiuser 4.0K May  3 11:37 UHR_Rep1
drwxrwxr-x 2 vdiuser vdiuser 4.0K May  3 11:37 UHR_Rep2
drwxrwxr-x 2 vdiuser vdiuser 4.0K May  3 11:37 UHR_Rep3
```

Create a tidy expression matrix

Run below to get the matrix for TPM.

```
./stringtie_expression_matrix.pl \
        --expression_metric=TPM \
        --result_dirs='HBR_Rep1,HBR_Rep2,HBR_Rep3,UHR_Rep1,UHR_Rep2,UHR_Rep3' \
        --transcript_matrix_file=transcript_tpm_all_samples.tsv \
        --gene_matrix_file=gene_tpm_all_samples.tsv
```

Run below to get the matrix for FPKM.

```
./stringtie_expression_matrix.pl \
        --expression_metric=FPKM \
        --result_dirs='HBR_Rep1,HBR_Rep2,HBR_Rep3,UHR_Rep1,UHR_Rep2,UHR_Rep3' \
        --transcript_matrix_file=transcript_fpkm_all_samples.tsv \
        --gene_matrix_file=gene_fpkm_all_samples.tsv
```


Create a tidy expression matrix

Run below to get the matrix for Coverage.

```
./stringtie_expression_matrix.pl \
        --expression_metric=Coverage \
        --result_dirs='HBR_Rep1,HBR_Rep2,HBR_Rep3,UHR_Rep1,UHR_Rep2,UHR_Rep3' \
        --transcript_matrix_file=transcript_coverage_all_samples.tsv \
        --gene_matrix_file=gene_coverage_all_samples.tsv
```

