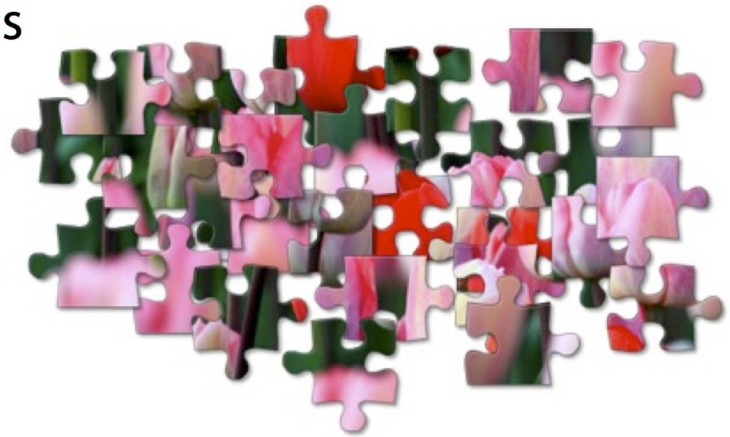
Genome and Transcriptome Assembly

FAS Informatics
October 23, 2014

Genome assembly: the goal

Reads



+

Reference genome

How to assemble puzzle without the benefit of knowing what the finished product looks like?

Input DNA

Slide courtesy Ben Langmead
http://www.langmead-lab.org/teaching-materials/

Genome assembly: the challenge

Reconstruct this

CTAGGCCCTCAATTTTT
GGCGTCTATATCT
CTCTAGGCCCTCAATTTTT
TCTATATCTCGGCTCTAGG
GGCTCTAGGCCCTCATTTTTT
CTCGGCTCTAGCCCCTCATTTT
TATCTCGACTCTAGGCCCTCA
GGCGTCGATATCT
TATCTCGACTCTAGGCC
GGCGTCTATATCTCG

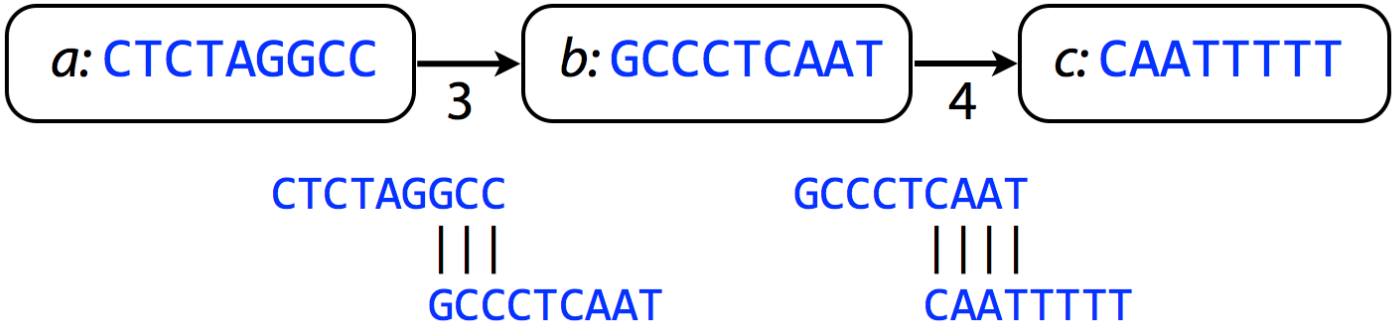
From these

GGCGTCTATATCTCGGCTCTAGGCCCTCATTTTTT

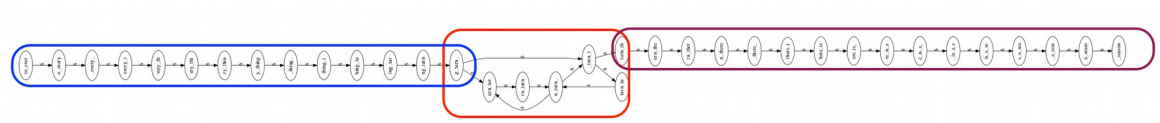
Slide courtesy Ben Langmead
http://www.langmead-lab.org/teaching-materials/

