

De novo transcriptome assembly using Trinity

Robert Bukowski, Qi Sun

Bioinformatics Facility

Institute of Biotechnology

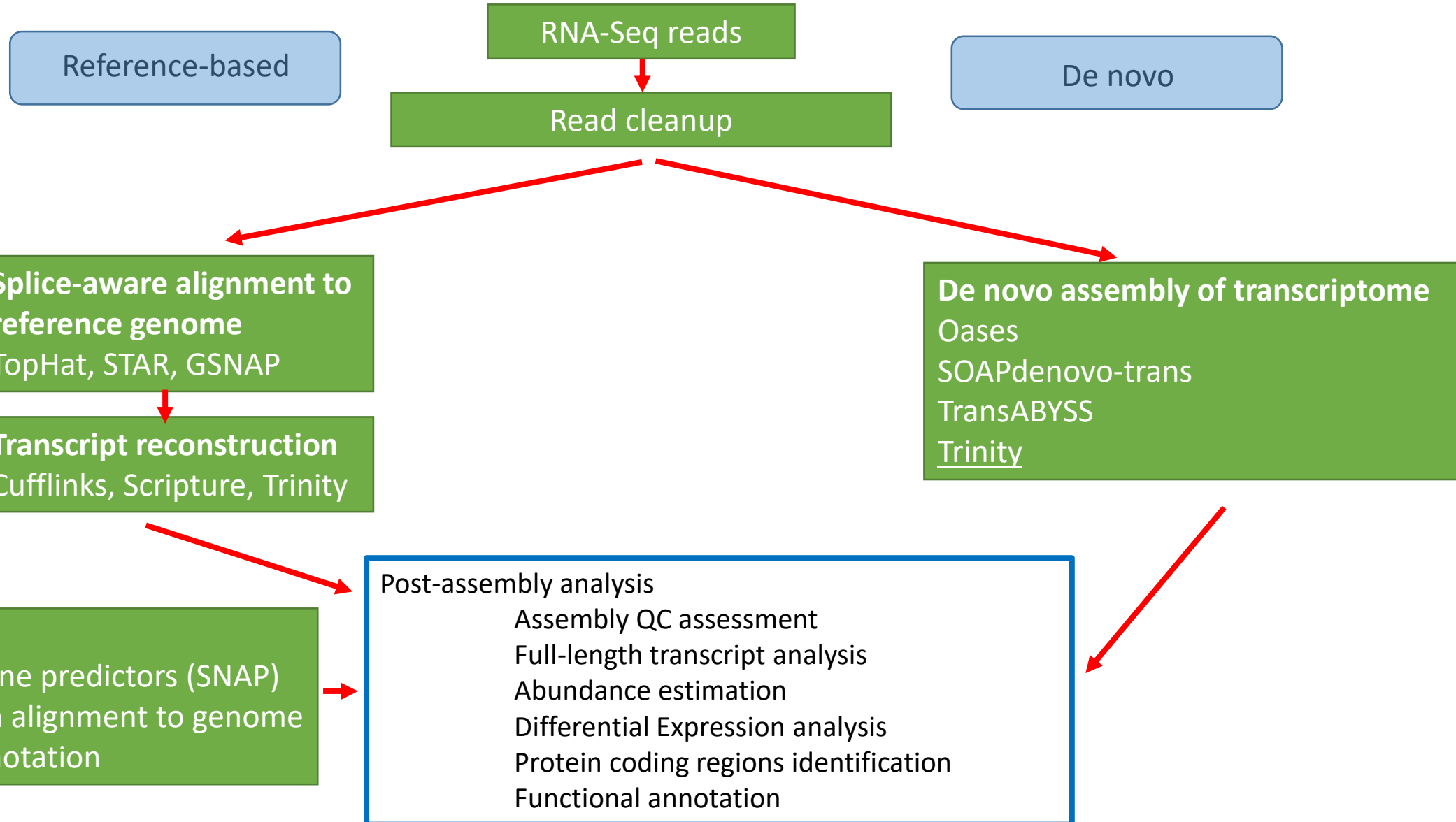
Slides: http://biohpc.cornell.edu/lab/doc/Trinity_workshop.pdf

Exercise instructions: http://biohpc.cornell.edu/lab/doc/Trinity_exercise1.pdf

http://biohpc.cornell.edu/lab/doc/Trinity_exercise2.pdf

Workshop office hours: **Friday, Oct. 25, 3:30-5pm, Rhodes 624**
(no appointment required)

Strategies for transcriptome assembly from RNA-Seq data



Trinity: state of the art de novo RNA-Seq assembly and analysis package

Assembler



(this part's topic)

Variety of post-assembly tools

- Assembly QC*
- Abundance estimation*
- Differential Expression analysis*
- Protein coding regions identification
- Functional annotation*
- Full-length transcript analysis*

(second part's topic)

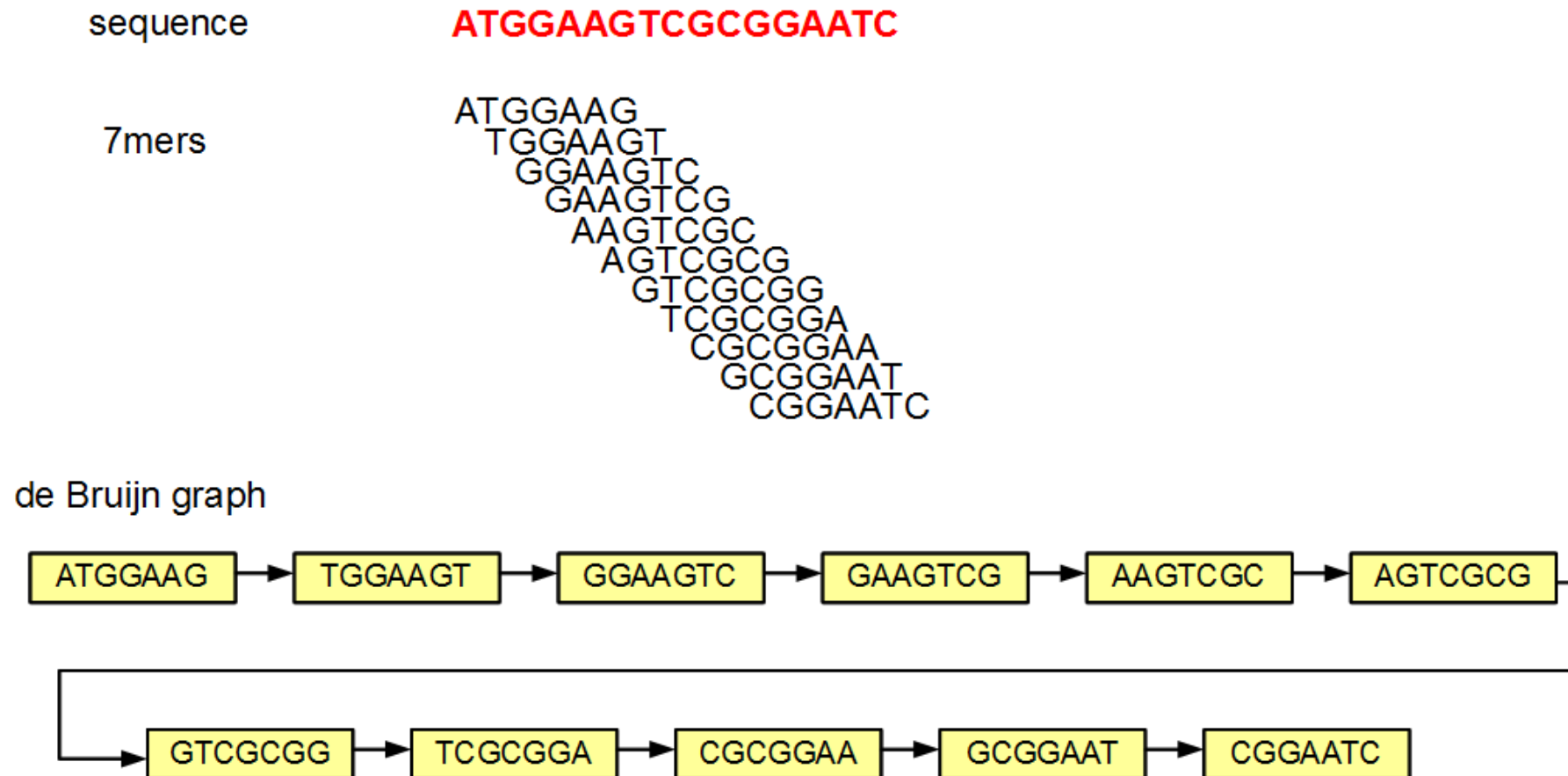
Developed at **Broad Institute** and **Hebrew University of Jerusalem**

N. G Grabherr et. al., Nature Biotechnology 29, 644–652 (2011) doi:10.1038/nbt.1883

B. J. Haas et. al., Nature Protocols 8, 1494–1512 (2013) doi:10.1038/nprot.2013.084

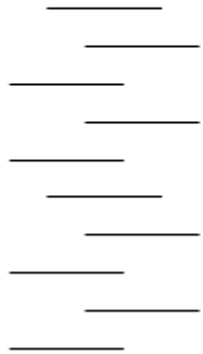
As other short read assemblers, Trinity works in K-mer space (K=25)

What are K-mers?

