De novo transcriptome assembly using Trinity

Marina Linardić, PhD
Department of Molecular, Cell and Developmental Biology
UCLA

linardicm@ucla.edu

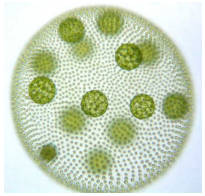
Part of presentation adapted from Brian Haas's Trinity presentations

My research – brown algal evo-devo

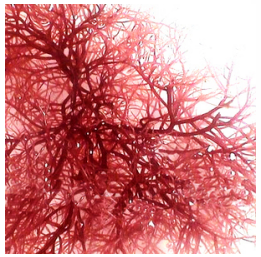
Arabidopsis

Chara

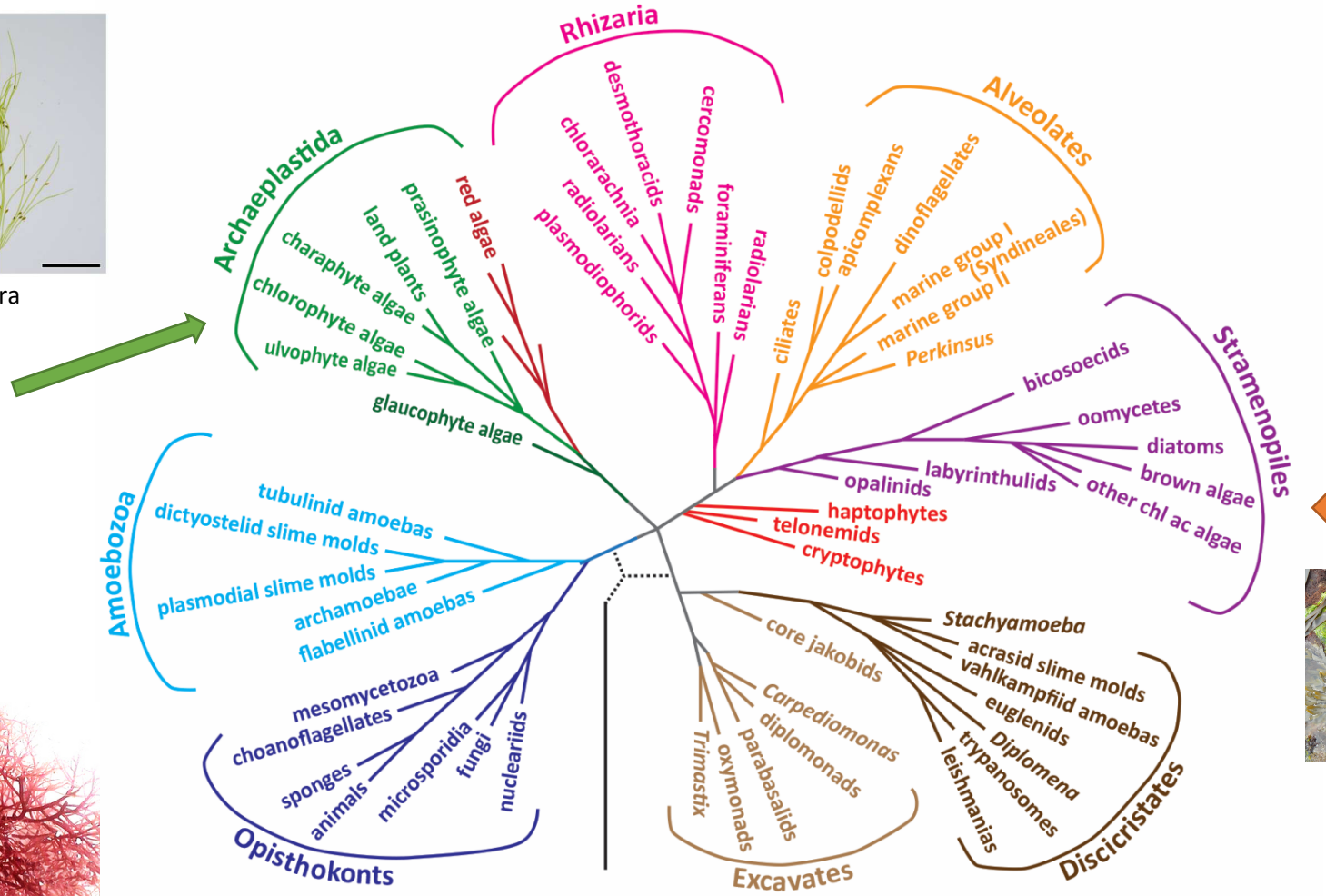
Ulva

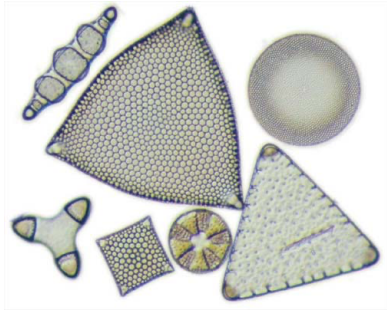


Volvox



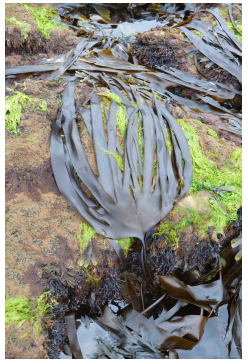
Gracilaria





diatoms

Fucus



Laminaria

Sargassum

Adapted from Baldauf (2003)

My research – brown algal evo-devo

- Understanding developmental mechanisms of growth regulated by cell walls
- Identifying molecular processes involved in abiotic stress tolerance

Very few genomes sequenced!

Fucus

Laminaria

Sargassum

This workshop

Day 1

1. Introduction to Trinity

2. Hands-on practice:

- Preparing RNA-seq reads
- Running Trinity assembly

Day 2

1. Introduction to post-assembly analysis

2. Hands-on practice:

- Assembly quality assessment
- Abundance estimation
- Differential expression analysis
- Coding region identification
- Functional annotation

Learning goals

- Principles of de novo transcriptome assembly
- Understanding basic pipeline to run Trinity and analyze RNA-seq data
- Troubleshooting errors

Running Trinity on your samples from start to (a) finish

What I expect

- You have basic understanding of Unix (e.g. you took the Intro to Unix workshop)
- You understand the basics of RNA-seq analysis
- You have an account on Hoffman2

What I don’t expect

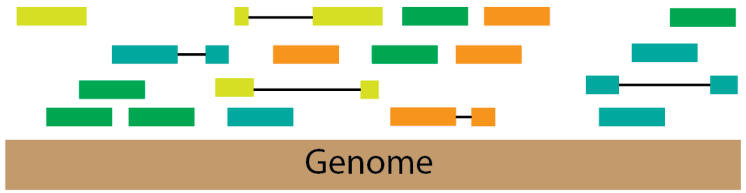
- You haven’t done any RNA-seq bioinformatic analysis on a cluster (e.g. Hoffman2)

Transcriptome assembly using RNA-seq data

RNA-seq reads

Read cleanup

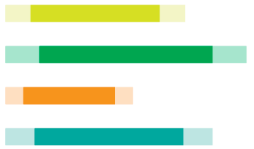
Reference-based



Alignment to genome
e.g. STAR
TopHat

Transcript reconstruction
e.g. Cufflinks

De novo



De novo assembly
e.g. Trinity
Velvet/Oases
SOAPdenovo-Trans
TransABYSS

Post assembly analysis
QC assessment
Transcript analysis
Abundance estimation
Differential Expression analysis
Protein coding region identification
Functional annotation

Trinity package

developed at Broad Institute and Hebrew University of Jerusalem

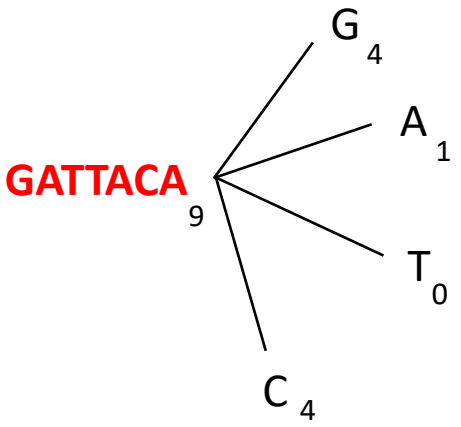
Assembler + Post-assembly analysis

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

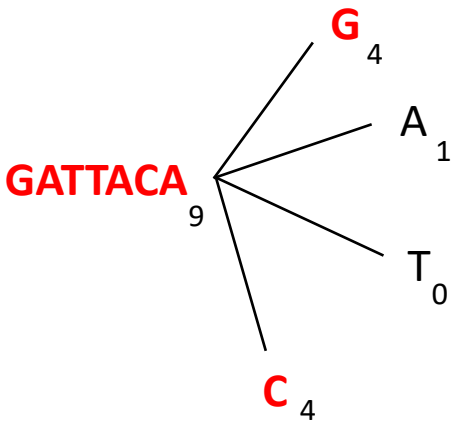
Grabherr MG, et al. Full-length transcriptome assembly from RNA-seq data without a reference genome. Nat Biotechnol. 2011 May 15;29(7):644-52. doi: 10.1038/nbt.1883.

Inchworm

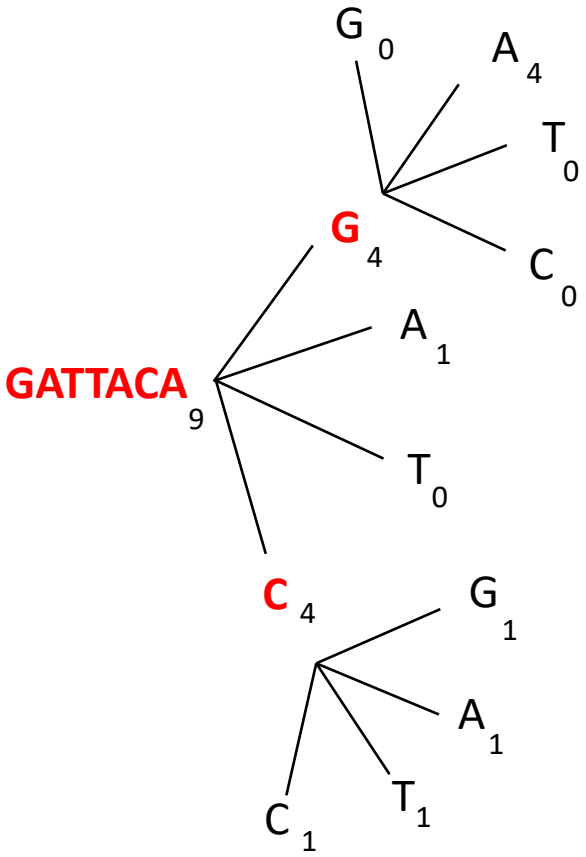
1. Decompose all reads into overlapping Kmers (25-mers)
2. Identify seed kmer as most abundant Kmer, ignoring low-complexity kmers
3. Extend kmer at 3’ end, guided by coverage