Traditional approach: OLC

Overlap



Layout



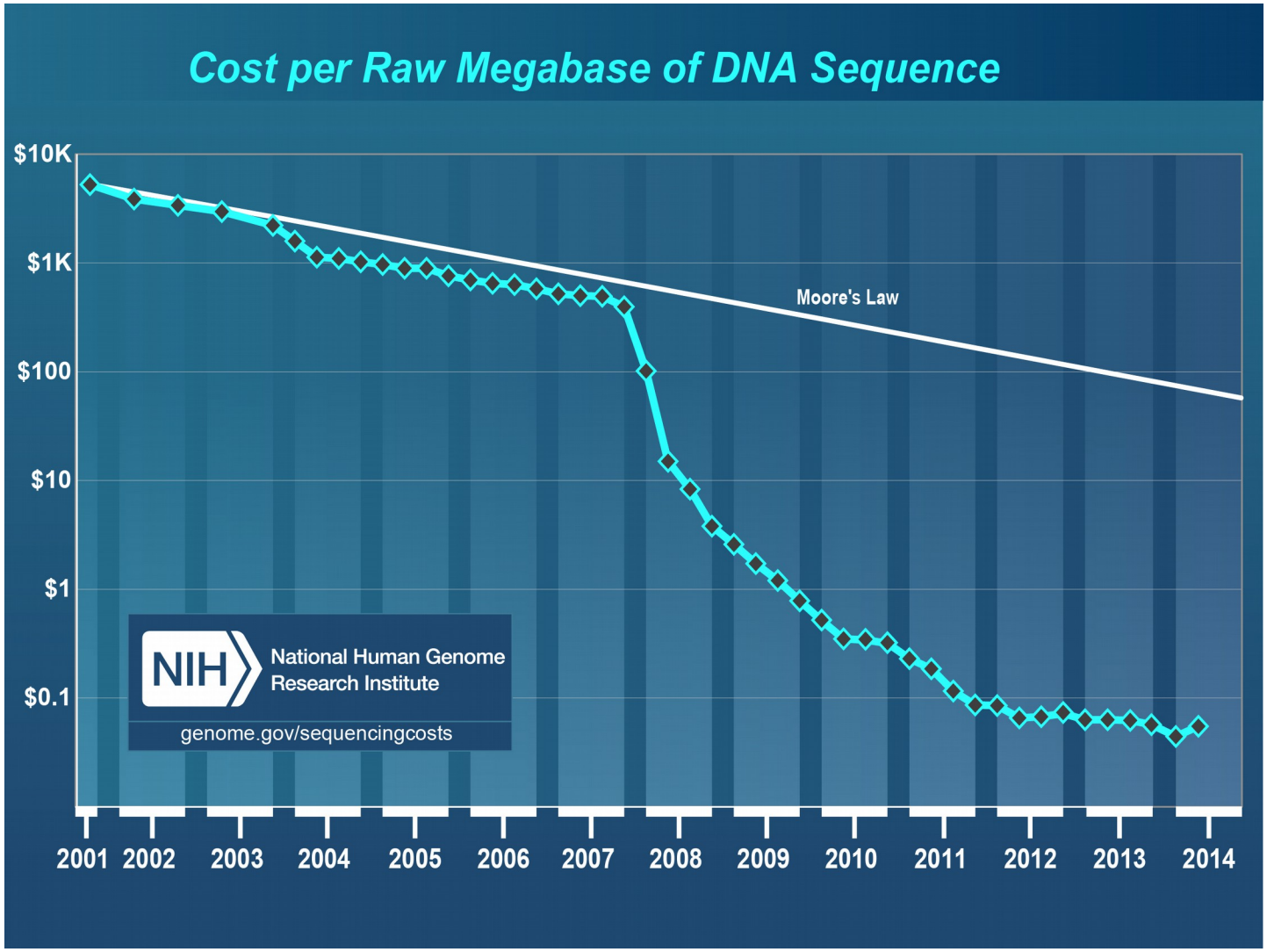
Consensus

```
TAGATTACACAGATTACTGA TTGATGGCGTAA CTA
TAGATTACACAGATTACTGACTTGATGGCGTAAACTA
TAG TTACACAGATTATTGACTTCATGGCGTAA CTA
TAGATTACACAGATTACTGACTTGATGGCGTAA CTA
TAGATTACACAGATTACTGACTTGATGGCGTAA CTA
```

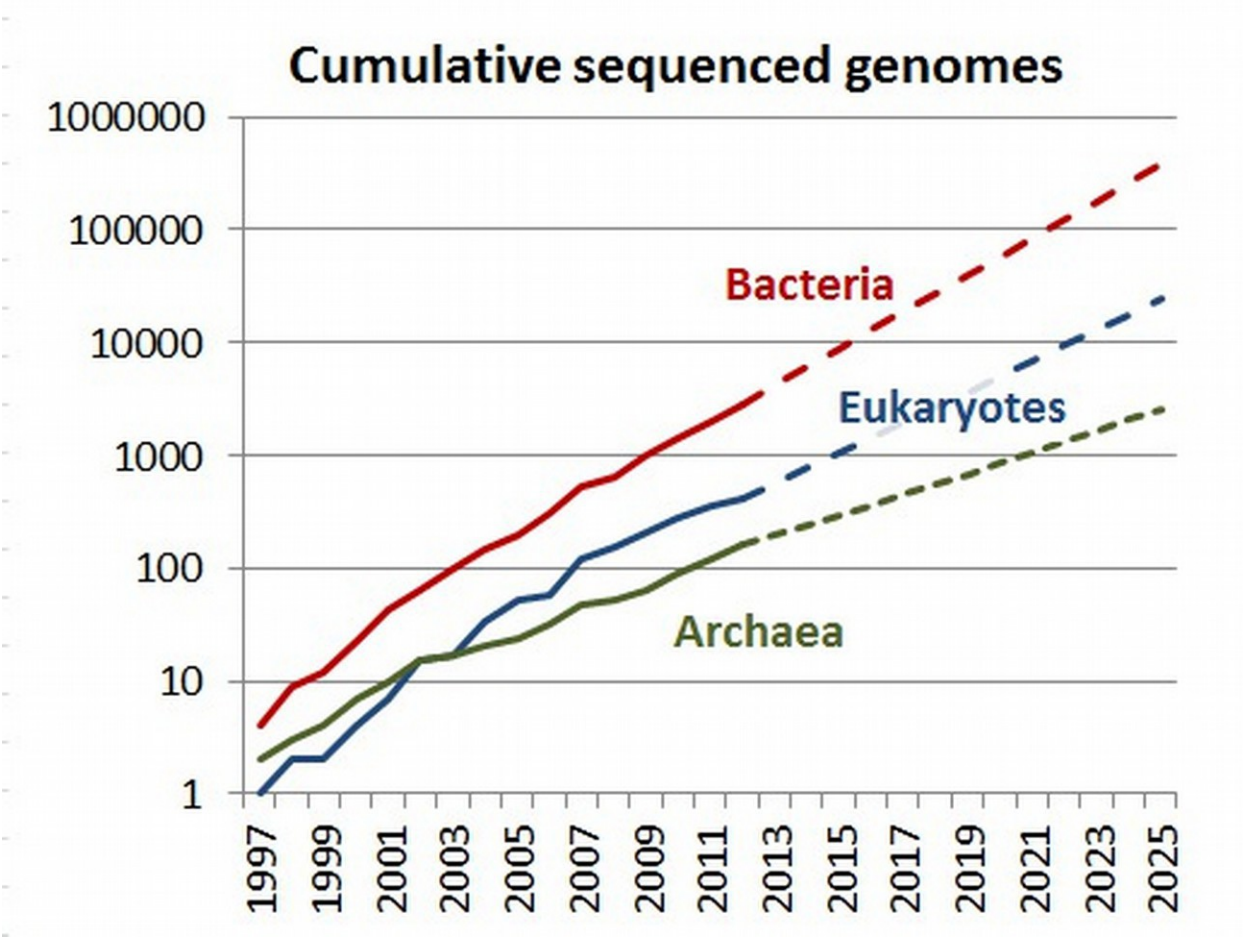
```
TAGATTACACAGATTACTGACTTGATGGCGTAA CTA
```