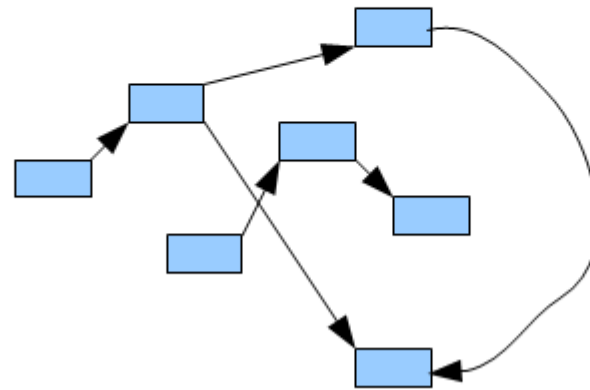


What an assembler is trying to do

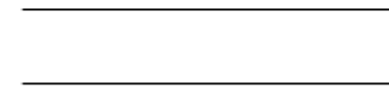
NGS library



de Bruijn Graph



Genome



K-mer extraction
from reads (and
counting)

Path validation using
reads

Complications:

- Sequencing errors add nodes (and complexity)
- Short or long K-mers?
- No positional info on repeats (less important for transcriptomes)

Three stages of Trinity

0. Jellyfish

- Extracts and counts K-mers (K=25, up to K=32 possible) from reads

a. Inchworm:

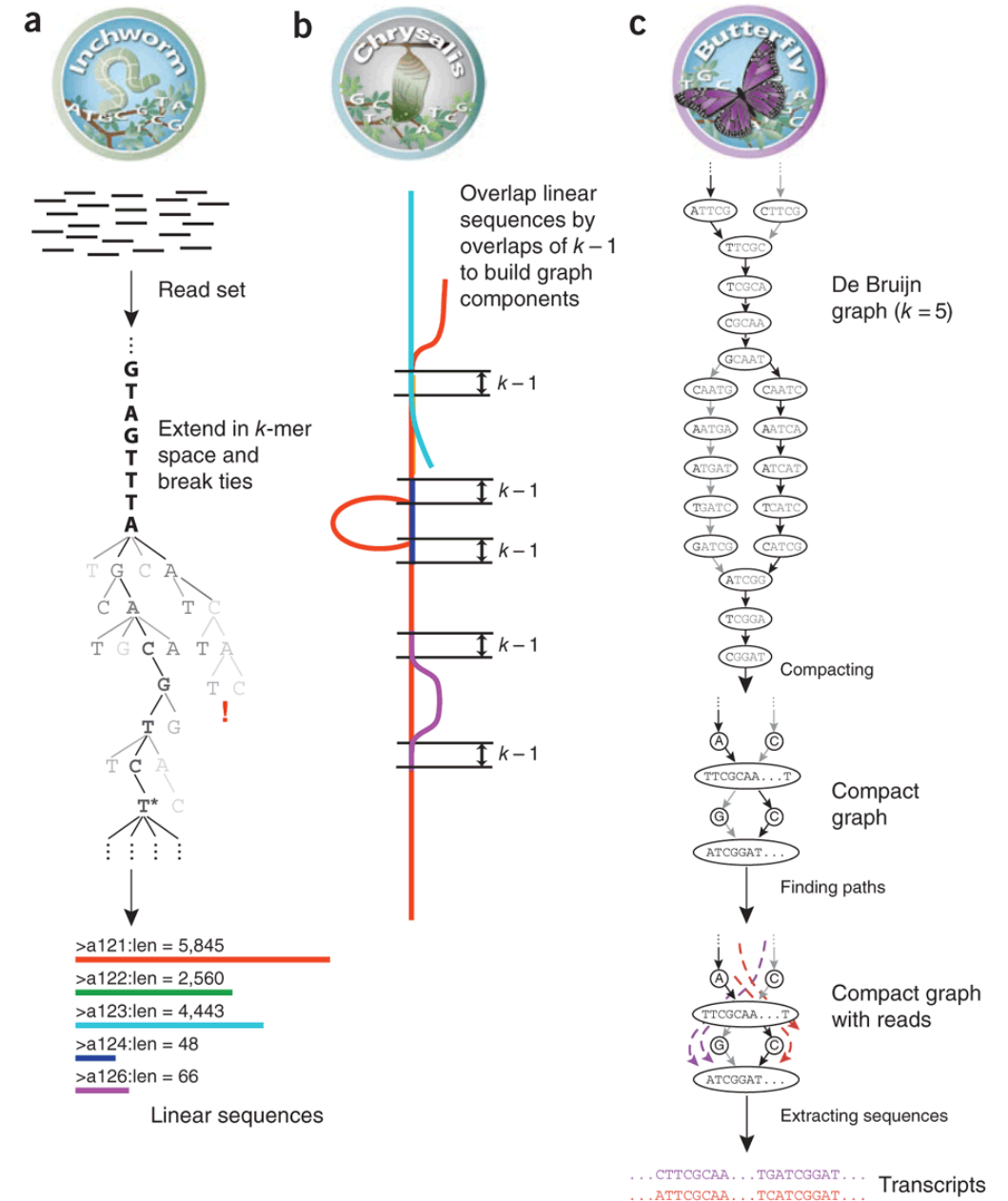
- Assembles initial contigs by “greedily” extending sequences with most abundant K-mers

b. Chrysalis:

- Clusters overlapping Inchworm contigs, builds de Bruijn graphs for each cluster, partitions reads between clusters

c. Butterfly:

- resolves alternatively spliced and paralogous transcripts independently for each cluster (in parallel)



Trinity programs

Trinity proper

- **Trinity** (perl script to glue it all together)
- **Inchworm**
- **Chrysalis**
- **Butterfly** (Java code)
- various **utility and analysis scripts** (in perl)

Bundled third-party software

- **Trimmomatic**: clean up reads by trimming and removing adapter remnants (Bolger, A. M., Lohse, M., & Usadel, B)
- **ParaFly**: parallel driver (Broad Institute)
- **Slust**: a utility that performs single-linkage clustering with the option of applying a Jaccard similarity coefficient to break weakly bound clusters into distinct clusters (Brian Haas)
- **Collectl** : system performance monitoring (Peter Seger)
- Post-assembly analysis helper scripts (in perl)

External software assembler depends on

- **Jellyfish** (k-mer counting software), **samtools**, **Bowtie2**, **kallisto**, **salmon** (alignment-free abundance estimation)

External software used in post-assembly analysis

- **R** with **Bioconductor**, **Blast**, **Picard**, **GATK4**, **Hisat2**, **STAR**
- **RSEM**, **eXpress**: alignment-based abundance estimation (Bo Li and Colin Dewey)
- **Transcoder**: identify candidate coding regions in within transcripts (Brian Haas - Broad, Alexie Papanicolaou – CSIRO)

Notation convention used on the following slides

Trinity commands will be abbreviated using the variable `TRINITY_HOME` to denote the location of the Trinity package

```
TRINITY_HOME=/programs/trinityrnaseq-v2.8.6
```

(this is the latest version installed on BioHPC Cloud)