```
(RNAseq_env) vdiuser@vdj-33xefq:~/RNAseq-Workshop/expression/stringtie$ ls -lh
total 1.2M
-rw-rw-r-- 1 vdiuser vdiuser  84K May  3 11:49 gene_coverage_all_samples.tsv
-rw-rw-r-- 1 vdiuser vdiuser  89K May  3 11:48 gene_fpkm_all_samples.tsv
-rw-rw-r-- 1 vdiuser vdiuser  89K May  3 11:48 gene_tpm_all_samples.tsv
drwxrwxr-x 2 vdiuser vdiuser 4.0K May  3 11:37 HBR_Rep1
drwxrwxr-x 2 vdiuser vdiuser 4.0K May  3 11:37 HBR_Rep2
drwxrwxr-x 2 vdiuser vdiuser 4.0K May  3 11:37 HBR_Rep3
-rwxrwxr-x 1 vdiuser vdiuser 6.6K May  3 11:40 stringtie_expression_matrix.pl
-rw-rw-r-- 1 vdiuser vdiuser 235K May  3 11:49 transcript_coverage_all_samples.tsv
-rw-rw-r-- 1 vdiuser vdiuser 305K May  3 11:48 transcript_fpkm_all_samples.tsv
-rw-rw-r-- 1 vdiuser vdiuser 306K May  3 11:48 transcript_tpm_all_samples.tsv
```


View the cleaned matrix

Run `column -t gene_tpm_all_samples.tsv | less -S` to view the TPM estimates for all samples and all genes.

Gene_ID	HBR_Rep1	HBR_Rep2	HBR_Rep3	UHR_Rep1	UHR_Rep2	UHR_Rep3
ENSG00000008735	1163.456665	1275.826050	1269.005737	20.970343	36.918056	13.452017
ENSG00000015475	207.458893	176.533752	265.524353	308.163177	258.963837	334.818298
ENSG00000025708	92.550392	81.754044	141.377975	170.258911	96.027733	111.648132
ENSG00000025770	549.929688	495.314636	495.945282	729.647095	604.227234	649.800537
ENSG00000040608	438.414642	454.538605	447.433838	55.292873	53.429340	35.440155
ENSG00000054611	124.497116	147.903412	145.139175	113.788231	104.756676	129.794113
ENSG00000056487	54.545818	32.842770	15.734045	18.614506	28.115952	59.088139
ENSG00000063515	0.0	0.0	0.0	0.0	0.0	0.0
ENSG00000069998	145.511261	153.356567	194.077118	384.906738	337.415466	511.992462
ENSG00000070010	156.468674	204.794724	170.608826	494.911285	407.196045	411.472351
ENSG00000070371	64.032578	75.685326	79.053749	107.172722	93.338600	110.794441
ENSG00000070413	860.786499	813.743774	790.778137	422.234375	348.692749	436.592529

Run `column -t transcript_tpm_all_samples.tsv | less -S` to view the TPM estimates for all samples and all transcripts.

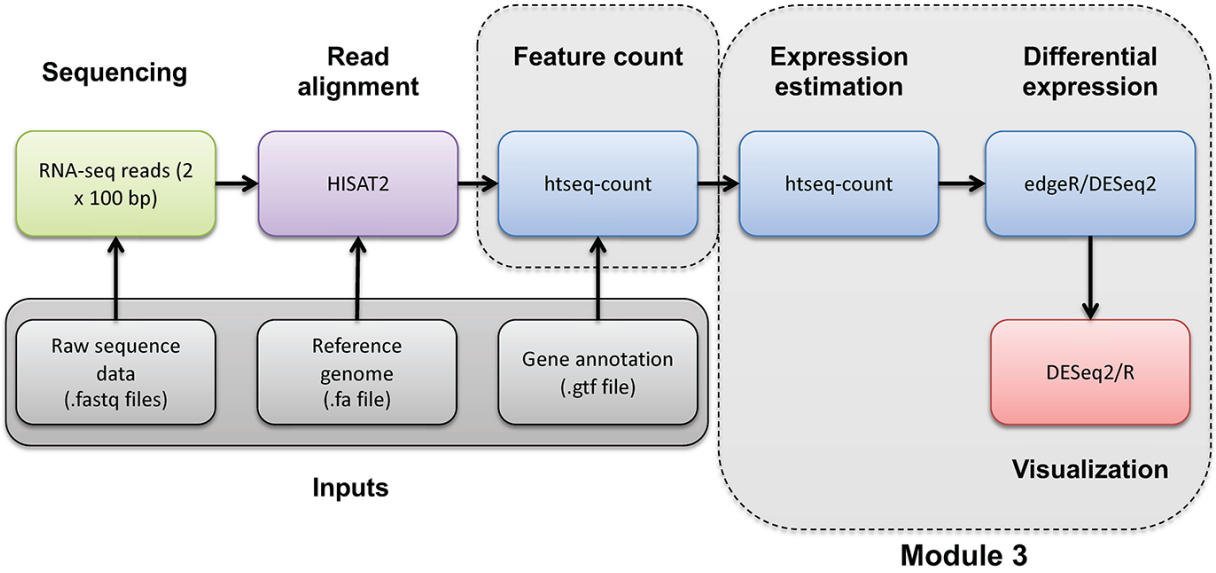
Transcript_ID	HBR_Rep1	HBR_Rep2	HBR_Rep3	UHR_Rep1	UHR_Rep2	UHR_Rep3
ENST00000006251	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
ENST00000008876	464.860199	550.127014	512.989258	1.786603	12.201631	4.759975
ENST00000043402	422.633118	427.446259	443.219299	15.573021	23.150223	35.440155
ENST00000086933	0.0	0.0	0.0	0.0	0.0	0.0
ENST00000155674	0.000000	0.000000	0.000000	8.185657	34.239082	0.000000
ENST00000159647	184.042191	64.080864	0.000000	10.475878	0.000000	0.000000
ENST00000207636	43.293640	13.589554	14.269886	66.044769	0.000000	40.448284
ENST00000215659	0.000000	68.789299	23.351023	61.521519	40.252106	98.702591
ENST00000215727	0.000000	8.778346	0.000000	334.839813	313.282471	305.822784
ENST00000215730	117.284706	130.425171	121.520920	168.073318	175.209061	141.162140
ENST00000215739	168.574066	297.268158	142.888931	257.711334	203.351059	292.096710

Generate raw counts with `htseq-count`

HTSeq: High-throughput sequence analysis in Python

Author	Fabio Zanini, Simon Anders, Givanna Putri and contributors
Date	Dec 14, 2023
Version	2.0.5

`htseq-count`: counting reads within features



Generate raw counts with `htseq-count`

htseq-count is part of HTSeq (https://htseq.readthedocs.io/en/latest/index.html).

It is used to **count how many reads map to each gene** in RNA-seq data. It outputs a table of gene-level counts that you can use for downstream differential expression analysis (e.g., in **DESeq2** or **edgeR**).

First, let's create a new folder to store our htseq-count result.