Images courtesy Ben Langmead
http://www.langmead-lab.org/teaching-materials/

Sequencing is increasingly cheap...



...and increasingly easy



http://figshare.com/articles/Cumulative_sequenced_genomes_GOLD_/723384

OLC is computationally impractical with 100s of millions to billions of reads.

New approaches required to deal with next generations sequencing.

De Bruijn Graph Assembly I

As usual, we start with a collection of reads, which are substrings of the reference genome.

AAA, AAB, ABB, BBB, BBA

AAB is a *k*-mer ($k = 3$). AA is its *left* *k*-1-mer, and AB is its right *k*-1-mer.

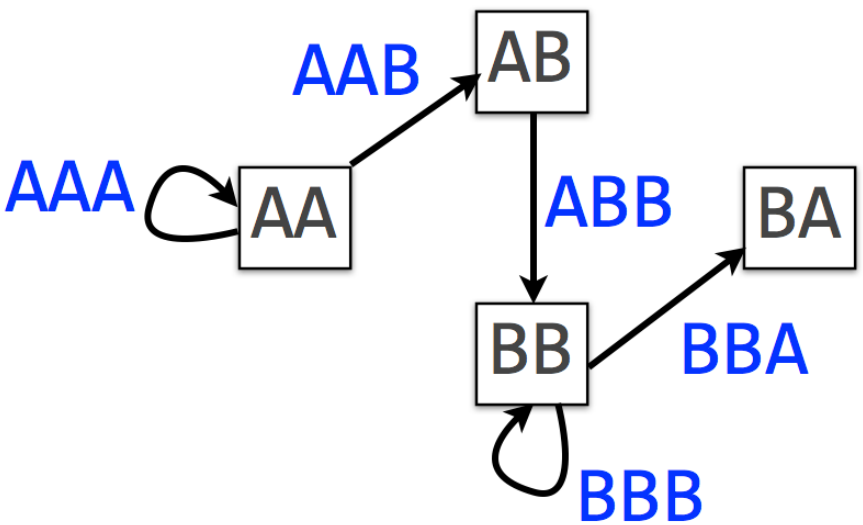
AAB 3-mer

AA L — AB R

AAB's left 2-mer — AAB's right 2-mer

Slide courtesy Ben Langmead
http://www.langmead-lab.org/teaching-materials/

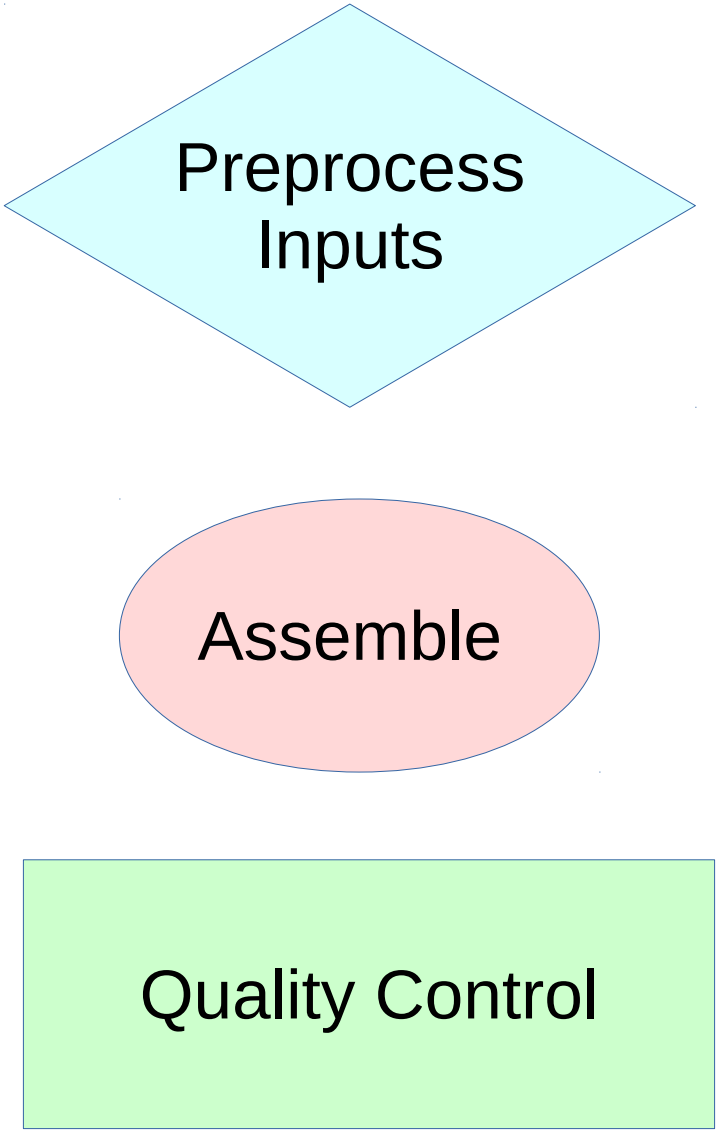
De Bruijn Graph Assembly II



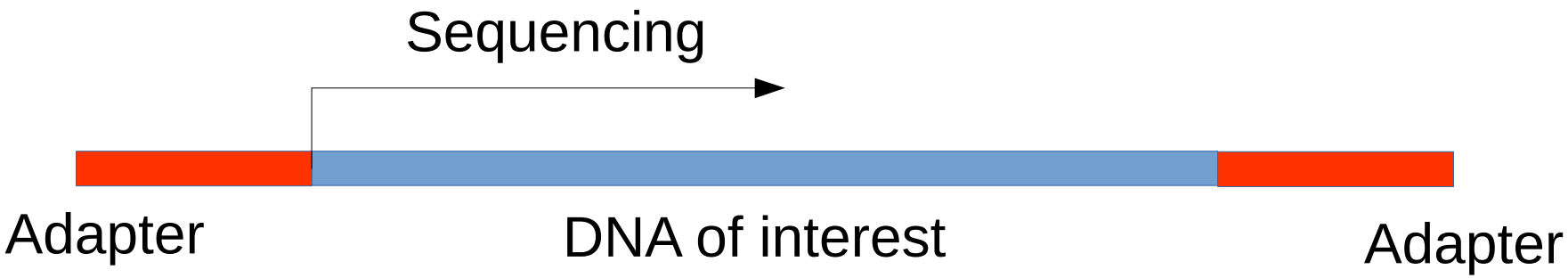
An edge corresponds to an overlap (of length *k*-2) between two *k*-1 mers.
More precisely, it corresponds to a *k*-mer from the input.

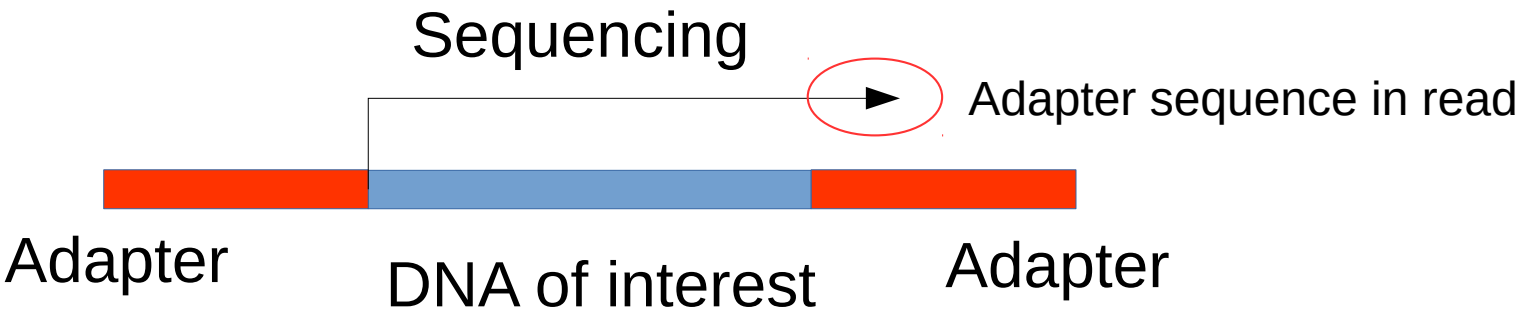
Slide courtesy Ben Langmead
http://www.langmead-lab.org/teaching-materials/