Thus, a command

```
$TRINITY_HOME/Trinity [options]
```

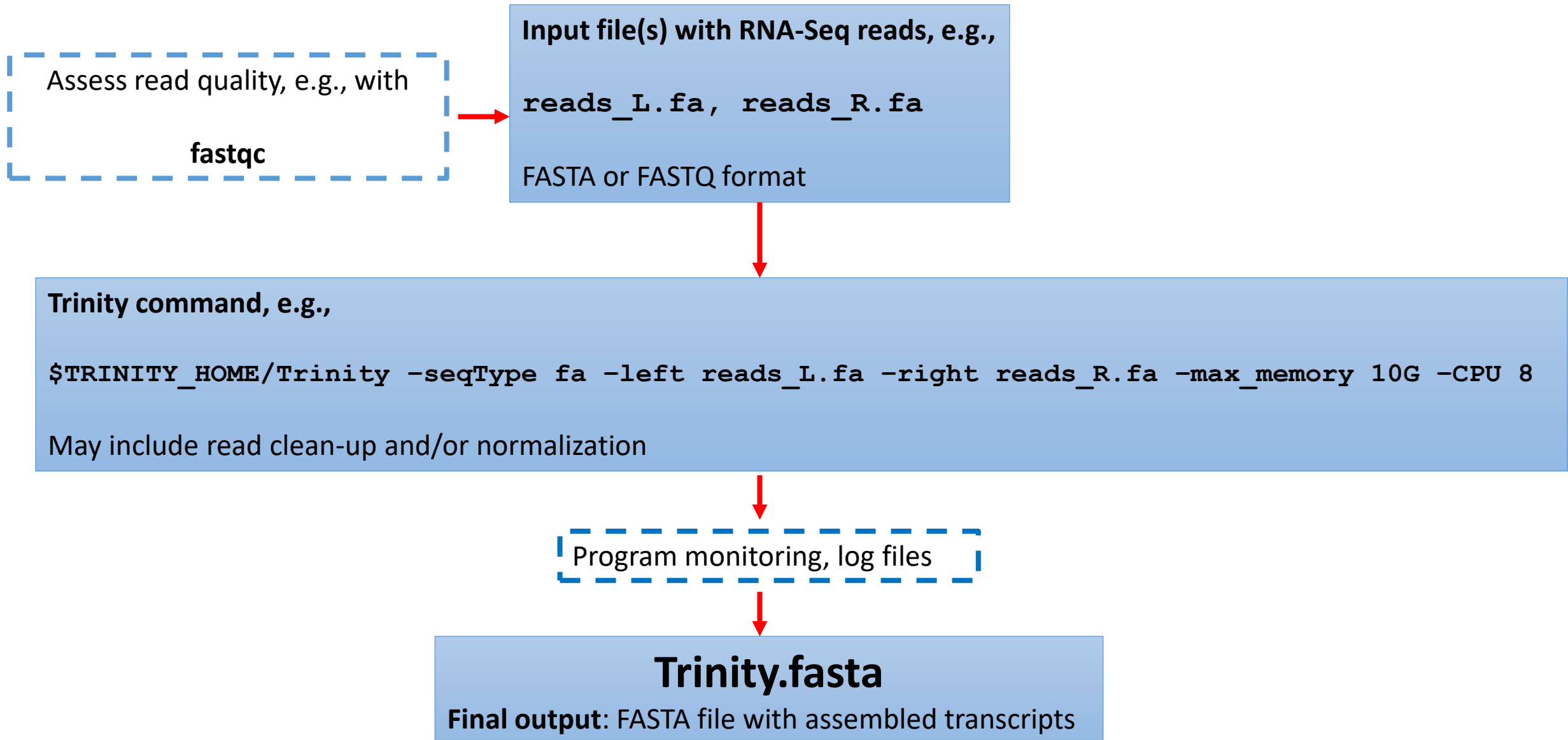
really means

```
/programs/trinityrnaseq-v2.8.6/Trinity [options]
```

The same notation is used in auxiliary shell scripts containing Trinity commands, used in the Exercise

Pipeline complicated, but easy to run

(in principle, just a single command)



Basic Trinity command dissected...

```
#!/bin/bash

$TRINITY_HOME/Trinity --seqType fq \
--left Sp_ds.left.fq.gz,Sp_hs.left.fq.gz,Sp_log.left.fq.gz,Sp_plat.left.fq.gz \
--right Sp_ds.right.fq.gz,Sp_hs.right.fq.gz,Sp_log.right.fq.gz,Sp_plat.right.fq.gz \
--SS_lib_type RF \
--max_memory 2G \
--CPU 2 \
--output /workdir/bukowski/trinity_out
```

NOTES:

- Use all reads from an individual (all conditions) to capture most genes
- Read files may be gzipped (as in this example) or not (then they should not have the “.gz” ending)
- Paired-end reads specified with **--left** and **--right**. If only single-end, use **--single** instead.
- 2G is the maximum memory to be used at any stage which allows memory limitation (jellyfish, sorting, etc.)
- At most 2 CPU cores will be used in any stage (some stages have their own CPU limits)
- Final output and intermediate files will be written in directory **/workdir/bukowski/my_trinity_out**
- **--SS_lib_type RF**: The PE fragments are strand-specific, with left end on the **Reverse** strand and the right end on Forward strand of the sequenced mRNA template
 - For non-strand specific reads, just skip the option **--SS_lib_type**

Final output: assembled transcriptome

FASTA file called **Trinity.fasta** (located in the run output directory)

```
>TRINITY_DN1000|c115_g5_i1 len=247 path=[31015:0-148 23018:149-246]  
AATCTTTTTTGGTATTGGCAGTACTGTGCTCTGGGTAGTGATTAGGGCAAAGAAGACAC  
ACAATAAAGAACCAGGTGTTAGACGTCAGCAAGTCAAGGCCTTGGTTCTCAGCAGACAGA  
AGACAGCCCTTCTCAATCCTCATCCCTTCCCTGAACAGACATGTCTTCTGCAAGCTTCTC  
CAAGTCAGTTGTTACAGGAACATCATCAGAATAAATTTGAAATTATGATTAGTATCTGA  
TAAAGCA
```

TRINITY_DN1000|c115_g5 is a **gene** identifier
i1 is an **isoform** identifier

The rest of the header shows nodes of de Buijn graph traversed by the transcript.

Advanced (but important) Trinity options to consider

General issues with de novo transcriptome assembly of RNA-Seq data

RNA-Seq reads often need pre-processing before assembly

- remove barcodes and Illumina adapters from reads
- clip read ends of low base quality
- **Trimmomatic** (<http://www.usadellab.org/cms/?page=trimmomatic>)
- remove contamination with other species

Very non-uniform coverage

- highly vs lowly expressed genes
- high-coverage in some regions may imply more sequencing errors complicating assembly down the road
 - some “normalization” of read set needed (but some researchers do not like it as genes may be missed when reads are removed!)

Gene-dense genomes pose a challenge (assembly may produce chimeric transcripts)

- overlapping genes on opposite strands (strand-specific RNA-Seq protocols may help)
- some overlap of genes on the same strand (harder to handle)

Pre-assembly read clean-up: Trimmomatic

Trimmomatic is bundled with Trinity

To invoke from within Trinity, add option

```
--trimmomatic
```

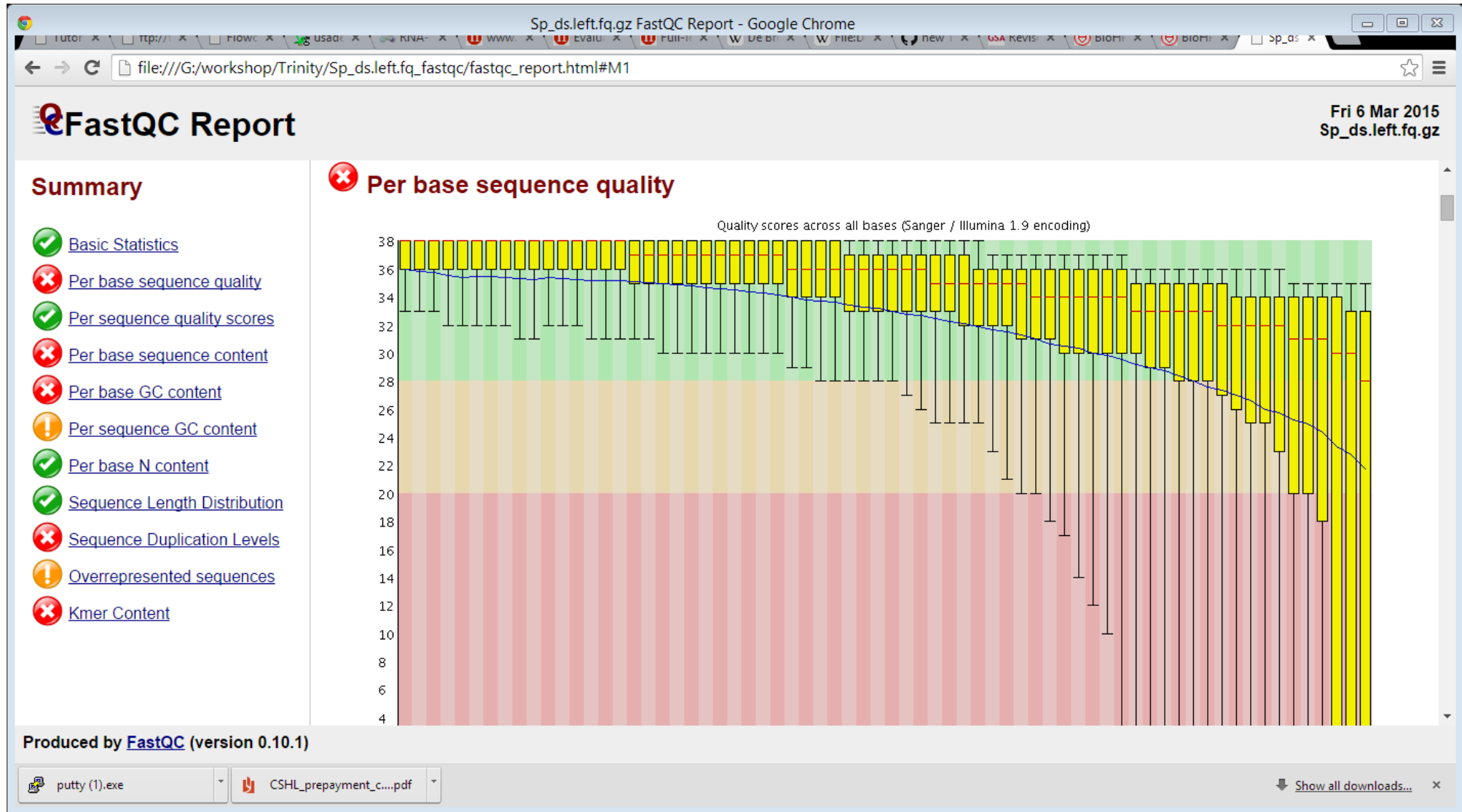
Also, one can request specific **Trimmomatic** settings by adding something similar to

```
--quality_trimming_params "ILLUMINACLIP:TruSeq3-PE.fa:2:30:10 SLIDINGWINDOW:6:5  
LEADING:20 TRAILING:10 MINLEN:30"
```

Trinity will use **all surviving reads**, treating the un-paired ones as single-end.