```
`mkdir -p ~/RNAseq-Workshop/expression/htseq_counts`
`cd ~/RNAseq-Workshop`
```

Run htseq-count for **UHR_Rep1**:

```
htseq-count --format bam --order pos --mode intersection-strict --stranded reverse \
            --minaqual 1 --type exon --idattr gene_id \
            align_result/UHR_Rep1.bam \
            reference/chr22_with_ERCC92.gtf > expression/htseq_counts/UHR_Rep1_gene.tsv
```


htseq-count Flag Explained

Flag	Meaning
--format bam	specify the input file format
--order pos	provide the sort order of the input
--mode intersection-strict	determine how to deal with reads that overlap more than one feature
--stranded reverse	specify strandness of input
--minaqual 1	will skip reads with alignment quality lower than specified
--type exon	specify feature type
--idattr gene_id	group feature counts by gene_id

Generate raw counts with `htseq-count`

```
(RNAseq_env) vdiuser@vdj-33xefq:~/RNAseq-Workshop$ htseq-count --format bam --order pos --mode intersection-strict
 --stranded reverse --minaqual 1 --type exon --idattr gene_id align_result/UHR_Rep1.bam reference/chr22_with_ERCC9
2.gtf > expression/htseq_counts/UHR_Rep1_gene.tsv
[E::idx_find_and_load] Could not retrieve index file for 'align_result/UHR_Rep1.bam'
56295 GFF lines processed.
[E::idx_find_and_load] Could not retrieve index file for 'align_result/UHR_Rep1.bam'
100000 alignment record pairs processed.
200000 alignment record pairs processed.
227392 alignment record pairs processed.
```

You will see warning messages saying, "could not retrieve index for bam file". It's okay htseq-count doesn't need an index file to run.

Exercise: write a for loop to run for all samples.

```
(RNAseq_env) vdiuser@vdj-33xefq:~/RNAseq-Workshop/expression/htseq_counts$ ls -lh
total 168K
-rw-rw-r-- 1 vdiuser vdiuser 25K May  3 13:03 HBR_Rep1_gene.tsv
-rw-rw-r-- 1 vdiuser vdiuser 25K May  3 13:03 HBR_Rep2_gene.tsv
-rw-rw-r-- 1 vdiuser vdiuser 25K May  3 13:03 HBR_Rep3_gene.tsv
-rw-rw-r-- 1 vdiuser vdiuser 26K May  3 13:02 UHR_Rep1_gene.tsv
-rw-rw-r-- 1 vdiuser vdiuser 25K May  3 13:02 UHR_Rep2_gene.tsv
-rw-rw-r-- 1 vdiuser vdiuser 25K May  3 13:03 UHR_Rep3_gene.tsv
(RNAseq_env) vdiuser@vdj-33xefq:~/RNAseq-Workshop/expression/htseq_counts$
```


View the output from htseq-count

Use `less -S UHR_Rep1_gene.tsv` to view the output file generated by htseq-count.

```
ENSG00000008735 13
ENSG00000015475 103
ENSG00000025708 46
ENSG00000025770 188
ENSG00000040608 18
ENSG00000054611 70
ENSG00000056487 12
ENSG00000063515 0
ENSG00000069998 123
ENSG00000070010 228
ENSG00000070371 82
ENSG00000070413 297
```

A simple table to understand.

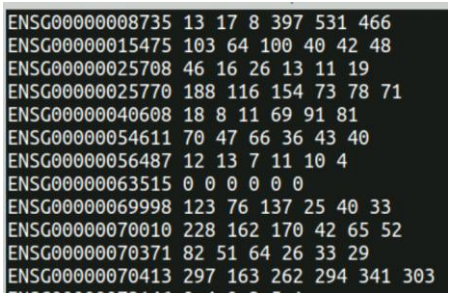
Now, we are going to merge all the files so it's ready for input to **edgeR**.

`join UHR_Rep1_gene.tsv UHR_Rep2_gene.tsv | join - UHR_Rep3_gene.tsv | join - HBR_Rep1_gene.tsv | join - HBR_Rep2_gene.tsv | join - HBR_Rep3_gene.tsv > gene_read_counts_table_all.tsv`

The `join` command can combine two text files based on a common field (usually the first column).

View the output from htseq-count

The joined table looks like this, we need to add a title for it.