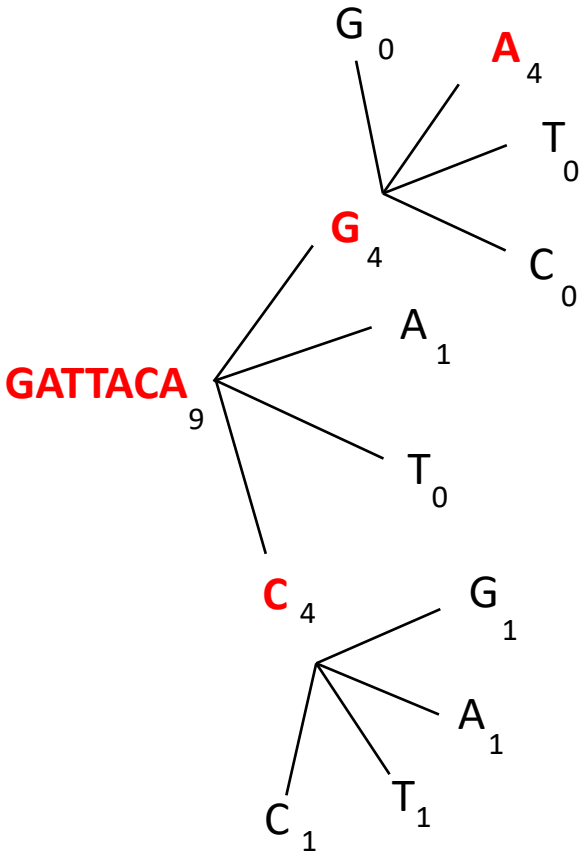
Inchworm



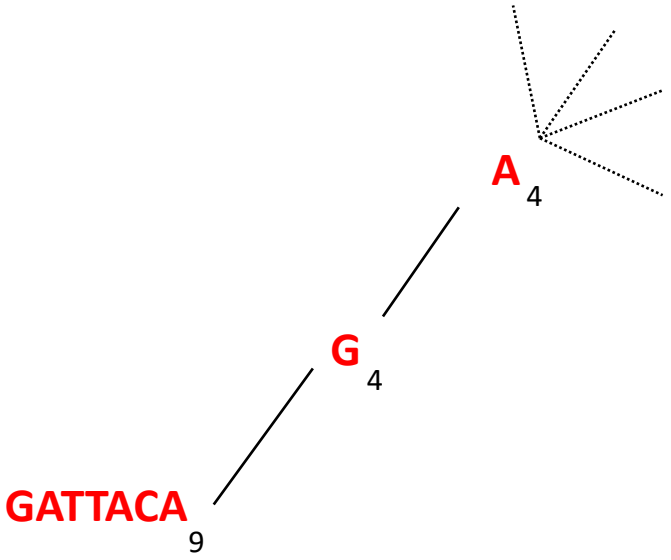
Inchworm



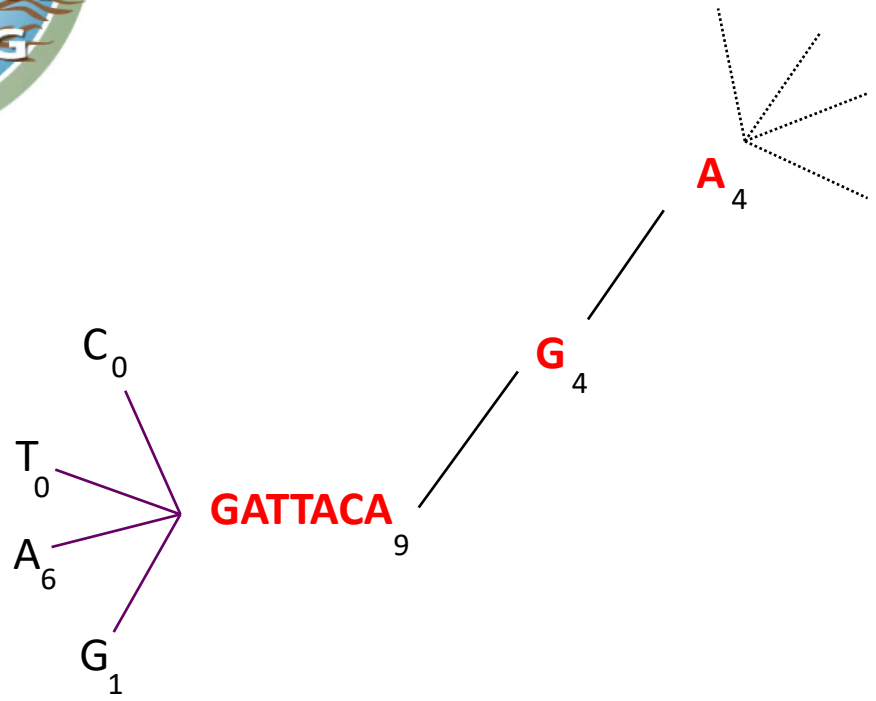
Inchworm



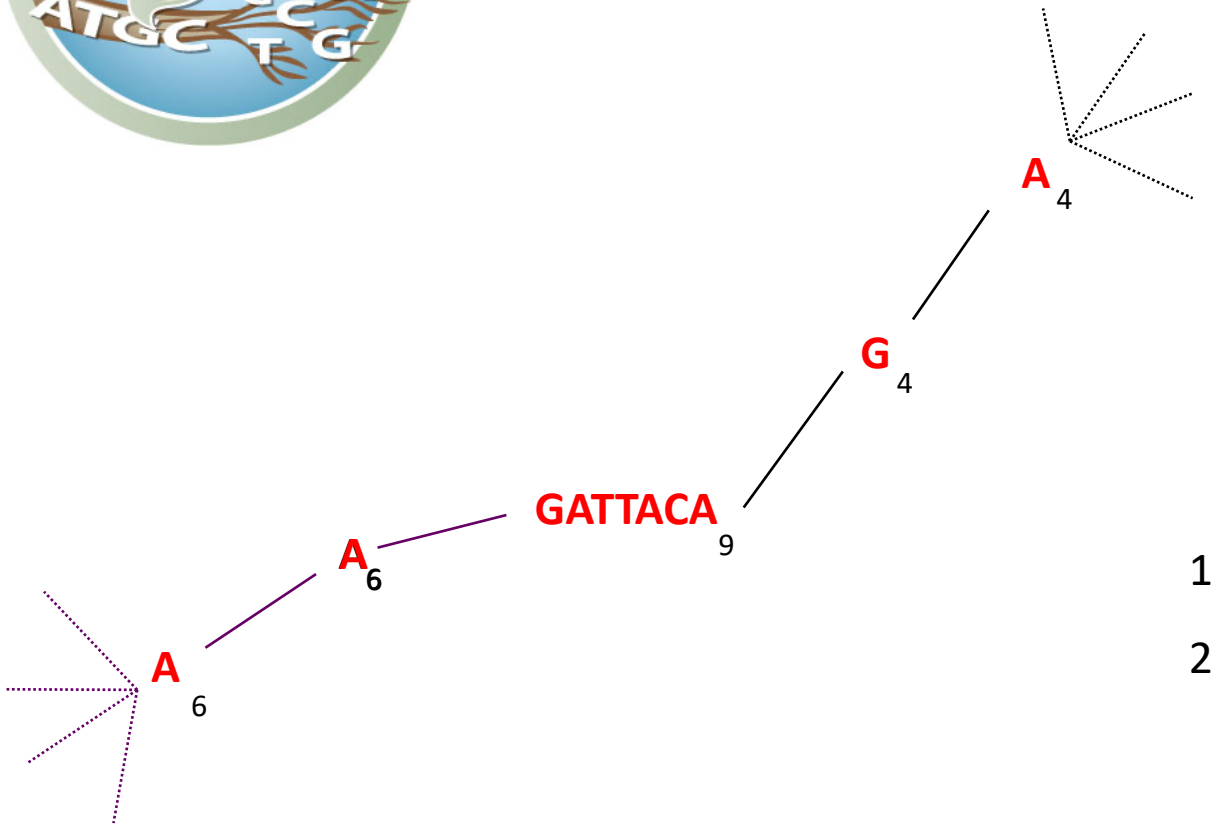
Inchworm



Inchworm



Inchworm



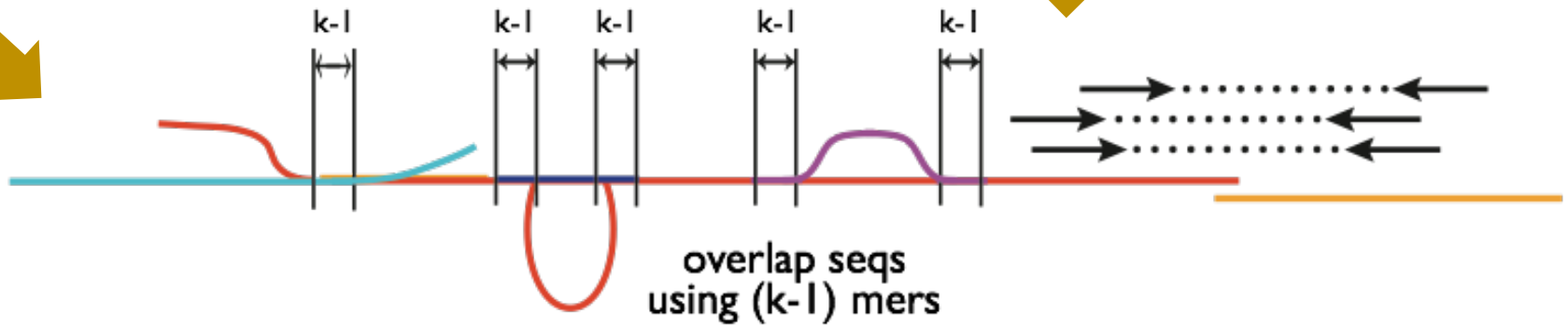
1. Report contig: ……**AAGATTACAGA**…….
2. Remove assembled kmers from catalog, then repeat the entire process

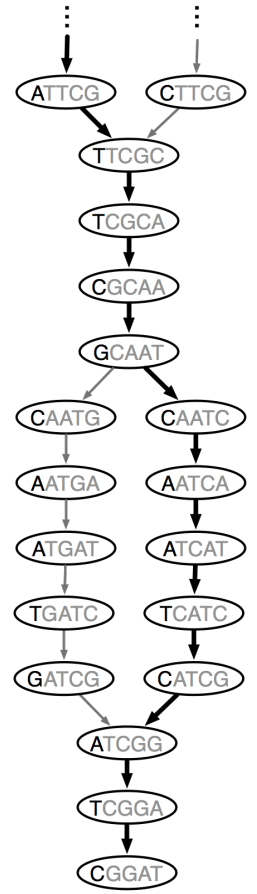
Chrysalis

Integrate isoforms via k-1 overlaps

Chrysalis

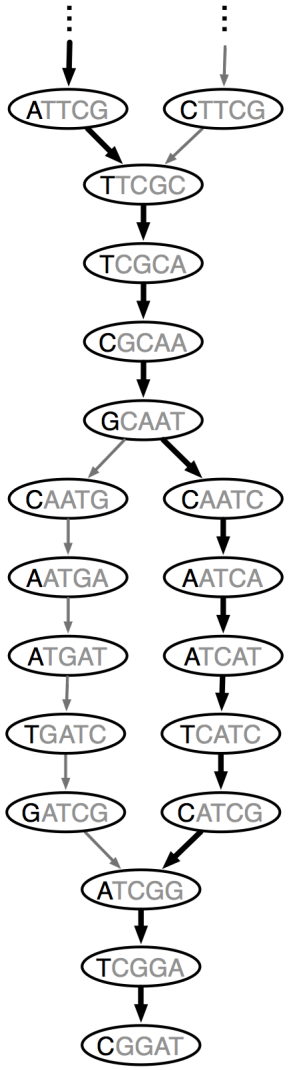
- Collect all contigs that share k-1-mers
- Build graph (disjoint “components”)
- Map reads to components