How to choose trimming parameters: Read quality assessment with fastqc

Many tests are expected to fail due to RNA_Seq characteristics



Run fastqc before running Trinity (part of the Exercise)

Dealing with very deep sequencing data

- Done to identify genes with low expression
- Several hundreds of millions of reads involved
- More sequencing errors possible with large depths, increasing the graph complexity

Suggested Trinity options/treatments:

- Use option

`--min_kmer_cov 2`

(singleton K-mers will not be included in initial Inchworm contigs)

Adjust
(default:200)

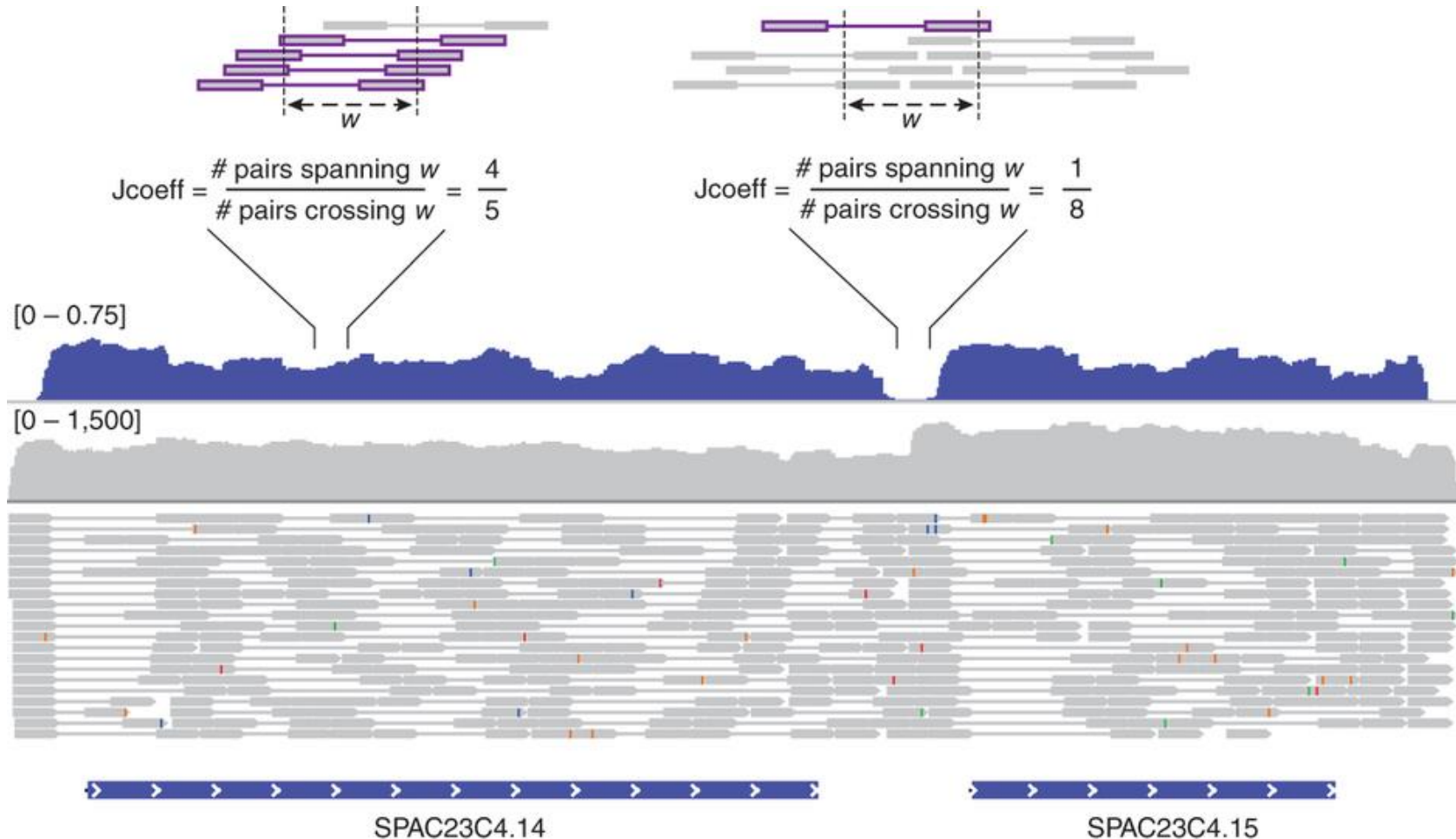


- Perform **K-mer based insilico read set normalization** (`--normalize_reads --normalize_max_read_cov 30`)
 - Normalization is default. Can be turned off with `--no_normalize_reads`
 - Effect: poorly covered regions unchanged, but reads down-sampled in high-coverage regions
 - May end up using just 20% of all reads reducing computational burden with no impact on assembly quality

This normalization method has “mixed reviews” – tends to skip whole genes

Avoiding chimeric transcripts from gene-dense genomes

- Use strand-specific RNA-Seq protocol
- Try option `-jaccard_clip` (time-consuming; no effect in case of gene-sparse genomes, maybe just check with IGV after DE calculation for important genes)



Resource requirements

How many reads are needed

- one whole lane
- pull all data together from all conditions and one individual

How long will the assembly take?

- $\frac{1}{2}$ - 1 hours per 1 million reads (does not include trimming or normalization, no jaccard clip option)

How much RAM is needed?

- 1 Gbyte per 1 million reads

How many processors?

- Most (not all) parts of Trinity are parallelized – makes sense to use most available CPUs (via `--CPU` option)
 - some programs may adjust it down (by default, Inchworm runs on at most 6 CPUs)
- Butterfly (last stage) benefits most from massive parallelization....
 -although running on too large a number of CPUs may lead to memory problems
 - controlled using `--bflyCPU` and `--bflyHeapSpaceMax` options
- Most runs should go through on BioHPC Lab medium-memory machines (cbsumm*, 128 GB of memory ad 24 CPU cores) with options

```
--CPU 20 --max_memory 100G
```

A couple of more “real” runs (skip ???)

Example 1:

Drosophila

Machine: **cbsum1c1b016** (8 CPU, 16 GB RAM – **small-memory**)

Options: **--CPU 8** (6 for Inchworm) **--max_memory 12G** **--bflyCPU 6** **--bflyHeapSpaceMax 2G**

Example 2:

Drosophila

Machine: **cbsumm02** (24 CPU, 128 GB RAM – **medium-memory**)

Options: **--CPU 20** (6 for Inchworm) **--max_memory 100G** **--bflyCPU 20** **--bflyHeapSpaceMax 4G**

Example 3:

Drosophila

Machine: **cbsumm02** (24 CPU, 128 GB RAM – **medium-memory**)

Options: **--CPU 20** (6 for Inchworm) **--max_memory 100G** **--bflyCPU 20** **--bflyHeapSpaceMax 4G** **--max_kmer_cov 2**

	Initial PE frags	PE frags after normalztn	Wall-clock times [minutes]					Total (excluding normalization)
			Normalization	Jellyfish	Inchworm	Chrysalis	Butterfly	
Example 1	40M	8.6M	100	5	30	80	523	638
Example 2	206M	19.6M	270	6	52	118	252	428
Example 3	206M	Not norm	-	13	35	6,800	396	7,244

NOTE: read normalization and trimming also take time!

Assembly Quality Assessment

Evaluation of assembly quality

<https://github.com/trinityrnaseq/trinityrnaseq/wiki/Transcriptome-Assembly-Quality-Assessment>

BUSCO Score

<https://biohpc.cornell.edu/lab/userguide.aspx?a=software&i=255#c>

C: 85% [S:34%,D:51%],

F: 14%,

M: 1%,

n: 1658

C: Complete

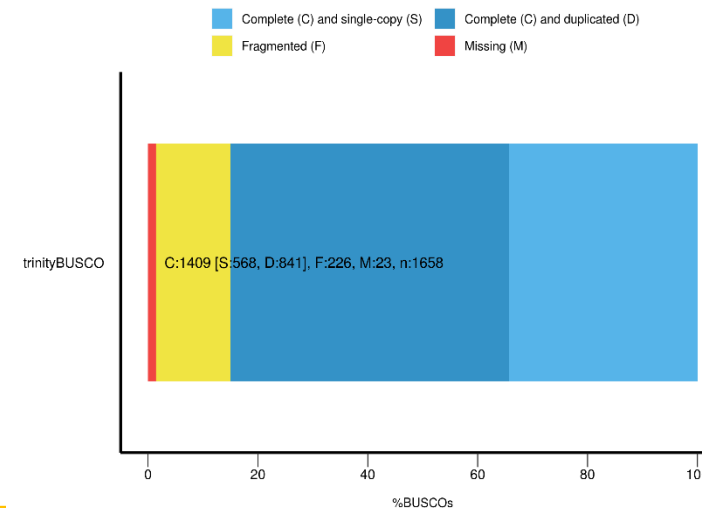
- S: single copy;
- D: duplicated;