```
ENSG00000008735 13 17 8 397 531 466
ENSG00000015475 103 64 100 40 42 48
ENSG00000025708 46 16 26 13 11 19
ENSG00000025770 188 116 154 73 78 71
ENSG00000040608 18 8 11 69 91 81
ENSG00000054611 70 47 66 36 43 40
ENSG00000056487 12 13 7 11 10 4
ENSG00000063515 0 0 0 0 0 0
ENSG00000069998 123 76 137 25 40 33
ENSG00000070010 228 162 170 42 65 52
ENSG00000070371 82 51 64 26 33 29
ENSG00000070413 297 163 262 294 341 303
```

`` `echo "GeneID UHR_Rep1 UHR_Rep2 UHR_Rep3 HBR_Rep1 HBR_Rep2 HBR_Rep3" > header.txt` `` this creates the title and save it to a txt file.

`` `cat header.txt gene_read_counts_table_all.tsv | grep -v "__" | awk -v OFS="\t" '$1=$1' > gene_read_counts_table_all_final.tsv` `` this combine the header and combines counts file together, and remove the end lines that start with __, and then use tab as the delimiter.

View the output from htseq-count

Use `head gene_read_counts_table_all_final.tsv | column -t` to have a look of the joined table.

```
(RNAseq_env) vdiuser@vdj-33xefq:~/RNAseq-Workshop/expression/htseq_counts$ head gene_read_counts_table_all_final.t
sv | column -t
GeneID           UHR_Rep1  UHR_Rep2  UHR_Rep3  HBR_Rep1  HBR_Rep2  HBR_Rep3
ENSG00000008735  13        17        8         397       531       466
ENSG00000015475  103       64        100       40        42        48
ENSG00000025708  46        16        26        13        11        19
ENSG00000025770  188       116       154       73        78        71
ENSG00000040608  18        8         11        69        91        81
ENSG00000054611  70        47        66        36        43        40
ENSG00000056487  12        13        7         11        10        4
ENSG00000063515  0         0         0         0         0         0
ENSG00000069998  123       76        137       25        40        33
(RNAseq_env) vdiuser@vdj-33xefq:~/RNAseq-Workshop/expression/htseq_counts$
```

The output from htseq-count are using Ensemble IDs instead of gene names or symbols. This is not very convenient for biological interpretation. The following command creates a mapping for Ensemble ID and gene names.

```
`cut -f 9 ../../reference/chr22_with_ERCC92.gtf | tr -d '"' | perl -ne 'chomp; if ($_ =~
/gene_id\s+(\S+);/){$gid = $1}; if ($_ =~ /gene_name\s+(\S+);/){$gname = $1}; print
"$gid\t$gname\n"' | sort | uniq > ENSG_ID2Name.txt`
```


View the output from htseq-count

Take a look of this mapping file. `less ENSG_ID2Name.txt`

```
ENSG00000008735 MAPK8IP2
ENSG00000015475 BID
ENSG00000025708 TYMP
ENSG00000025770 NCAPH2
ENSG00000040608 RTN4R
ENSG00000054611 TBC1D22A
ENSG00000056487 PHF21B
ENSG00000063515 GSC2
ENSG00000069998 CECR5
ENSG00000070010 UFD1L
ENSG00000070371 CLTCL1
```


Thank you

Contact us

Jiajia Li
Biological Data Science Institute

RN Robertson Building, 46 Sullivan's Creek Rd
The Australian National University
Canberra ACT 2600

E jiajia.li1@anu.edu.au
W https://bdsi.anu.edu.au/

Australian National University