Assembly in Practice

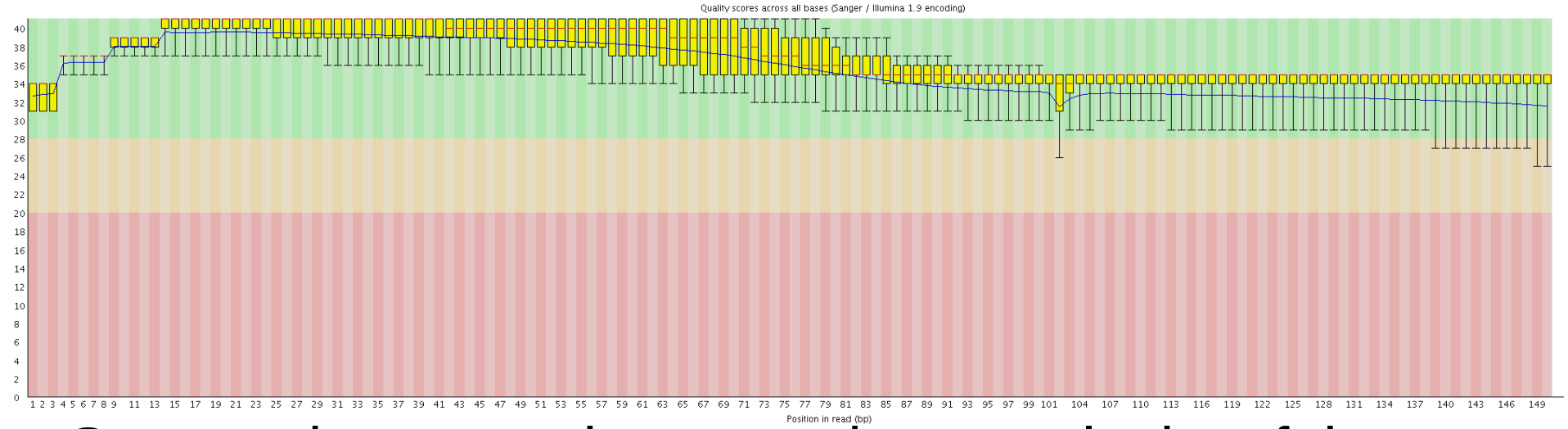


Genome Assembly – Preprocess: Remove adapter sequence





Genome Assembly – Preprocess: Trim Low Quality Sequence?

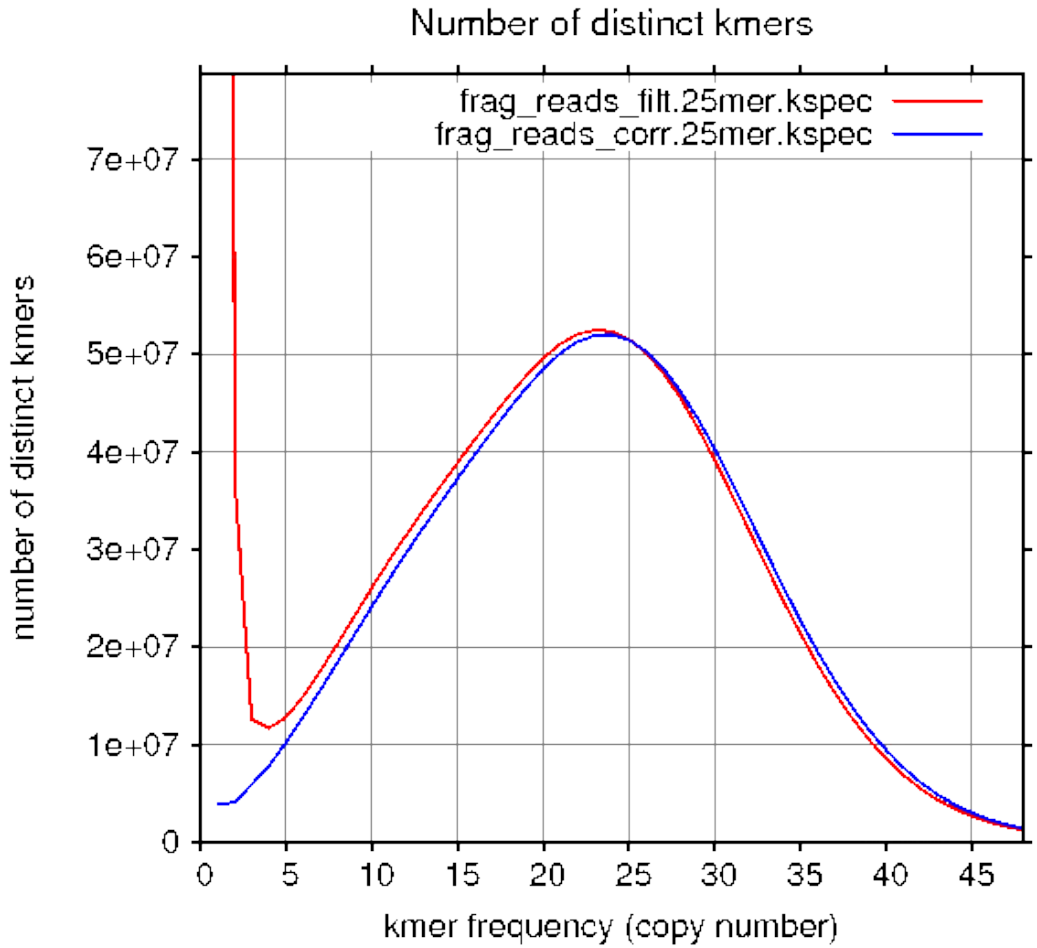


Sequencing errors increase the complexity of the De Bruijin graph.

Q20 = 99% accuracy
Q10 = 90% accuracy

Trimming throws out a LOT of correct data

Genome Assembly – Preprocess: Error Correction



Error correction, when possible, preserves data.

Not always feasible and not appropriate in some applications (RNA-seq)

Often build into assemblers

Genome Assembly – Preprocess: Code Example

```
#!/bin/bash

#SBATCH -p serial_requeue      #Which queue
#SBATCH -n 8                   #Number of CPUs
#SBATCH -N 1                   #Number of nodes
#SBATCH --mem 16000            #How much memory?
#SBATCH -t 8:00:00             #How much time? (HH:MM:SS)
#SBATCH -J trimFastq           #Job name
#SBATCH -o trim_%j.out         #STDERR (%j is replaced with jobid)
#SBATCH -e trim_%j.err         #STDOUT (%j is replaced with jobid)

module load centos6/Trimmomatic-0.32 #old module system

#Trimmomatic module creates a variable TRIMMOMATIC
#User inputs files - make sure there are NO SPACES OR SPECIAL CHARS

INFILE1=$1
INFILE2=$2
OUTFILE1=${1}_1P.fq.gz
OUTFILE2=${1}_1U.fq.gz
OUTFILE3=${2}_2P.fq.gz
OUTFILE4=${2}_2U.fq.gz
```