F: Fragmented;

M: Missing;

n: Total groups;

BUSCO Assessment Results



Compare to a known species

BLAST against *D. melanogaster* protein database

```
blastx \
-query Trinity.fasta \
-db melanogaster.pep.all.fa \
-out blastx.outfmt6
-evalue 1e-20
-num_threads 6 \
```

Analysis the blast results and generate histogram

```
analyze_blastPlus_topHit_coverage.pl \
blastx.outfmt6 \
Trinity.fasta \
melanogaster.pep.all.fa \
```

Compare the *Drosophila yakuba* assembly used in the exercise to *Drosophila melanogaster* proteins from flybase

#hit_pct_cov_bin	count_in_bin	>bin_below
100	5165	5165
90	1293	6458
80	1324	7782
70	1434	9216
60	1634	10850
50	1385	12235
40	1260	13495
30	1091	14586
20	992	15578
10	507	16085

6546 *D melanogaster* proteins are represented in the assembly with >90% of full length

Other scores

<https://github.com/trinityrnaseq/trinityrnaseq/wiki/Transcriptome-Assembly-Quality-Assessment>

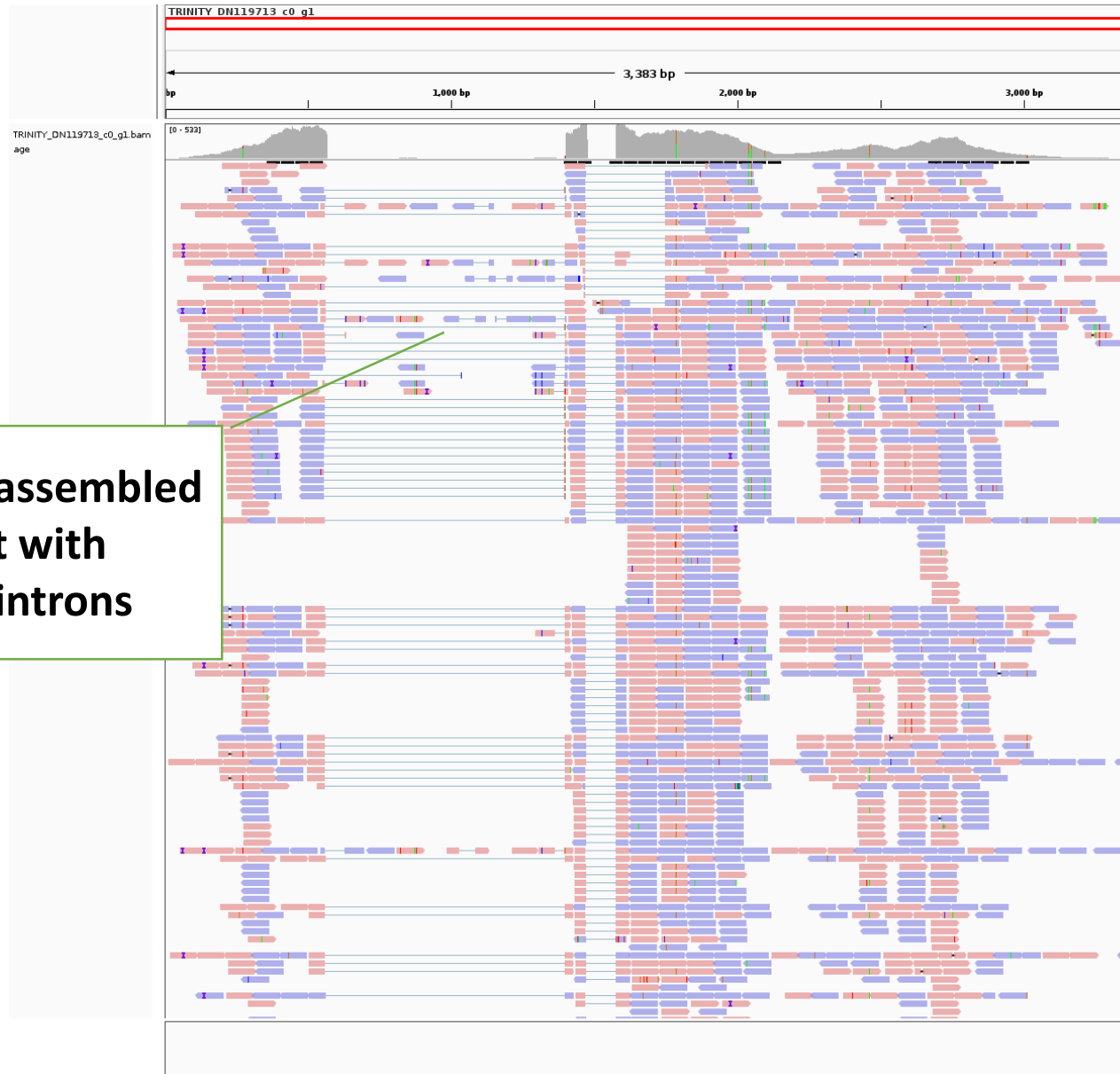
- **RNA-Seq read representation** : >80% of reads can be aligned
- **N50**: contig length at 50% of the transcriptome

Trinity assembled transcripts are not always correct

- Low expressed genes: not complete
- Highly expressed genes: pre-mRNA (with retained introns)
- Collapse of paralogous genes

Examine the assembled transcripts with IGV

Cheng Zou, Bioinformatics Facility



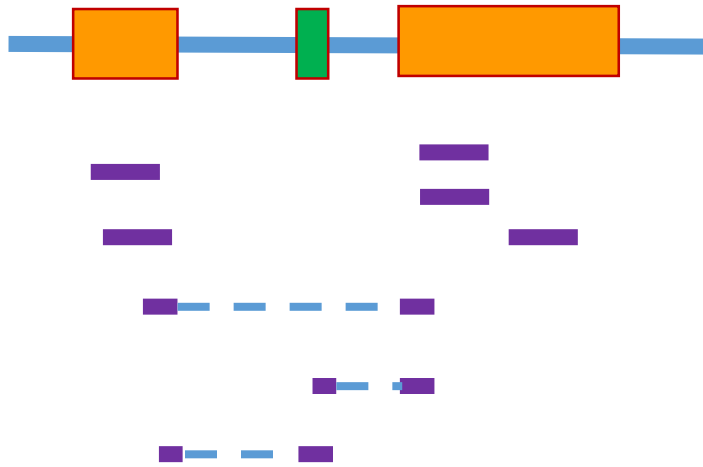
A Trinity assembled transcript with retained introns

Post Assembly Analysis

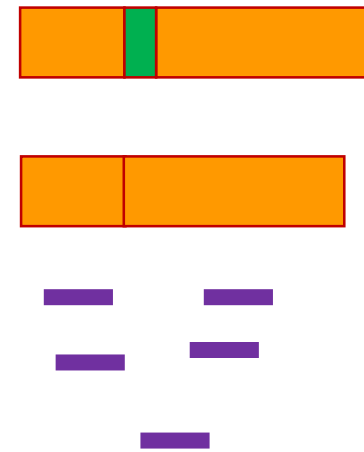
- Abundance quantification
- QC Samples and Biological Replicates
- Coding region identification
- Function annotation

Abundance estimation using RSEM

Genome as reference



Transcriptome as reference



How to distinguish **splicing isoforms**
vs **paralogs**.

Trinity transcriptome identifiers: **Genes vs. Transcripts**

```
>TRINITY_DN1000|c115_g5_i1 len=247 path=[31015:0-148 23018:149-246]  
AATCTTTTTTGGTATTGGCAGTACTGTGCTCTGGGTAGTGATTAGGGCAAAAGAAGACAC  
ACAATAAAGAACCAGGTGTTAGACGTCAGCAAGTCAAGGCCTTGGTTCTCAGCAGACAGA  
AGACAGCCCTTCTCAATCCTCATCCCTTCCCTGAACAGACATGTCTTCTGCAAGCTTCTC  
CAAGTCAGTTGTTCACAGGAACATCATCAGAATAAATTTGAAATTATGATTAGTATCTGA  
TAAAGCA
```