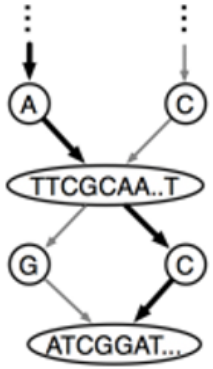


Build de Bruijn Graphs (ideally, one per gene)

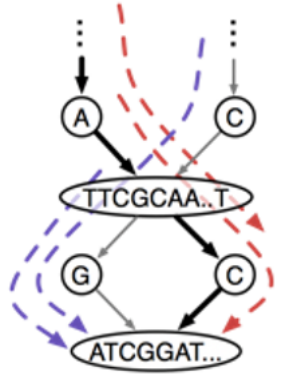
Butterfly



compacting



finding paths



extracting sequences

..CTTCGCAA..TGATCGGAT...

..ATTCGCAA..TCATCGGAT...

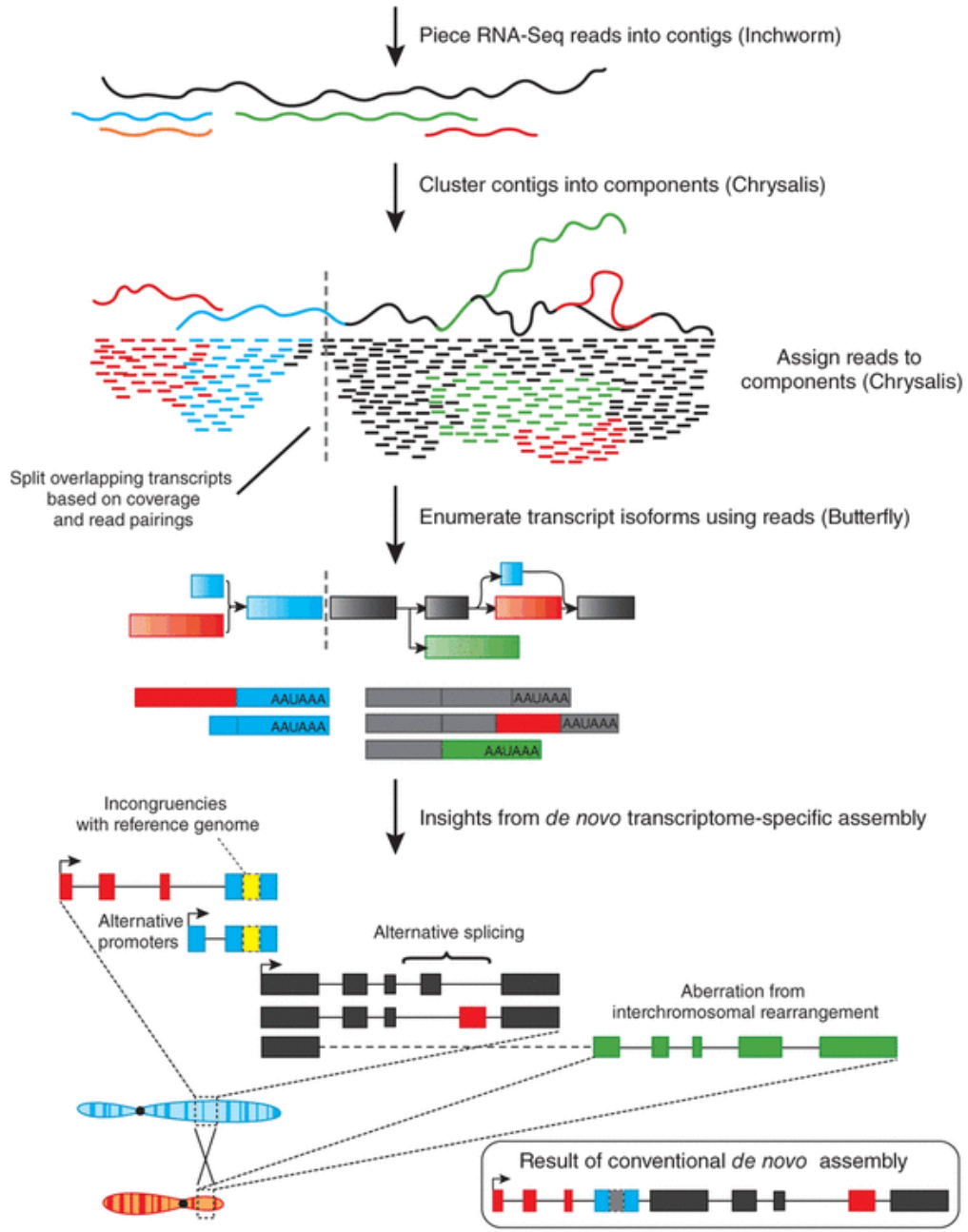
De Brujin graph

compact graph

compact graph with reads

sequence

Summary



*Iyer MK, Chinnaiyan AM (2011) Nature Biotechnology **29**, 599–600*

Trinity assembly pipeline

1) Input

Cleaned RNA-seq reads
FASTA or FASTQ format

⇩

2) Run Trinity

⇩

3) Output

Trinity.fasta
FASTA file with assembled transcripts

⇨ **Transcriptome assembly quality assessment**

Trinity programs

- Trinity perl script – Inchworm + Chrysalis + Butterfly
- Various perl analysis scripts

- Third-party software (e.g. Trimmomatic, Jellyfish)

- External software Trinity depends on (needs to be in the search PATH):
  - samtools
  - bowtie2
  - RSEM, eXpress: alignment-based abundance estimation (Bo Li and Colin Dewey)
  - kallisto, salmon: alignment-free abundance estimation
  - Transcoder: identify candidate coding regions in within transcripts

Running Trinity

A Trinity command can be created in a **bash script (vi/vim/nano)**

```
#!/bin/bash

$TRINITY_DIR/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 20G \
--CPU 2 \
--output /results/trinity_out
```

Dissecting the command line

```
$TRINITY_DIR/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 /results/trinity_out
```

Dissecting the command line

```
$TRINITY_DIR/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 /results/trinity_out
```

Paired-end reads specified with **--left** and **--right**. If only single-end, use **--single** instead.

- You can concatenate all the right and left reads to have only one input per paired end:

```
cat sample1_right.fastq sample2_right.fastq > all_right.fastq
cat sample1_left.fastq sample2_left.fastq > all_left.fastq
```

- Use all reads from an individual (all conditions) to capture most genes

Dissecting the command line

```
$TRINITY_DIR/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 /results/trinity_out
```

The PE fragments are strand-specific, with left end on the **R**everse strand and the right end on
Forward strand of the sequenced mRNA template

For non-strand specific reads, skip the option **--SS_lib_type**

Dissecting the command line

```
$TRINITY_DIR/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 /results/trinity_out
```

e.g.
- 2G is the maximum memory to be used at any stage which allows memory limitation
- at most 2 CPU cores will be used in any stage

Dissecting the command line

```
$TRINITY_DIR/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 /results/trinity_out
```

- Specify path to your output folder-
- If you don't, it will automatically build a *trinity_outdir* folder into your current working directory

More commands can be found here:
https://github.com/trinityrnaseq/trinityrnaseq/wiki/Running-Trinity

Some issues to consider with de novo transcriptome assembly of RNA-Seq data

- RNA-Seq reads often need pre-processing before assembly
  - barcodes and Illumina adapters need to be removed
  - low base quality reads (or part of reads) need to be trimmed off
  - remove contamination with other species

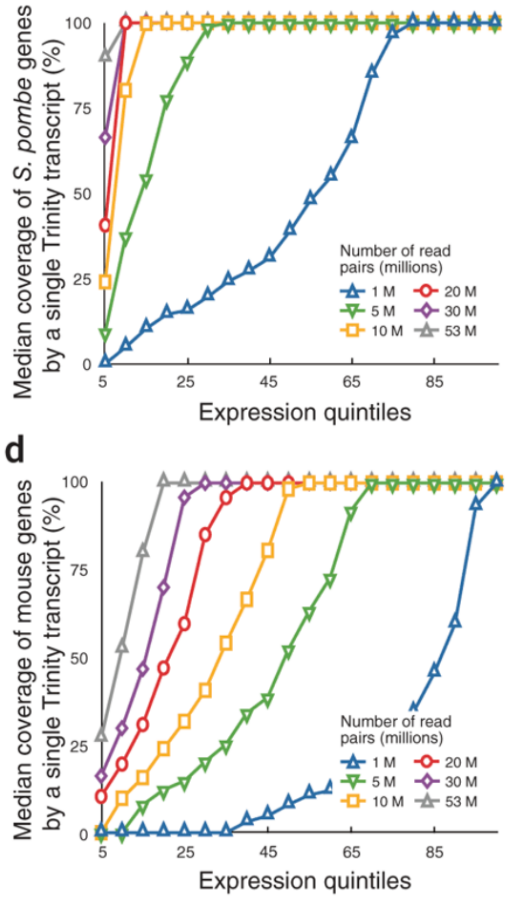
- Trimmomatic included in the Trinity package
- Need to include `--trimmomatic` option with desired requirements

- Assembly run time
  - Read normalization

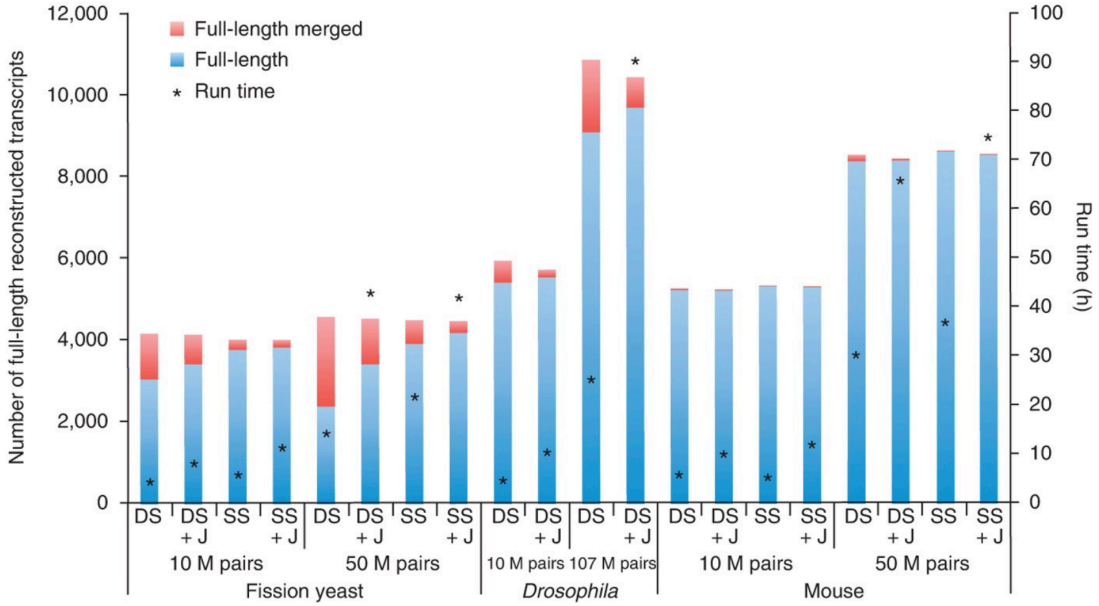
- in silico normalization in Trinity happens by default

System requirements

How many reads are needed?