Genome Assembly – Preprocess: Code Example

(continued)

```
java -jar $TRIMMOMATIC/trimmomatic-0.32.jar PE -threads 8 \
    $INFILE1 $INFILE2 $OUTFILE1 $OUTFILE2 $OUTFILE3 $OUTFILE4 \
    ILLUMINACLIP:$TRIMMOMATIC/adapters/TruSeq3-PE:2:30:10:1:true \
    LEADING:3 \
    TRAILING:3 \
    SLIDINGWINDOW:4:10 \
    MINLEN:25
```

Run this script as:

```
                                  $1                 $2
sbatch trimfastq.sh infile1.fastq.gz infile2.fastq.gz
```

Will output:

```
infile1.fastq.gz_1P.fq.gz → reads from infile1 with pair intact
infile1.fastq.gz_1U.fq.gz → reads from infile1 where pair was removed
infile2.fastq.gz_2P.fq.gz → reads from infile2 with pair intact
infile2.fastq.gz_2U.fq.gz → reads from infile2 where pair was removed
```

If this doesn't look familiar to you, go to Bob Freeman's “Intro to Odyssey” workshop!

Next one is: **11/12/2014**

RSVP: http://goo.gl/FQ6z0T

Genome Assembly – Assemble

Reconstruct this

CTAGGCCCTCAATTTTT
GGCGTCTATATCT
CTCTAGGCCCTCAATTTTT
TCTATATCTCGGCTCTAGG
GGCTCTAGGCCCTCATTTTTT
CTCGGCTCTAGCCCCTCATTTT
TATCTCGACTCTAGGCCCTCA
GGCGTCGATATCT
TATCTCGACTCTAGGCC
GGCGTCTATATCTCG