TRINITY_DN1000|c115_g5_i1

Gene ID

Transcript ID

RSEM for quantification

```
$TRINITY_HOME/util/align_and_estimate_abundance.pl
```

```
--gene_trans_map Gene_to_Transcript_MapFile
```

or

```
--trinity_mode
```



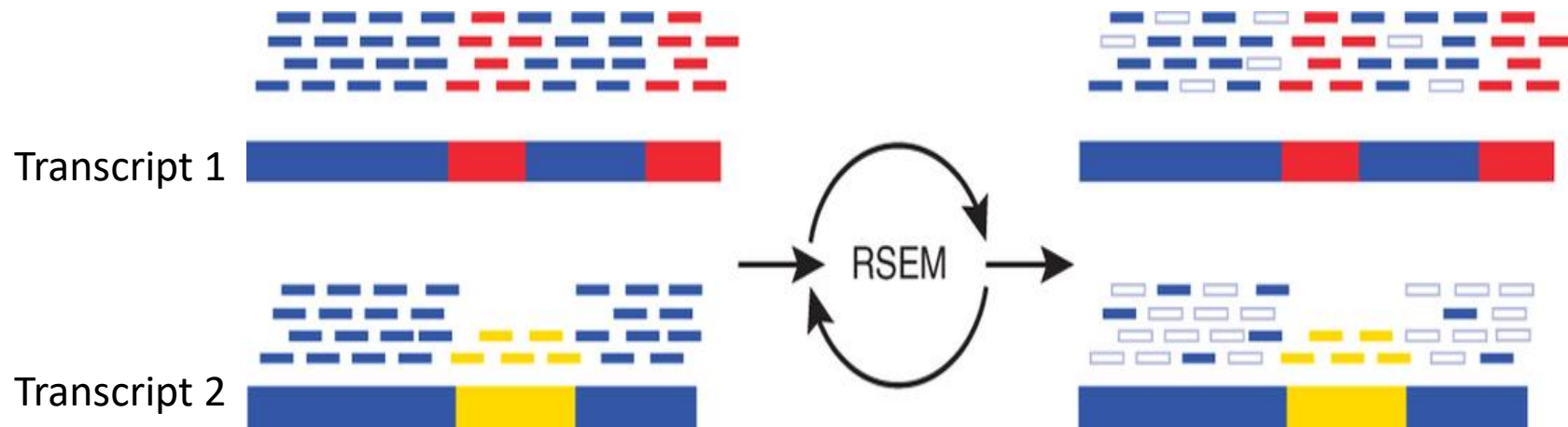
**Trinity formatted
transcript identifier**

RSEM for quantification

How RSEM deal with mapping ambiguity

Red & yellow: unique regions

Blue: regions shared between two transcripts



Parameters for `align_and_estimate_abundance.pl`

--aln_method: alignment method

Default: "bowtie" .

Alignment file from other aligner might not be supported.

--est_method: abundance estimation method

Default : RSEM, slightly more accurate.

Optional: eXpress, faster and less RAM required.

--thread_count: number of threads

--trinity_mode: the input reference is from Trinity.

Non-trinity reference requires a gene-isoform mapping file
(`--gene_trans_map`).

Output files from RSEM (two files per sample)

*.isoforms.results table

transcript_id	gene_id	length	effective_length	expected_count	TPM	FPKM	IsoPct
gene1_isoform1	gene1	2169	2004.97	22.1	3.63	3.93	92.08
gene1_isoform2	gene1	2170	2005.97	1.9	0.31	0.34	7.92
...							

*.genes.results table

gene_id	transcript_id(s)	length	effective_length	expected_count	TPM	FPKM
gene1	gene1_isoform1 ,gene1_isoform 2	2169.1	2005.04	24	3.94	4.27
...						

Output files from RSEM (two files per sample)

*.isoforms.results table

transcript_id	gene_id	length	effective_length	expected_count	TPM	FPKM	IsoPct
gene1_isoform1	gene1	2169	2004.97	22.1	3.63	3.93	92.08
gene1_isoform2	gene1	2170	2005.97	1.9	0.31	0.34	7.92
...							

Percentage of an isoform in a gene

*.genes.results table

gene_id	transcript_id(s)	length	effective_length	expected_count	TPM	FPKM
gene1	gene1_isoform1 ,gene1_isoform2	2169.1	2005.04	24	3.94	4.27
...						

sum

Filtering transcriptome reference based on RSEM

filter_fasta_by_rsem_values.pl

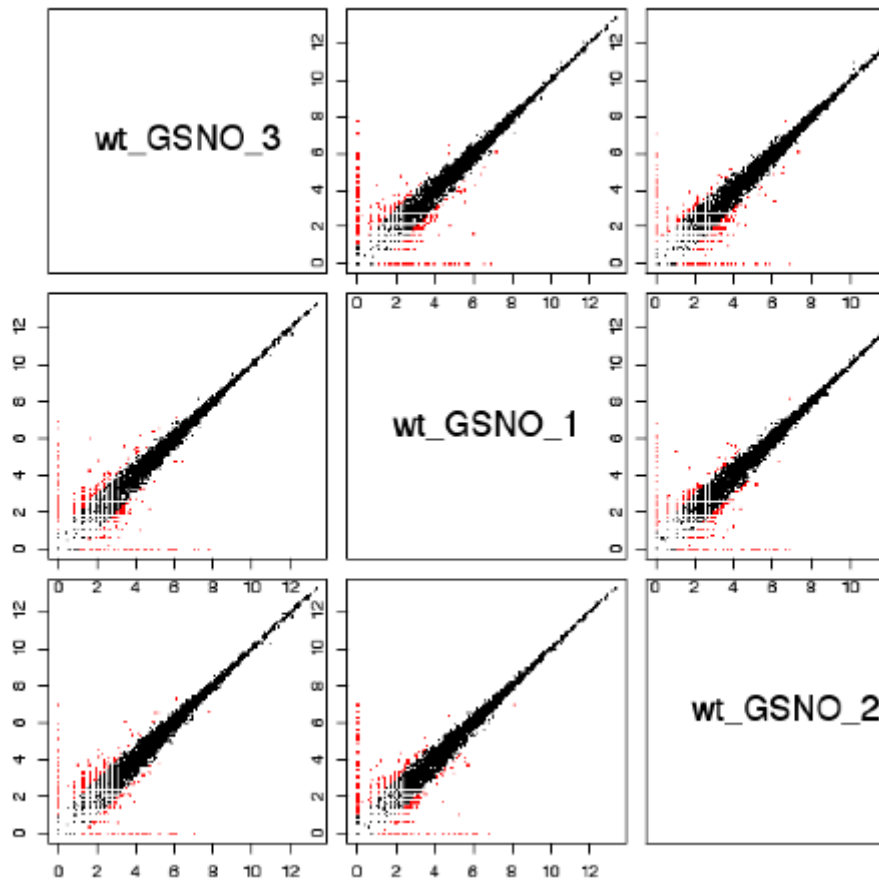
```
filter_fasta_by_rsem_values.pl \
--rsem_output=s1.isoforms.results,s2.isoforms.results \
--fasta=Trinity.fasta \
--output=Trinity.filtered.fasta \
--isopct_cutoff=5 \
--fpkm_cutoff=10 \
--tpm_cutoff=10 \
```

- Can be filtered by multiple RSEM files simultaneously. (criteria met in any one filter will be filtered)