How long will it take?



How much memory do we need?

- Memory needed:

1GB of RAM/1 million reads

- Timing:

~1 hours/1 million reads

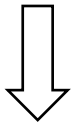
N. G Grabherr et. al., Nature Biotechnology 29, 644–652 (2011)

B. J. Haas et. al., Nature Protocols 8, 1494–1512 (2013)

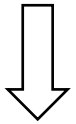
Trinity assembly pipeline

1) Input

Cleaned RNA-seq reads
FASTA or FASTQ format



2) Run Trinity



3) Output

Trinity.fasta
FASTA file with assembled transcripts

Let’s practice!

Log into Hoffman2

1. Connect to Hoffman2 (on Terminal; Linux and Mac)

```
$ ssh username@hoffman2.
idre.ucla.edu
```

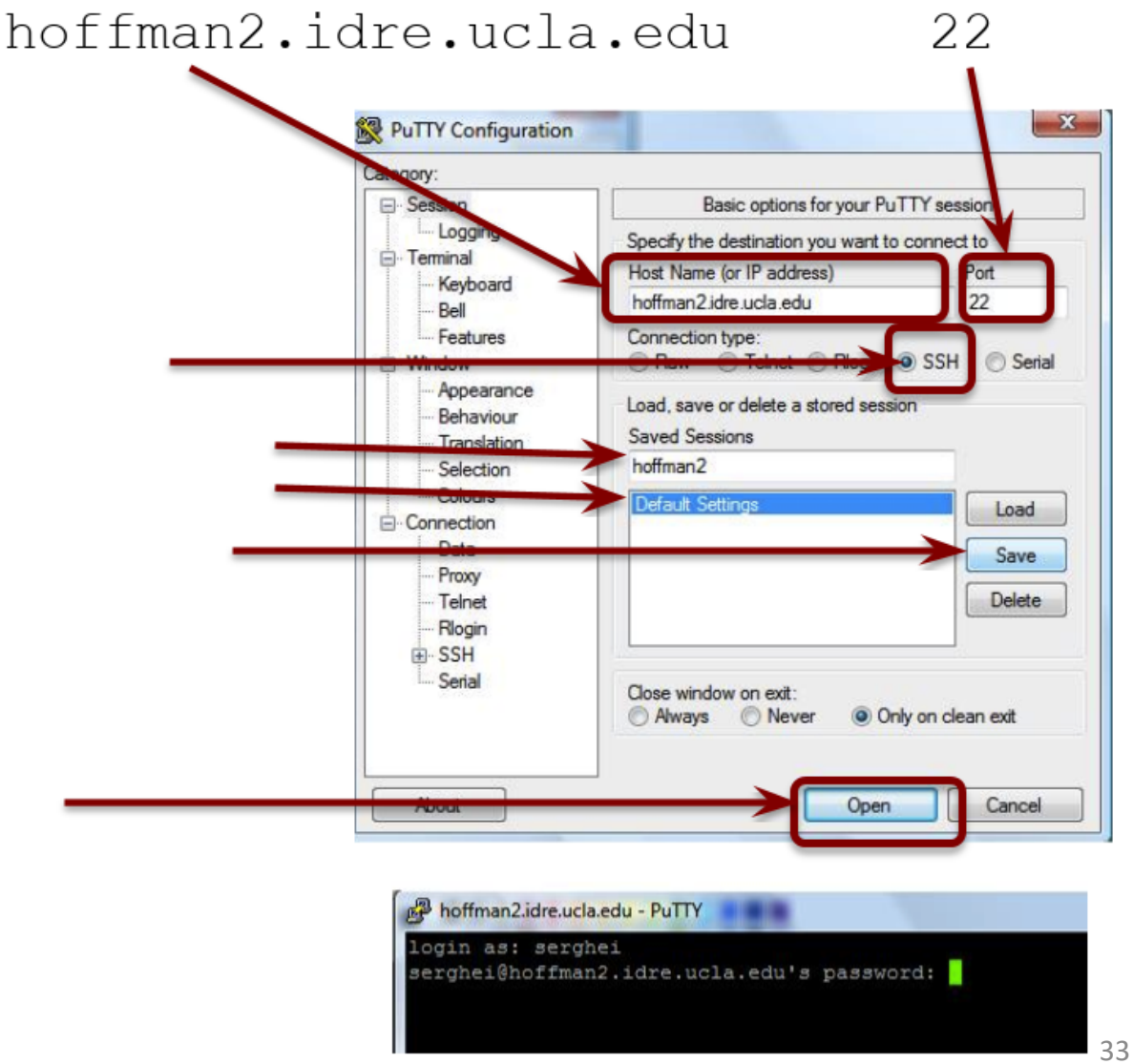
2. Input password (! You will not see the password as you are writing it down so be careful)

```
Marinas-MacBook-Pro:~ marina$ ssh braybroo@hoffman2.idre.ucla.edu
braybroo@hoffman2.idre.ucla.edu's password:
Last login: Wed Aug 12 04:42:23 2020 from 193.52.39.1
Welcome to the Hoffman2 Cluster!

Hoffman2 Home Page:     http://www.hoffman2.idre.ucla.edu
Consulting:             https://support.idre.ucla.edu/helpdesk

All login nodes should be accessed via "hoffman2.idre.ucla.edu".

Please do NOT compute on the login nodes.

Processes running on the login nodes which seriously degrade others'
use of the system may be terminated without warning.  Use qrsh to obtain
an interactive shell on a compute node for CPU or I/O intensive tasks.

The following news items are currently posted:

   Mathematica version 12.1
   MATLAB version 9.8 (R2020a) Total Academic Headcount
   Q Chem version 5.3.0
   IDRE Workshops and Training Sessions

Enter shownews to read the full text of a news item.
[braybroo@login4 ~]$
```

Log into Hoffman2

- If you have Windows PC, you need to download PuTTY: https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

- Setting up SSH on PuTTY



Request an interactive node - qrsh

```
Welcome to the Hoffman2 Cluster!

Hoffman2 Home Page:     http://www.hoffman2.idre.ucla.edu
Consulting:             https://support.idre.ucla.edu/helpdesk

All login nodes should be accessed via "hoffman2.idre.ucla.edu".

Please do NOT compute on the login nodes.

Processes running on the login nodes which seriously degrade others'
use of the system may be terminated without warning.  Use qrsh to obtain
an interactive shell on a compute node for CPU or I/O intensive tasks.

The following news items are currently posted:

   Mathematica version 12.1
   MATLAB version 9.8 (R2020a) Total Academic Headcount
   Q Chem version 5.3.0
   IDRE Workshops and Training Sessions

Enter shownews to read the full text of a news item.
[braybroo@login3 ~]$ qrsh
Last login: Mon May  4 06:50:03 2020 from login3
[braybroo@n2238 ~]$
```

Let’s put our data in the working directory

Cyberduck (Windows and Mac)

- Open the Hoffman2 connection and drag and drop the workshop folder into your home directory

Terminal (Mac and Linux)

```
scp -r ./RNAseqIII_workshop username@hoffman2.idre.ucla.edu:path/to/directory
```

or

```
git clone
```

1) Input – RNA-seq files

- Paired-end sequencing
- Samples:

```
0h_1_R1.fastq
0h_1_R2.fastq
0h_2_R1.fastq
0h_2_R2.fastq
0h_3_R1.fastq
0h_3_R2.fastq
```

```
6h_1_R1.fastq
6h_1_R2.fastq
6h_2_R1.fastq
6h_2_R2.fastq
6h_3_R1.fastq
6h_3_R2.fastq
```

- FASTQ files:

```
$ head 0h_1_R1.fastq
$ head 0h_1_R2.fastq
```

```
@K00208:YAP074:4:1101:71.19:54.26#0/1
GTTGGAGGAGATTTCGGTGCCGTTCGAGTCCTCGATGTCCACGACCGTTCTCCCCACGTGGTTCACCTCATTGTTCCTAAATATGTTGTTTTCGTTCACAC
+
`[`eeeieiiiee`i`ieV[eVVVeei`LL``ieiV[ee[eeiii`VVLL[VLV`iiVV`[LLVV`LVeeLLLLLLVLL[`eeieLVeLVLLLLLLLLLV
```

read1

```
@K00208:YAP074:4:1101:71.19:54.26#0/2
NTCATTCCAGGATCGACAAACAACGTGCTCGAACAGAACGTGTTGAAAGCTAACCGTTACAGCAACATCTACGCCTACTATGGTCCCCAAGAGCAAATCCC
+
@`[L[[eLeeLVLLL```LeiiieLLLeLLVLeee[`LVV`LL`[[LLLLVeei[eV[LLLLLLLL[LVLV[[``VLVLVVLLLVVHLHHVVVLVLLLLLV
```

read2

2) Run Trinity

- Trinity can be downloaded from https://github.com/trinityrnaseq/trinityrnaseq/releases
- Steps:
  1. Use wget to download the Trinity software tar.gz file

```
wget https://github.com/trinityrnaseq/trinityrnaseq/releases/download/v2.10.0/trinityrnaseq-v2.10.0.FULL.tar.gz
```

  2. Unzip the tar.gz

```
tar -xvzf trinityrnaseq-v2.10.0.FULL.tar.gz
```

  3. Compile the software

```
module load cmake/3.16.14
make
make plugins
```

- Trinity commands can be abbreviated using the variable TRINITY_HOME to denote the location of the Trinity package

```
export TRINITY_HOME=/Path/To/Folder/trinityrnaseq-Trinity-v2.10.0
```

This means that when you type the command $TRINITY_HOME/Trinity it actually refers to /Path/To/Folder/trinityrnaseq-Trinity-v2.10.0/Trinity

2) Run Trinity

Next:
- concatenate reads

```
cat s1_R1.fastq s2_R1.fastq s2_R1.fastq s2_R1.fastq > all_R1.fastq
cat s1_R2.fastq s2_R2.fastq s2_R2.fastq s2_R2.fastq > all_R2.fastq
```

2) Run Trinity

```
#!/bin/bash

$TRINITY_DIR/Trinity
--seqType fq \
--left all_R1.fastq\
--right all_R2.fastq\
--SS_lib_type RF \
--max_memory 20G \
--CPU 2 \
--output /results/trinity_out
```

```
Path_to/Trinity --seqType fq --left all_R1.fastq --right all_R2.fastq --max_memory 2G
```

2) Run Trinity

```
#!/bin/bash

$TRINITY_DIR/Tr
--seqType fq \
--left all_R1.fast
--right all_R2.fas
--SS_lib_type RF
--max_memory
--CPU 2 \
--output /results
```

ERROR!