From these

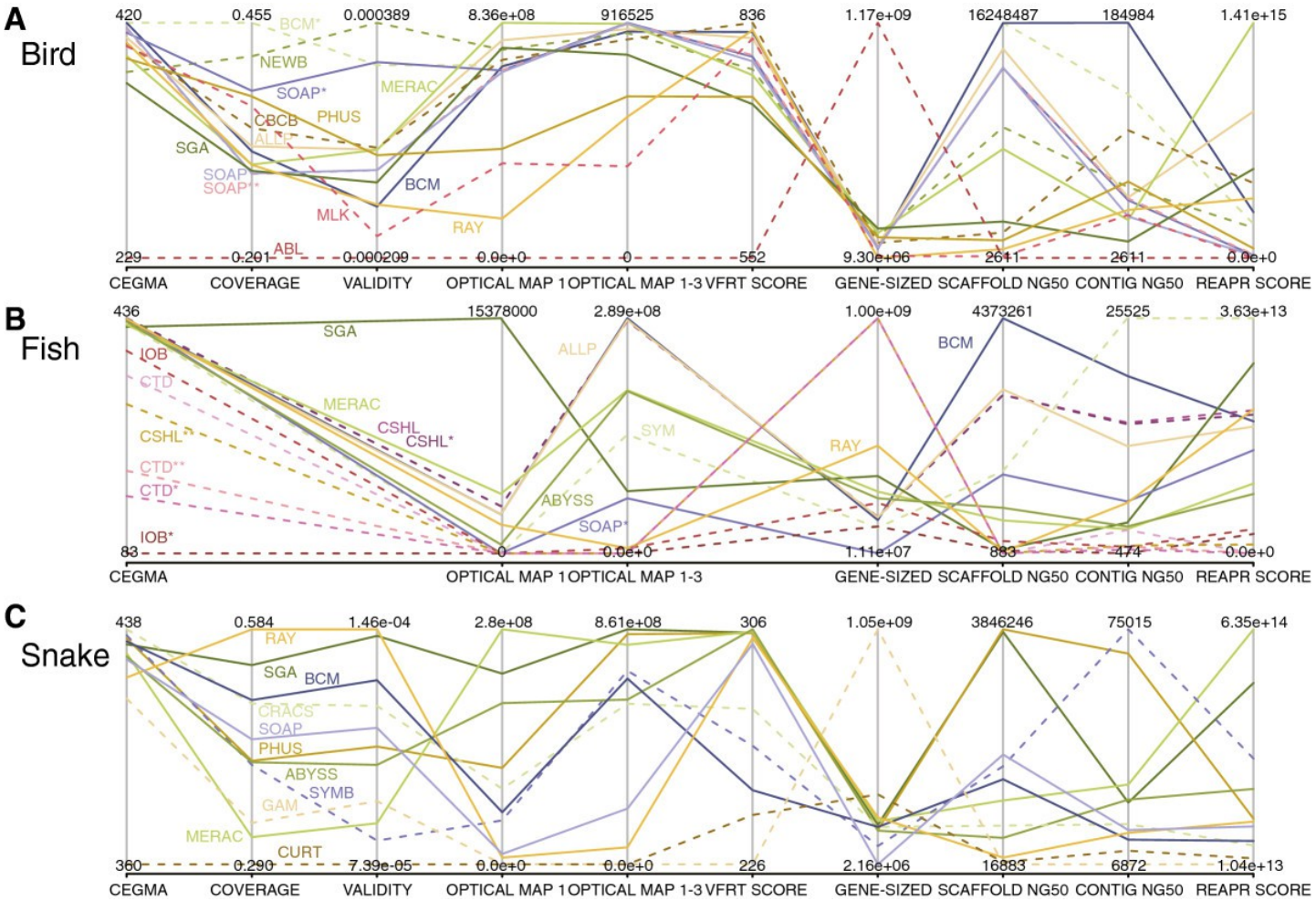
GGCGTCTATATCTCGGCTCTAGGCCCTCATTTTTT

Genome Assembly – Assemble: Tools

- Many options – the right choice depends on the data you have

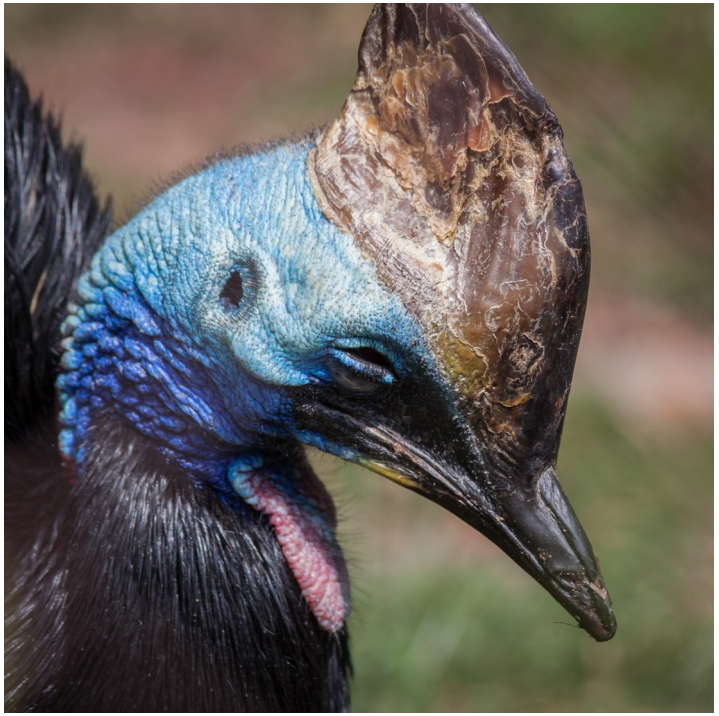
- Some examples:
  - ALLPATHS-LG
  - ABySS
  - SOAPdenovo
  - Velvet

Genome Assembly – Assemble: No Single Best Program



Bradnam et al. GigaScience 2013 2:10

Genome Assembly – Assemble: Example



Southern cassowary
(*Casuarius casuarius*)

419,359,436 fragment reads
(125 bp paired end,
280 bp insert size)

459,568,992 jumping reads
(125 bp mate pair,
3kb insert size)

ALLPATHS

1.1 GB assembly
133kb contig N50
3.7 MB scaffold N50
~$5000 total cost

Genome Assembly – Assemble: Code Example

```
RunAllPathsLG PRE=/n/regal/edwards_lab/ratites/allpaths_runs \
   REFERENCE_NAME=casCas \
   DATA_SUBDIR=RAWSEQ \
   RUN=20140828 \
   HAPLOIDIFY=TRUE \
   THREADS=32 \
   OVERWRITE=TRUE \
   1> casCas_20140828.log \
   2> CasCas_20140828.err
```