Differentially Expressed Gene Identification

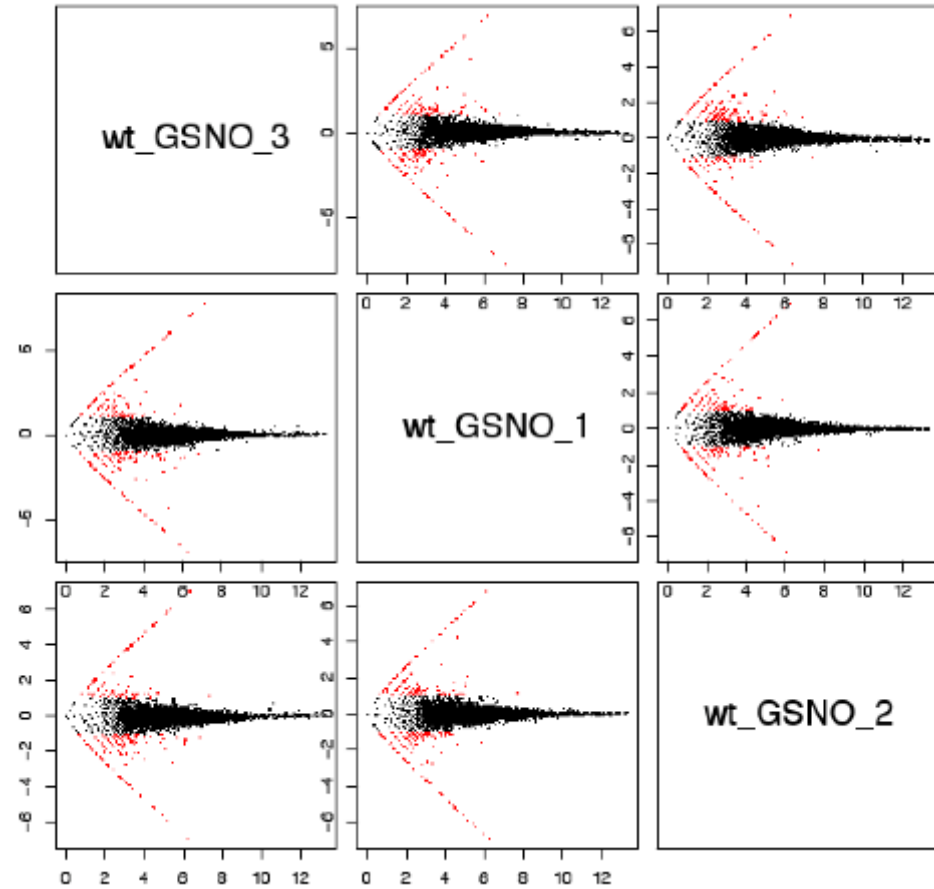
Covered in the next lecture

Sample QC



Dot plots

x-axis: $\log(\text{CPM})$ sample 1
y-axis: $\log(\text{CPM})$ sample 2

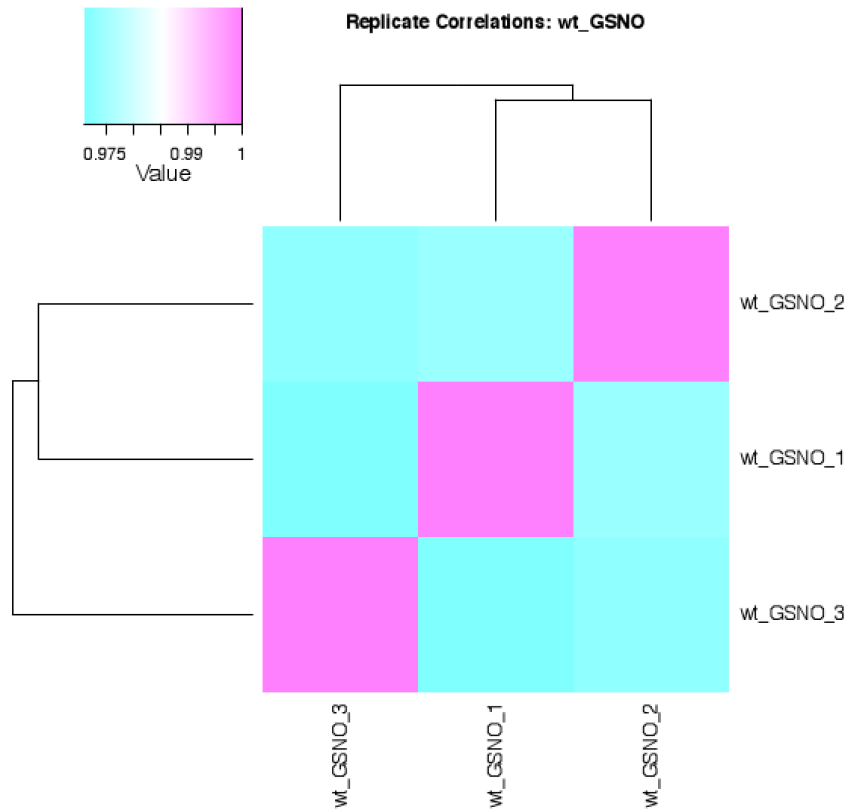


MA plots

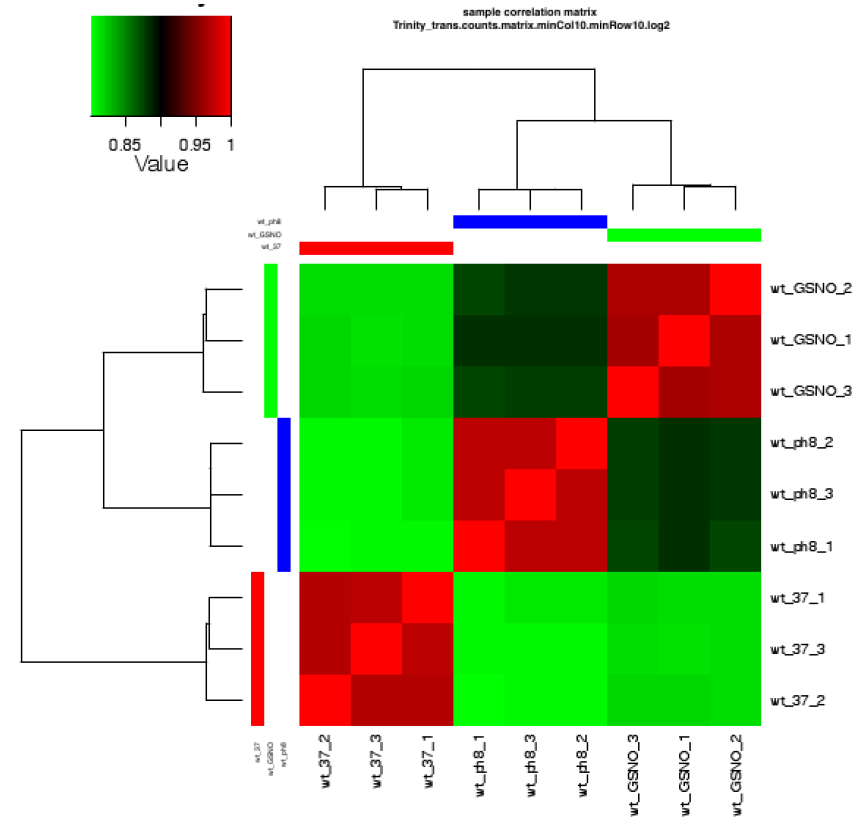
x-axis: mean $\log(\text{CPM})$
y-axis: $\log(\text{fold_change})$

Sample QC

Compare within replicate group
(3 replicate samples)

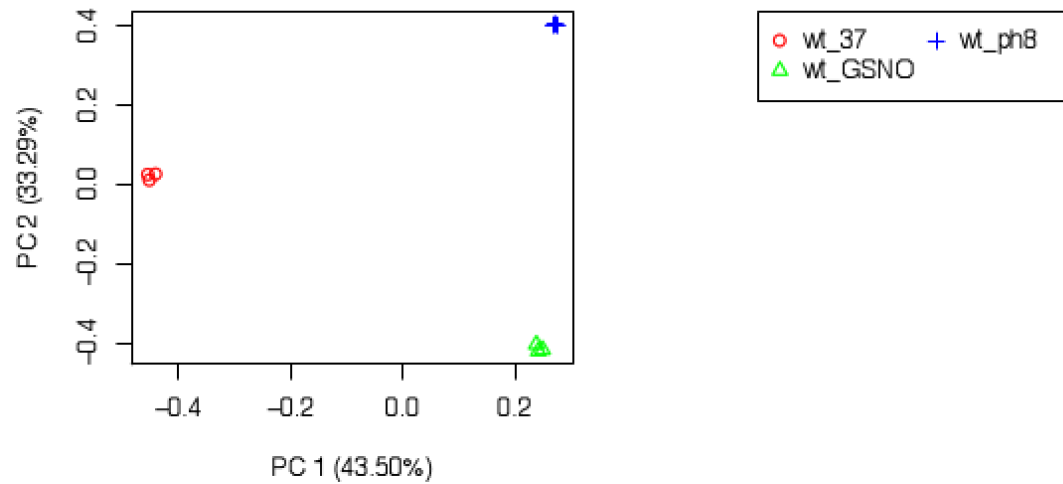


Compare between replicates group
(3 conditions, 3 replicates per condition)

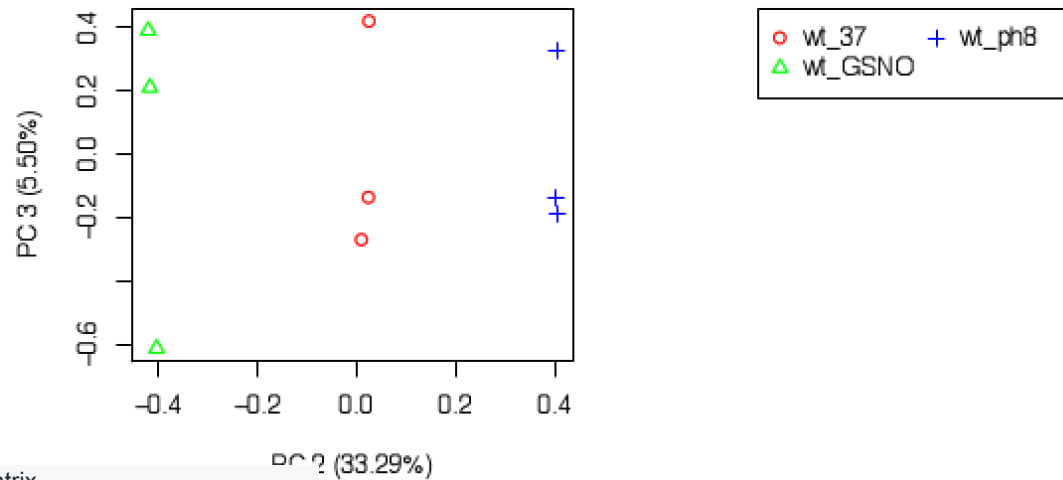


```
$TRINITY_HOME/Analysis/DifferentialExpression/PtR \ --matrix  
Trinity_trans.counts.matrix \ --min_rowSums 10 \ -s samples.txt --log2 --  
CPM --sample_cor_matrix
```