We need to load in additional software required by Trinity

```
Path_to/Trinit                                                    tq –max_memory 2G
```

2) Run Trinity

```
PATH=$PATH:/path/to/salmon/
module load samtools/1.9
module load bowtie2
module load jellyfish/2.2.10
module load python/3.7.0
```

Found on hoffman

- To look for all available software:

`module avail` or `module available`

- To load in a package

`module load`

We need to download it from the website https://github.com/COMBINE-lab/salmon/releases

Using wget

```
wget https://github.com/COMBINE-lab/salmon/releases/download/v1.3.0/salmon-1.3.0_linux_x86_64.tar.gz
```

```
tar -xvzf salmon-1.3.0_linux_x86_64.tar.gz
```

2) Run Trinity

```
Path_to/Trinity --seqType fq --left all_R1.fastq --right all_R2.fastq --max_memory 2G
```

Running Trinity using qsub

- Here, we are using a subset of data – when running Trinity on your full dataset, it will take a lot of time (few days in some cases) so you should run the script

```
#!/bin/bash
Path_to/Trinity   \
--seqType fq     \
--left all_R1.fastq   \
--right all_R2.fastq.  \
--max_memory 2G    \
…other options…
```

- Submit the job (track the job with `myjobs` or by opening the log that is created by the run)

```
qsub -cwd -N log -l h_data=12G,h_rt=100:00:00 -pe shared 4 -M $USER -m bae trinity.sh
```

3) Output – Trinity.fasta

Trinity ID

> TRINITY_DN001_c0_g1_i1

gene isoform

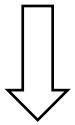
Transcriptome assembly quality assessment

- RNA-Seq read representation of the assembly
- Representation of full-length reconstructed protein-coding genes
- Contig Nx and ExN50 Statistics
- BUSCO
- DETONATE
- TransRate

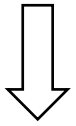
Trinity assembly pipeline

1) Input

Cleaned RNA-seq reads
FASTA or FASTQ format



2) Run Trinity



3) Output

Trinity.fasta
FASTA file with assembled transcripts

Let’s look how good our assembly is!

Some ways to do this:

Transcriptome assembly quality assessment

- The contig Nx statistic
- Assessing the read content of the transcriptome assembly
- Representation of full-length reconstructed protein-coding genes
- TransRate
- BUSCO
- DETONATE

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

Let’s look how good our assembly is!

Some ways to do this:

Transcriptome assembly quality assessment

- The contig Nx statistic
- Assessing the read content of the transcriptome assembly
- Representation of full-length reconstructed protein-coding genes
- TransRate

```
%  $TRINITY_HOME/util/TrinityStats.pl  Trinity.fasta

################################
## Counts of transcripts, etc.
################################
Total trinity 'genes':       1388798
Total trinity transcripts:   1554055
Percent GC: 44.52

########################################
Stats based on ALL transcript contigs:
########################################

Contig N10: 5264
Contig N20: 3136
Contig N30: 1803
Contig N40: 989
Contig N50: 606

Median contig length: 288
Average contig: 511.61
Total assembled bases: 795066996
```

Let’s look how good our assembly is!

Some ways to do this:

Transcriptome assembly quality assessment

- The contig Nx statistic
- Assessing the read content of the transcriptome assembly
- Representation of full-length reconstructed protein-coding genes
- TransRate

```
$TRINITY_HOME/util/misc/contig_ExN50_statistic.pl \
 transcripts.TMM.EXPR.matrix Trinity.fasta | tee ExN50.stats
```

Ex	ExN50	num_genes
2	2397	1
4	2397	2
5	2397	3
7	869	4
8	2397	5
...	.....	....
89	3351	19635
90	**3457**	**23471**
91	3560	28583
92	3655	35832
93	3706	47061
94	3658	66696
95	3444	104654
96	3109	171732
97	2683	275376
98	2163	428285
99	1512	668589
100	606	1554055

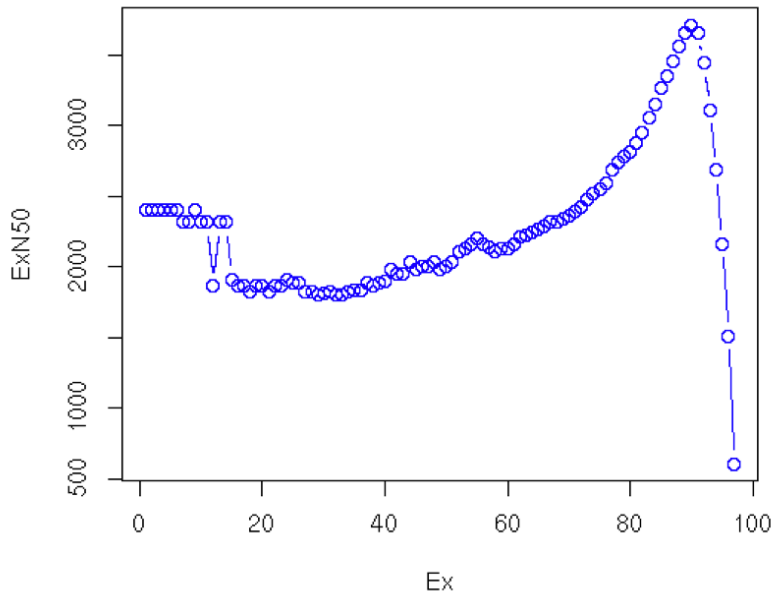
Let’s look how good our assembly is!

Some ways to do this:

Transcriptome assembly quality assessment

- The contig Nx statistic
- Assessing the read content of the transcriptome assembly
- Representation of full-length reconstructed protein-coding genes
- TransRate

```
$TRINITY_HOME/util/misc/contig_ExN50_statistic.pl \
 transcripts.TMM.EXPR.matrix Trinity.fasta | tee ExN50.stats
```



89	3351	19635
90	**3457**	**23471**
91	3560	28583
92	3655	35832
93	3706	47061
94	3658	66696
95	3444	104654
96	3109	171732
97	2683	275376
98	2163	428285
99	1512	668589
100	606	1554055

Let’s look how good our assembly is!

Some ways to do this:

Transcriptome assembly quality assessment

- The contig Nx statistic
- Assessing the read content of the transcriptome assembly
- Representation of full-length reconstructed protein-coding genes
- TransRate

```
bowtie2 -p 10 -q --no-unal -k 20 -x Trinity.fasta -1 reads_1.fq -2 reads_2.fq  \
   2>align_stats.txt| samtools view -@10 -Sb -o bowtie2.bam

cat 2>&1 align_stats.txt
```

```
76201190 reads; of these:
  76201190 (100.00%) were paired; of these:
    18166307 (23.84%) aligned concordantly 0 times
    17026716 (22.34%) aligned concordantly exactly 1 time
    41008167 (53.82%) aligned concordantly >1 times
    ----
    18166307 pairs aligned concordantly 0 times; of these:
      1769907 (9.74%) aligned discordantly 1 time
    ----
    16396400 pairs aligned 0 times concordantly or discordantly; of these:
      32792800 mates make up the pairs; of these:
        15287552 (46.62%) aligned 0 times
        3874965 (11.82%) aligned exactly 1 time
        13630283 (41.56%) aligned >1 times
89.97% overall alignment rate
```

Let’s look how good our assembly is!

Some ways to do this:

Transcriptome assembly quality assessment

- The contig Nx statistic
- Assessing the read content of the transcriptome assembly
- Representation of full-length reconstructed protein-coding genes
- TransRate

```
Build a blastable database:

%  makeblastdb -in uniprot_sprot.fasta -dbtype prot


Perform the blast search, reporting only the top alignment:

 % blastx -query Trinity.fasta -db uniprot_sprot.fasta -out blastx.outfmt6 \
       -evalue 1e-20 -num_threads 6 -max_target_seqs 1 -outfmt 6
```

```
$TRINITY_HOME/util/analyze_blastPlus_topHit_coverage.pl blastx.outfmt6 Trinity.fasta uniprot_sprot.fasta
```

Let’s look how good our assembly is!

Some ways to do this:

Transcriptome assembly quality assessment

- The contig Nx statistic
- Assessing the read content of the transcriptome assembly
- Representation of full-length reconstructed protein-coding genes
- TransRate

http://hibberdlab.com/transrate/index.html

Let’s practice!

The contig Nx statistic

- Log into the Hoffman2 account and locate your Trinity.fasta file
- Trinity package has a lot of scripts preloaded for post-assembly transcriptome quality check and analysis (let's have a look)

```
cd trinityrnaseq-v2.11.0/util
ls
```

- Run the TrinityStats.pl script on your Trinity.fasta file

```
./TrinityStats.pl Trinity.fasta
```

e.g. output:

```
################################
## Counts of transcripts, etc.
################################
Total trinity 'genes':       1388798
Total trinity transcripts:   1554055
Percent GC: 44.52

########################################
Stats based on ALL transcript contigs:
########################################

Contig N10: 5264
Contig N20: 3136
Contig N30: 1803
Contig N40: 989
Contig N50: 606

Median contig length: 288
Average contig: 511.61
Total assembled bases: 795066996
```

Assesing read content

- Build the bowtie index of the assembly

```
bowtie2-build Trinity.fasta Trinity.fasta
```

- Ailgn reads to the assembly

```
bowtie2 -p 10 -q --no-unal -k 20 -x Trinity.fasta -1 all_R1.fastq -2 all_R2.fastq \

2>align_stats.txt| samtools view -@10 -Sb -o bowtie2.bam
```

- Visualize statistics

```
cat 2>&1 align_stats.txt
```

Representation of full-length reconstructed protein-coding genes

- Download the BLAST software

```
wget https://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/2.9.0/ncbi-blast-2.9.0+-x64-linux.tar.gz
```

- Make a path for the BLAST directory

```
PATH=$PATH:path/to/directory
```

- Build a blastable database

```
makeblastdb -in uniprot_sprot.fasta -dbtype prot
```

- Perform the blast search, reporting only the top alignment

```
blastx -query Trinity.fasta -db uniprot_sprot.fasta -out blastx.outfmt6 \
-evalue 1e-20 -outfmt 0
```

- Examine the percent of the target being aligned to by the best matching Trinity transcript

```
$TRINITY_HOME/util/analyze_blastPlus_topHit_coverage.pl \
blastx.outfmt6 Trinity.fasta uniprot_sprot.fasta
```

OUTPUT: blastx.outfmt6.txt.w_pct_hit_length

Representation of full-length reconstructed protein-coding genes

OUTPUT: blastx.outfmt6.txt.w_pct_hit_length

hit_pct_cov_bin	count_in_bin	>bin_below
100	3242	3242
90	268	3510
80	186	3696
70	202	3898
60	216	4114
50	204	4318
40	164	4482
30	135	4617
20	76	4693
10	0	4693
0	0	4693

- There are 268 proteins that each match a Trinity transcript by >80% and <= 90% of their protein lengths.
- There are 3510 proteins that are represented by nearly full-length transcripts, having >80% alignment coverage.
- There are 3242 proteins that are covered by more than 90% of their protein lengths

Transrate

- Download Transrate

```
wget
https://bintray.com/artifact/download/blahah/generic/transrate-1.0.3-linux-x86_64.tar.gz

tar -xvzf
https://bintray.com/artifact/download/blahah/generic/transrate-1.0.3-linux-x86_64.tar.gz
```

- Run Transrate

```
./transrate --assembly transcripts.fa --left
all_R1.fastq --right all_R2.fastq
```

output example:

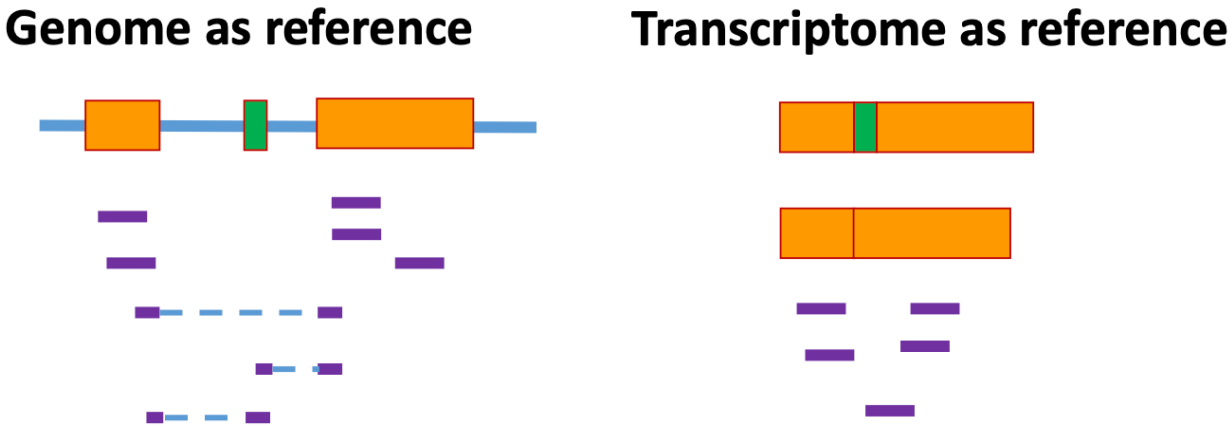
```
Contig metrics:
-----------------------------------
n seqs                        147600
smallest                          83
largest                         8133
n bases                     69200630
mean len                       447.77
n under 200                    23838
n over 1k                      12804
n over 10k                         0
n with orf                     24771
mean orf percent               64.33
N90                              276
N70                              441
N50                              670
N30                             1021
N10                             1878
gc                              0.44
bases n                       178748
proportion n                     0.0
```

Post Assembly Analysis

- Abundance quantification
- QC Samples and Biological Replicates
- Differential gene analysis
- Coding region identification
- Functional annotation

Abundance quantification

- Trinity supports abundance quantification using RSEM, Kallisto and Salmon
- RSEM - alignment-based method (aligning reads to the transcript assembly)
- Kallisto/Salmon - alignment-free methods (typically examining k-mer abundances in the reads and in the resulting assemblies)



- Trinity does not come prepackaged with these so make sure to install them before running

Abundance quantification

Abundance estimation can be run on gene or isoform level

```
>TRINITY_DN96_c0_g1_i1 len=376 path=[0:0-174 2:175-375]
ATCTATCCCCGGTTCTCGCCTACGAGCACACCTATGTACAGCAGTAGGCTCTATGCGAAAAACCCTGTGTTCGAAATTGTTCAAAAGGAGAAGCGTGAAGGTCTATCATCTACACAACATGTACAACGGGAATATTCA
GCAGAACCCTCTATGCGTGGAATCCTGTGTCACTAAGAACATCTATCCCCGGTTCTCGCCTACGAGCACACCTATGTACAGCAGTAGGCTCTATGCGAAAAACCCTGTGTTCGAAATTGTTCAAAAGGAGAAGCGTGA
AGGTCTATCATCTACACATGTACAACGGGAATATTCAGCAGAACCCTCTATGCGTGGAATCCTGTGTCACTAAGAACATCTATCCCCGGTTCTCGCCTAC
```

Trinity ID

> TRINITY_DN001_c0_g1_i1

gene isoform

Abundance quantification

- The Trinity toolkit comes with a script to facilitate running your choice of the above tools

```
$TRINITY_HOME/util/align_and_estimate_abundance.pl
--transcripts <string>
--seqType <string>
--left <string>
--right <string>
--samples_file <string>
```

```
condition_A_name     replicate_A1_name
condition_A_name     replicate_A2_name

condition_B_name     replicate_B1_name
condition_B_name     replicate_B2_name
```

```
--est_method <string>
--output_dir <string>
--aln_method <string>


--trinity_mode
```

transcript fasta file
fq|fa

tab-delimited text file indicating biological replicate relationships.
abundance estimation method
write all files to output directory
bowtie2|(path to bam file);
alignment method (if alignment_based est_method)
input reference is from Trinity

https://github.com/trinityrnaseq/trinityrnaseq/wiki/Trinity-Transcript-Quantification

Abundance quantification

Output (for salmon):

- quant.sf : transcript abundance estimates
- quant.sf.genes : gene-level abundance estimates

Name	Length	EffectiveLength	TPM	NumReads
TRINITY_DN10_c0_g1_i1	334	135.084	0	0
TRINITY_DN11_c0_g1_i1	319	120.926	6158.51	13
TRINITY_DN12_c0_g1_i1	244	57.9931	0	0
TRINITY_DN17_c0_g1_i1	229	47.8216	2395.84	2
TRINITY_DN18_c0_g1_i1	633	432.567	662.169	5
TRINITY_DN18_c1_g1_i1	289	93.8233	3663.47	6
TRINITY_DN19_c0_g1_i1	283	88.6594	646.141	1
TRINITY_DN21_c0_g1_i1	242	56.5791	1012.5	1

Abundance quantification

Next:
Using the transcript and gene-level abundance estimates for each sample, construct a matrix of counts and a matrix of normalized expression values:

```
$TRINITY_HOME/util/abundance_estimates_to_matrix.pl
--est_method <string>

--gene_trans_map <string>
```

RSEM|kallisto|salmon (needs to know what format to expect)
the gene-to-transcript mapping file.
(if you don't want gene estimates, indicate 'none'

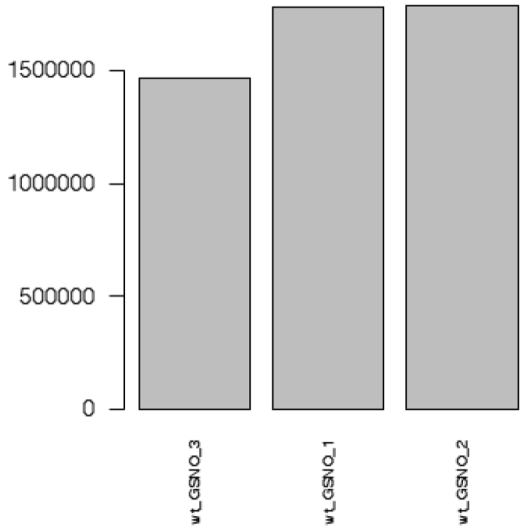
Output (for isoforms):

salmon.isoform.counts.matrix : the estimated RNA-Seq fragment counts (raw counts)
salmon.isoform.TPM.not_cross_norm : a matrix of TPM expression values (not cross-sample normalized)
salmon.isoform.TMM.EXPR.matrix : a matrix of TMM-normalized expression values

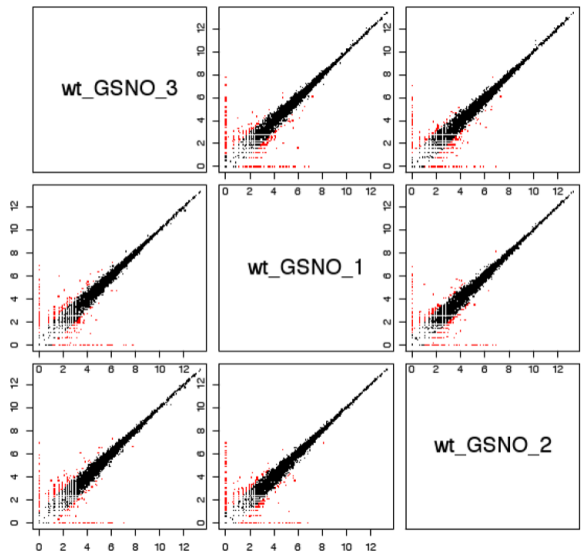
QC Samples and Biological Replicates

- Trinity has a script that can be used to generate a variety of plots for exploring your matrix of expression data.
- Needed to run:
  1. Fragment counts matrix ('counts.matrix')
  2. 'samples.txt' file, with tab-delimited format (replicate names should match identically to the column headers of your 'TMM.EXPR.matrix' and 'counts.matrix' file).

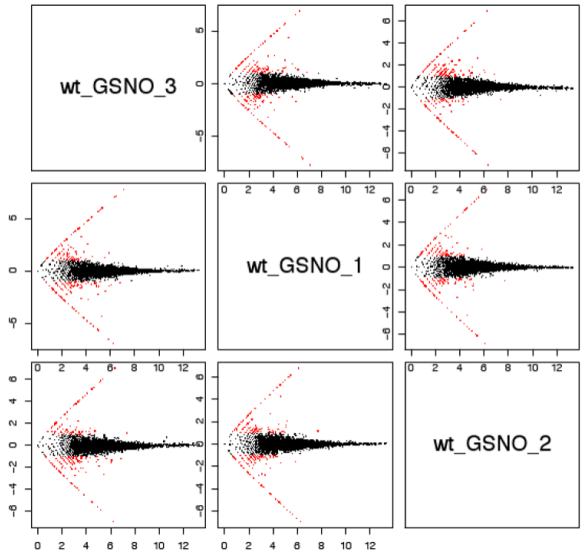
```
$TRINITY_HOME/Analysis/DifferentialExpression/PtR --matrix counts.matrix --samples samples.txt --log2 --min_rowSums 10 \
--compare_replicates
```