Sample QC

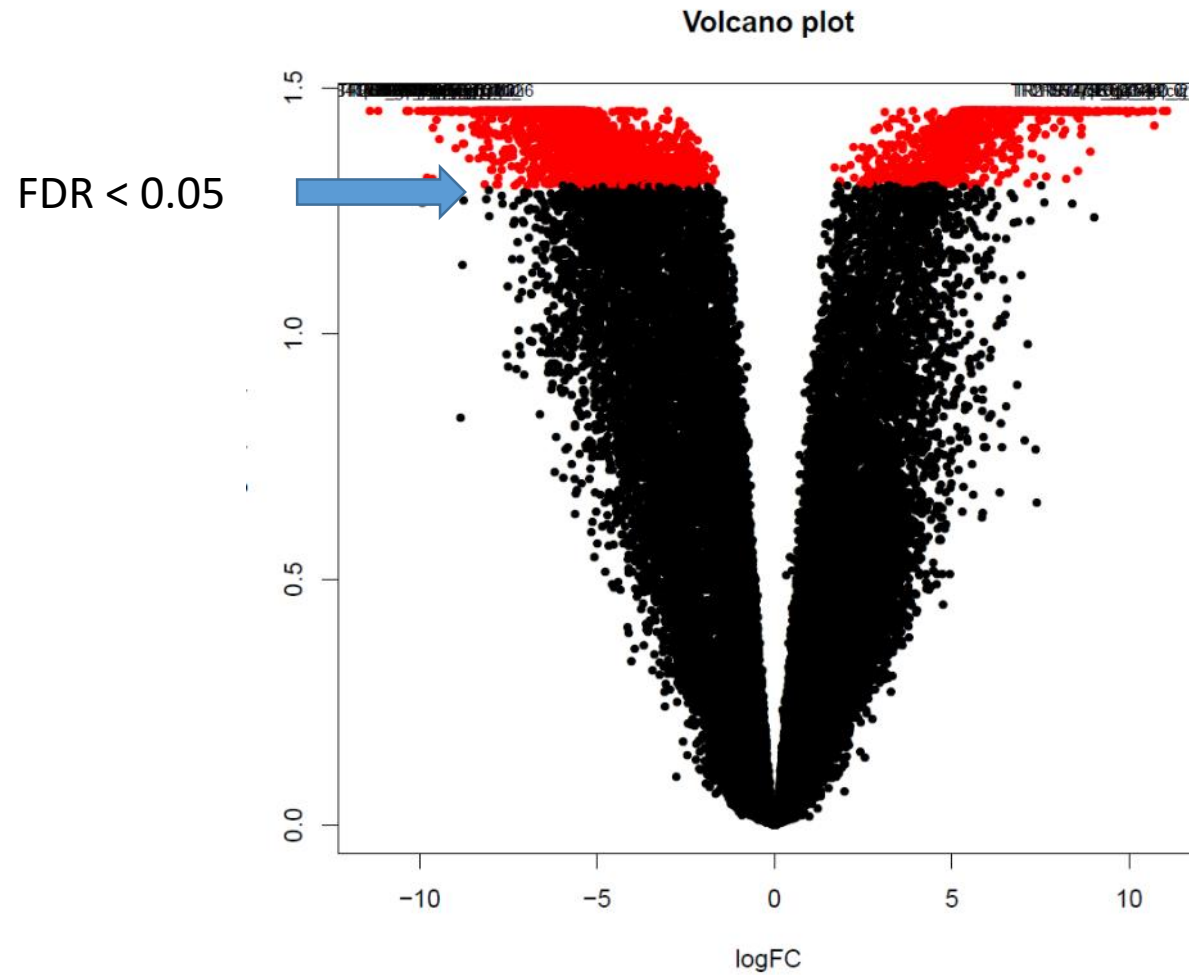


PCA Plots



```
$TRINITY_HOME/Analysis/DifferentialExpression/PtR \ --matrix  
Trinity_trans.counts.matrix \ -s samples.txt --min_rowSums 10 --log2 \ --CPM --  
center_rows \ --prin_comp 3
```

Volcano Plot
X axis: $\log(\text{Fold_Change})$
Y axis: $-\log(\text{FDR})$



Gene Function Annotation

1. Two ways converting DNA to protein sequences

- 6-frame translation
- Predicted ORF with Transdecoder (recommended, it saves computing time)

2. Gene Ontology annotation

- BLAST2GO (recommended, available on BioHPC)
- TRINOTATE
- INTERPROSCAN

Transdecoder – Identify ORF from Transcripts

#identify longest ORF

```
TransDecoder.LongOrfs -t myassembly.fa
```

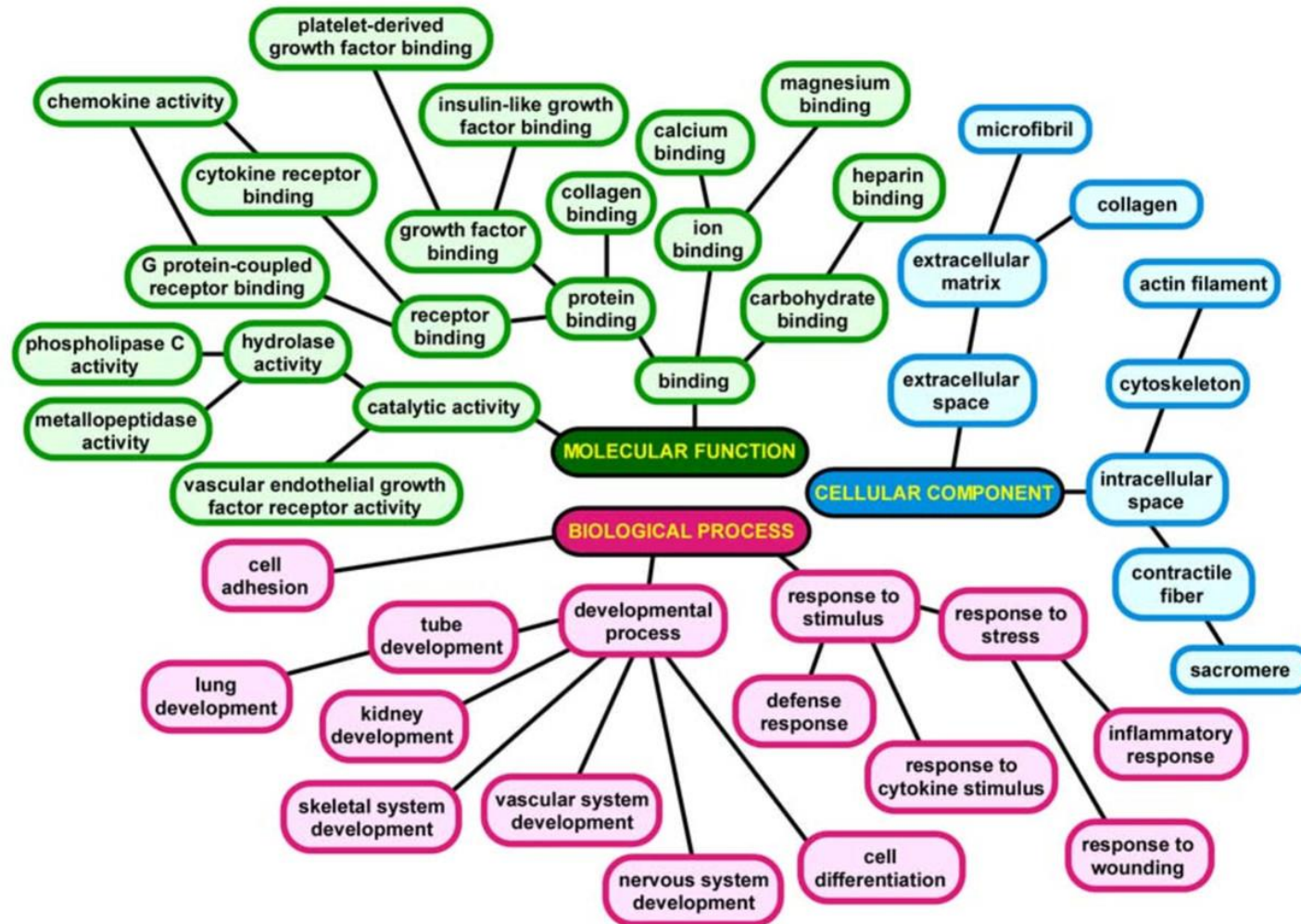
#BLAST ORF against known protein database

```
blastp -query myassembly.fa.transdecoder_dir/longest_orfs.pep \  
-db swissprot \  
-max_target_seqs 1 \  
-outfmt 6 -evalue 1e-5 \  
-num_threads 8 > blastp.outfmt6
```

#Predict ORF based on a Markov model of coding sequence (trained from long ORF and known proteins)

```
TransDecoder.Predict --cpu 8 \  
--retain_blastp_hits blastp.outfmt6 \  
-t myassembly.fa
```

Hierarchical structure of gene ontology?



Function Annotation Results

gene1	molecular_function	GO:0008270	zinc ion binding
gene1	molecular_function	GO:0046872	metal ion binding
gene1	cellular_component	GO:0005622	intracellular
gene1	cellular_component	GO:0019005	SCF ubiquitin ligase complex
gene1	biological_process	GO:0009733	response to auxin
gene2	molecular_function	GO:0003677	DNA binding
gene2	cellular_component	GO:0005634	nucleus
gene2	cellular_component	GO:0005694	chromosome
gene2	biological_process	GO:0006259	DNA metabolic process
gene2	biological_process	GO:0034641	cellular nitrogen compound metabolic process

For more information visit Trinity site:

<https://github.com/trinityrnaseq/trinityrnaseq/wiki>

For exercise-related questions contact

bukowski@cornell.edu, qisun@cornell.edu