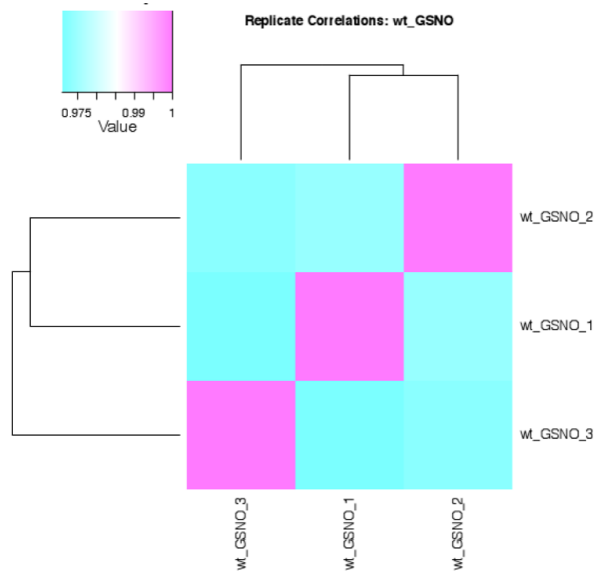
the sum of mapped fragments



Pairwise comparisons of replicate log(CPM) values



Pairwise MA plots (x-axis: mean log(CPM), y-axis log(fold_change))

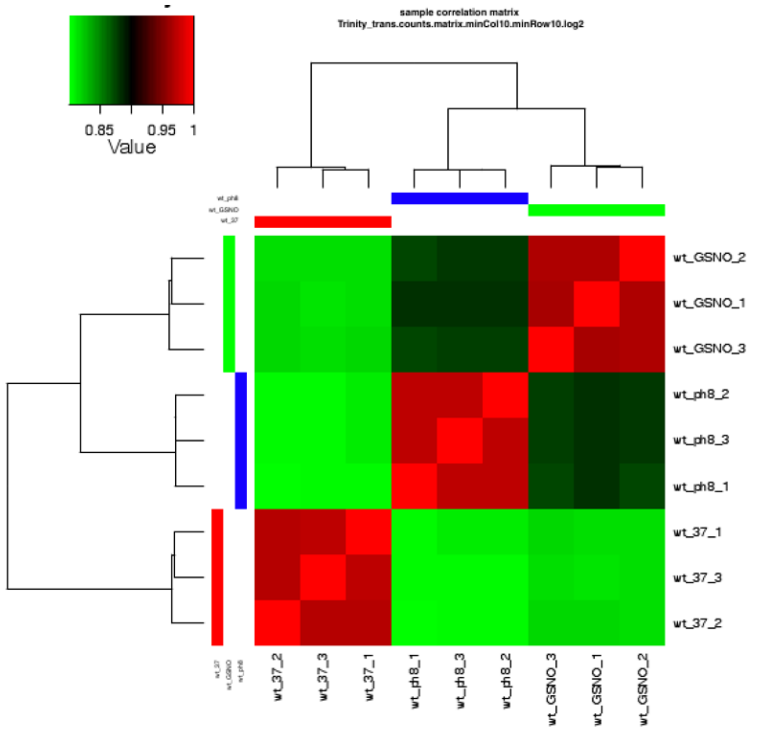


replicate Pearson correlation heatmap

QC Samples and Biological Replicates

- Compare replicates across all samples

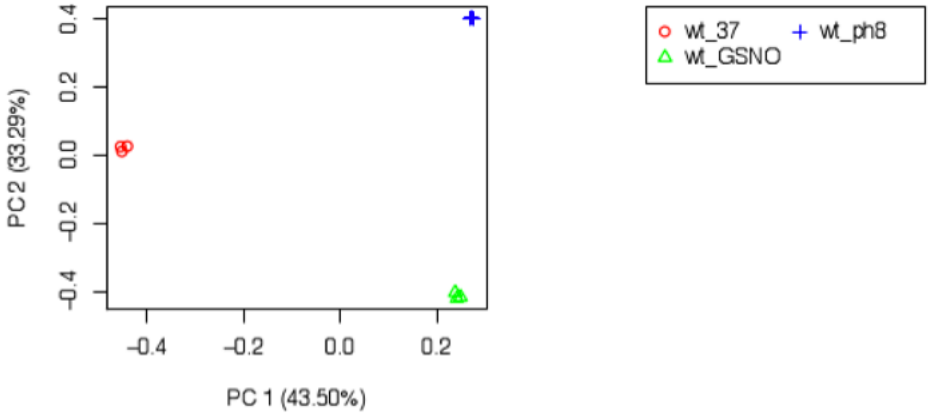
```
$TRINITY_HOME/Analysis/DifferentialExpression/PtR --matrix Trinity_trans.counts.matrix \
--min_rowSums 10 --s samples.txt --log2 --CPM --sample_cor_matrix
```

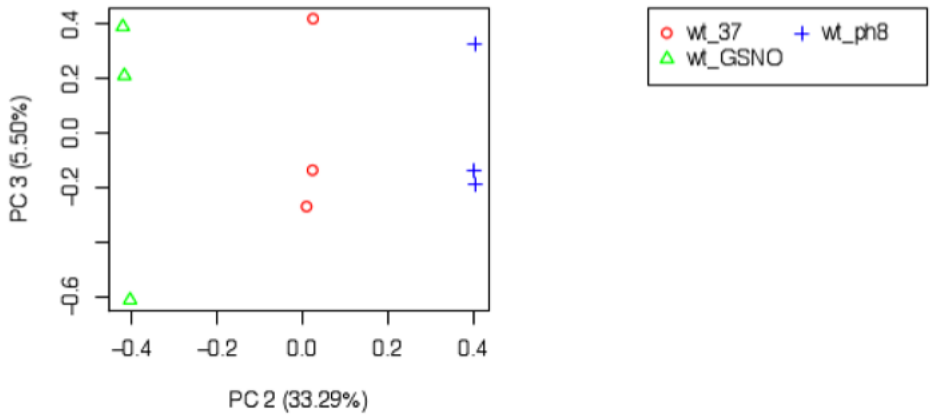


QC Samples and Biological Replicates

- Compare replicates across all samples - PCA

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





Identifying DE genes

- Reported for each gene:
- Normalized read count (Log2 count-per-million);
- Fold change between biological conditions (Log2 fold)
- Q(FDR).

```
$TRINITY_HOME/Analysis/DifferentialExpression/run_DE_analysis.pl
```

`--matrix|m <string>`	matrix of raw read counts (not normalized)
`--method`	edgeR|DESeq2|voom
`--samples_file|s <string>`	

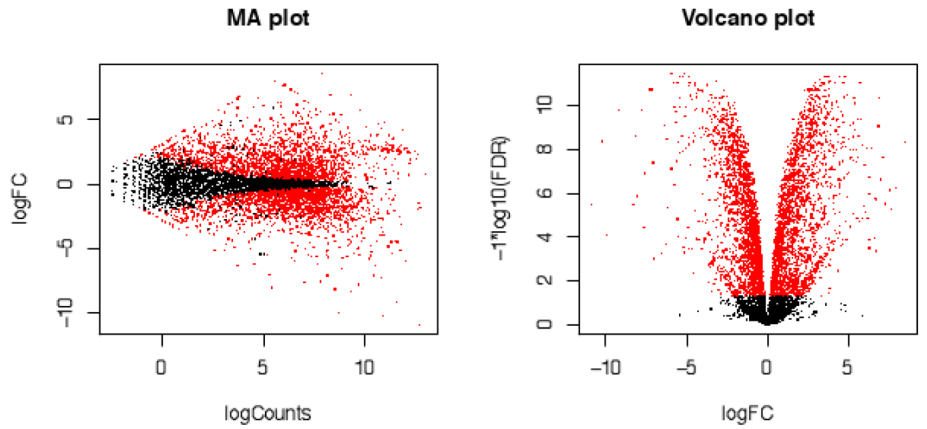
Identifying DE genes

Several output files:

- the DE analysis results table

	logFC	logCPM	pval
TR3679_c0_g1_i1	-5.11541024334567	11.7653852968686	7.23188081710377e-16
TR2820_c0_g1_i1	-5.94194741644588	12.0478207481011	1.02760683781471e-15
TR3880_c0_g1_i1	-4.841676068963	11.1650356947446	2.03198563430982e-15
TR4554_c0_g1_i1	3.58688559685832	11.4005717047623	3.41037012439576e-15
TR2827_c0_g1_i1	3.76108564650662	11.1790325703142	5.01055012548484e-15

- MA and volcano plots



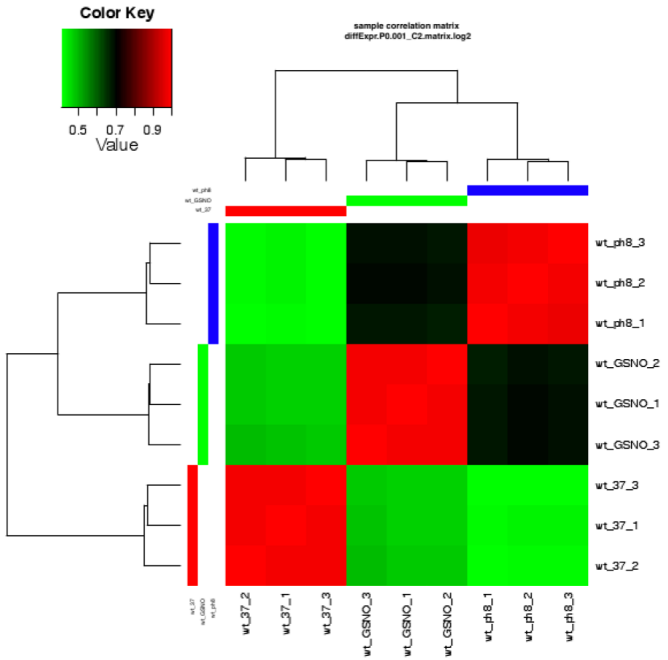
Clustering DE genes

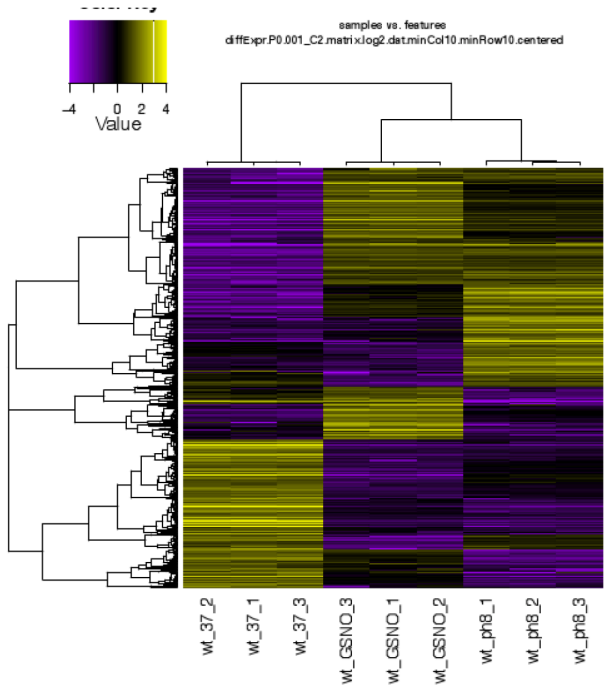
```
$TRINITY_HOME/Analysis/DifferentialExpression/analyze_diff_expr.pl
```

```
--matrix <string>
--P <float>
--C <float>
```

TMM.EXPR.matrix
p-value cutoff for FDR (default: 0.001)
min abs(log2(a/b)) fold change (default: 2
(meaning 2^(2) or 4-fold).

Output: several files including a correlation matrix and an expression heatmap (both table and visual representation)

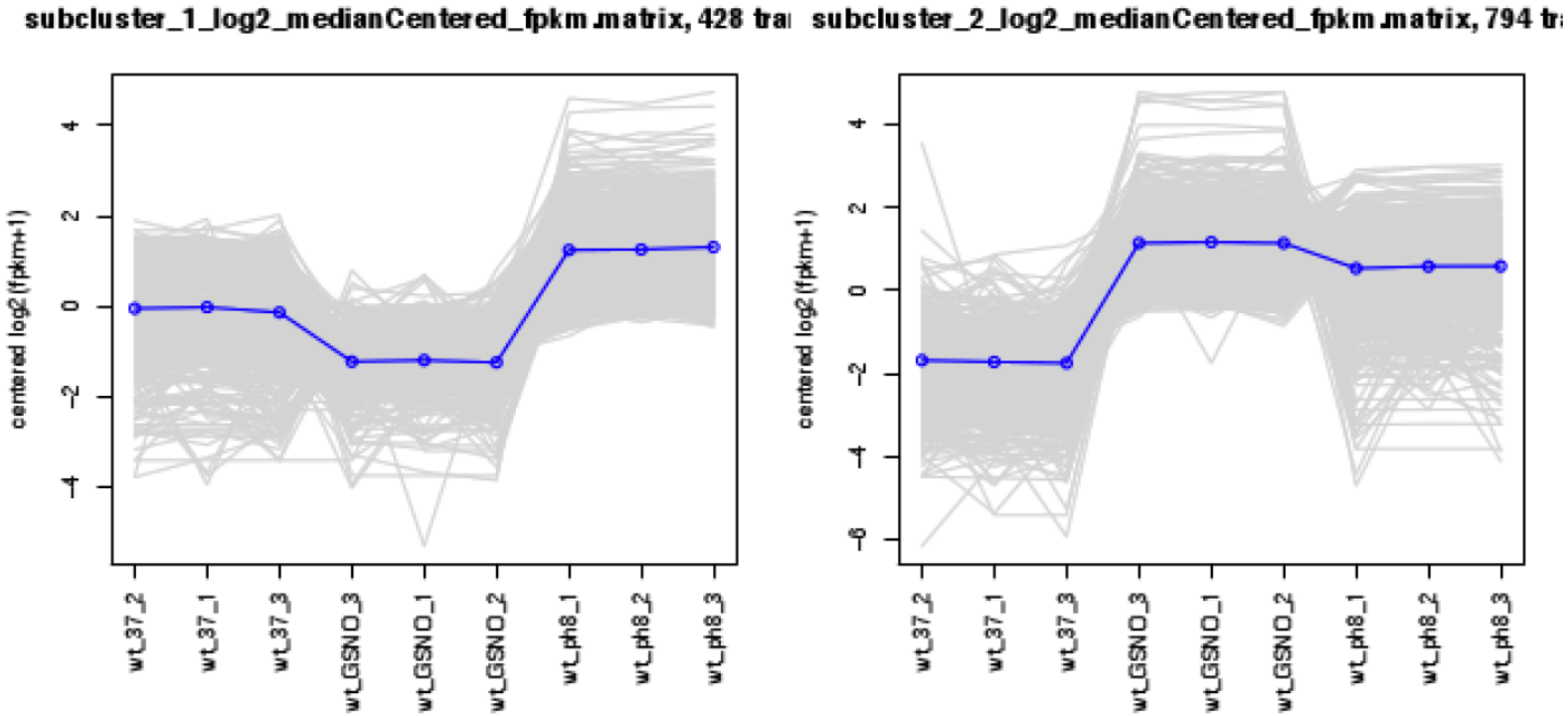




Clustering DE genes

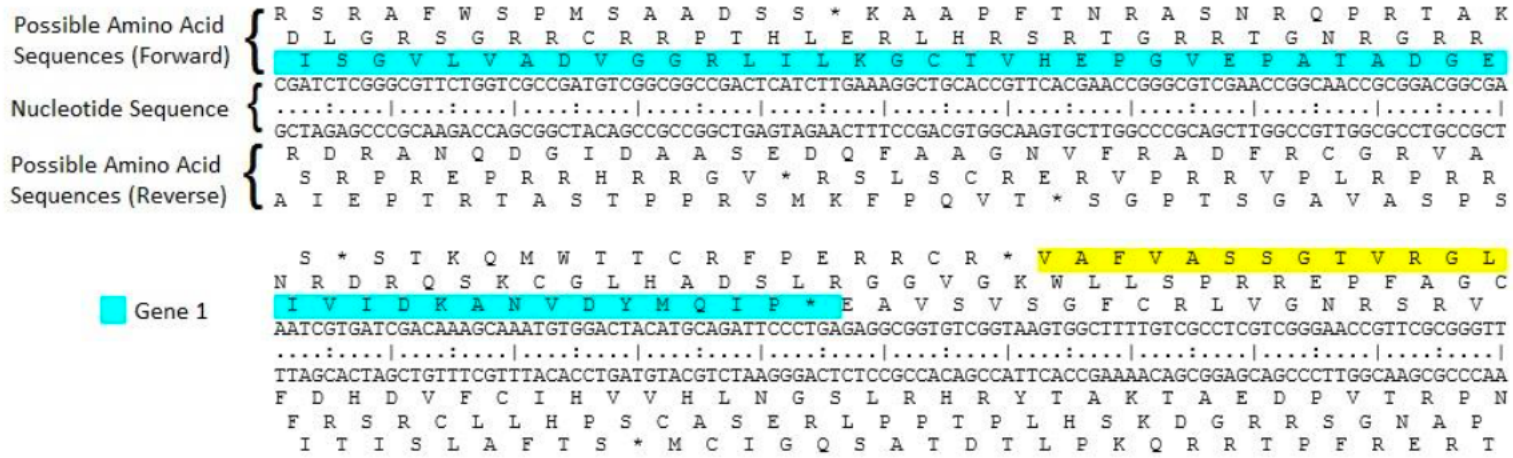
- Partitioning genes into expression clusters

```
$TRINITY_HOME/Analysis/DifferentialExpression/define_clusters_by_cutting_tree.pl
--R diffExpr.P0.001_C2.matrix.RData    \
--Ptree 60
```



Coding region identification

Using Transdecoder
https://github.com/TransDecoder/TransDecoder/wiki



- Step 1: extract the long open reading frames

```
TransDecoder.LongOrfs -t Trinity.fasta
```

- Step 2: identify ORFs with homology to known proteins via blast or pfam searches

```
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
```

- Step 3: predict the likely coding regions

```
TransDecoder.Predict -t target_transcripts.fasta
```

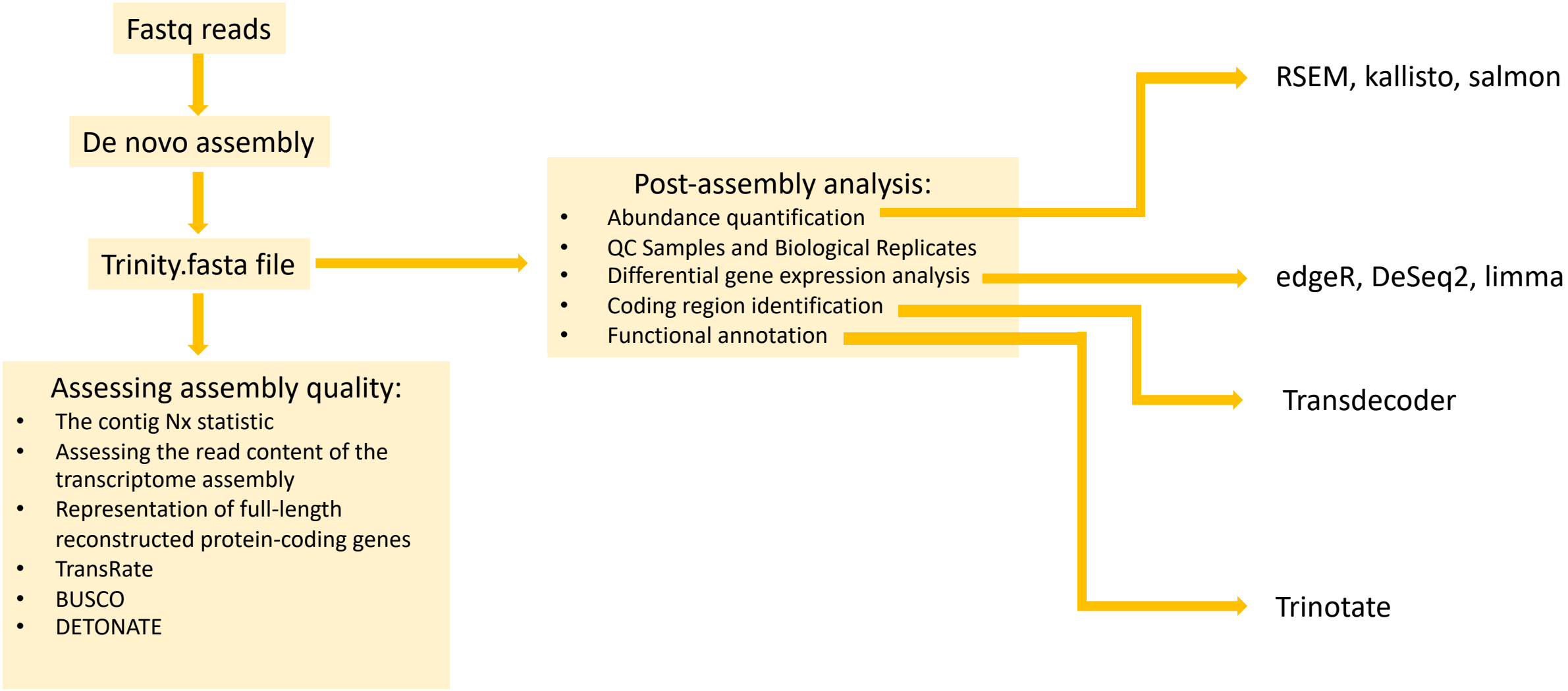
Functional annotation

Using Trinotate
https://github.com/Trinotate/Trinotate.github.io/wiki

- Several steps:
    1. Install required software
    2. Download required sequence databases (SwissProt, PFAM)
    3. Running sequence analysis
    4. Import the results from the analysis into a Trinotate SQLite database
    5. Output an annotation report

Let’s practice!

SUMMARY


Fastq reads
De novo assembly
Trinity.fasta file
Assessing assembly quality:
The contig Nx statistic
Assessing the read content of the transcriptome assembly
Representation of full-length reconstructed protein-coding genes
TransRate
BUSCO
DETONATE
Post-assembly analysis:
Abundance quantification
QC Samples and Biological Replicates
Differential gene expression analysis
Coding region identification
Functional annotation
RSEM, kallisto, salmon
edgeR, DeSeq2, limma
Transdecoder
Trinotate