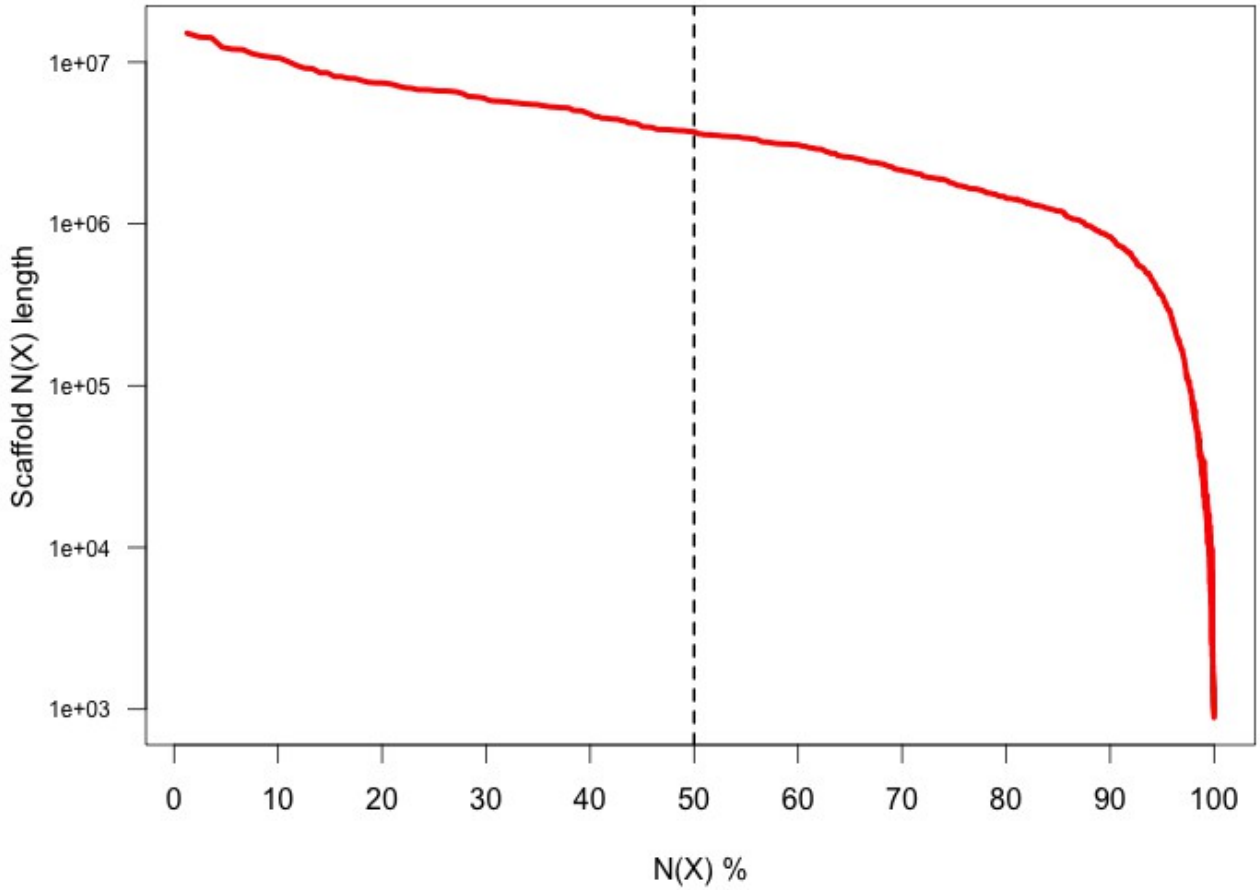
Genome Assembly – QC: Approach

- How *complete* is your assembly?
  - Contiguity
  - Gappiness
- How *accurate* is your assembly?
  - Correctness

Genome Assembly – QC: Approach

- How *complete* is your assembly?
  - Contiguity
  - Gappiness
- How *accurate* is your assembly?
  - Correctness

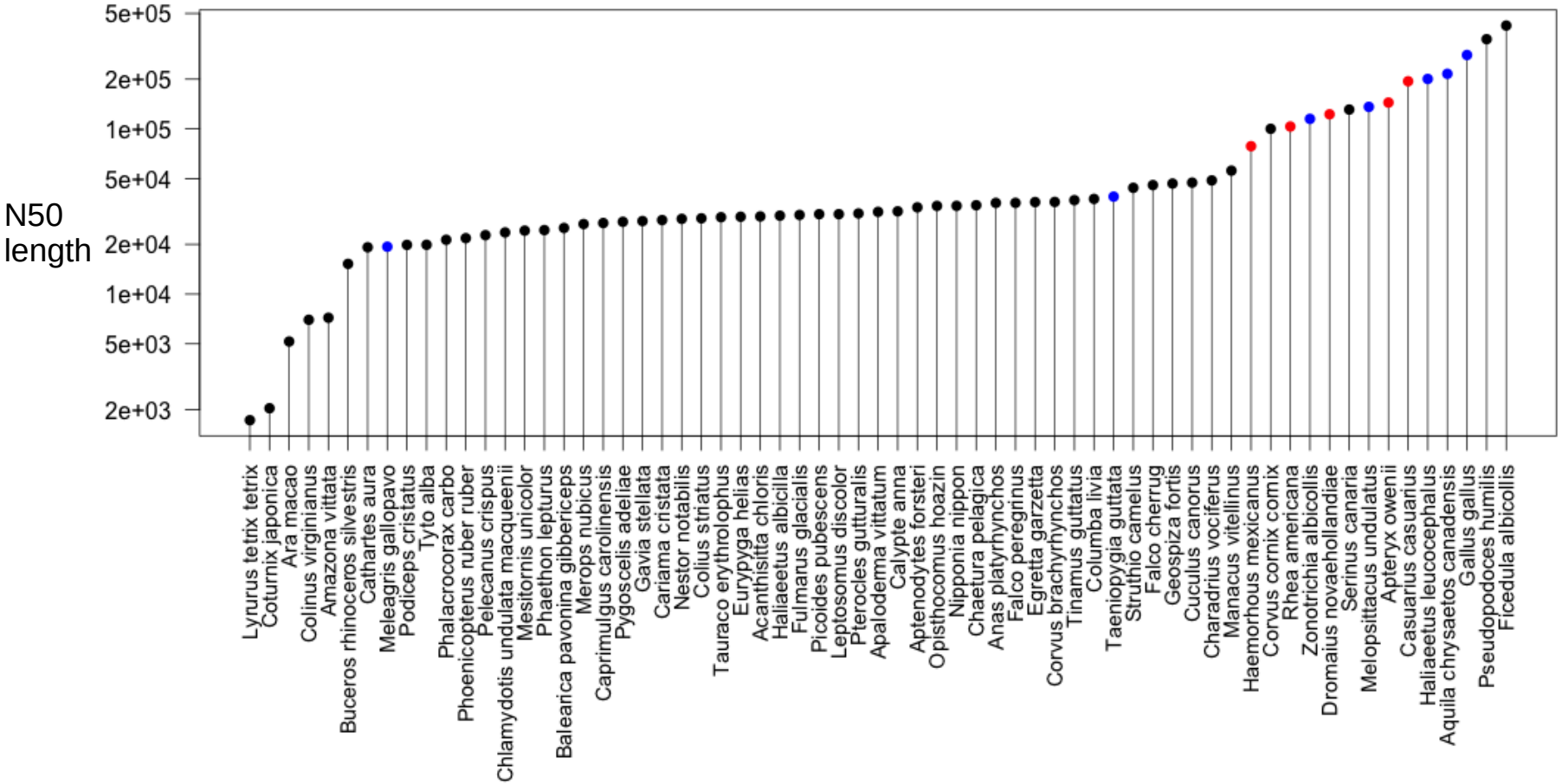
Genome Assembly – QC: Contiguity



Example: Cassowary

Genome Assembly – QC: Contiguity

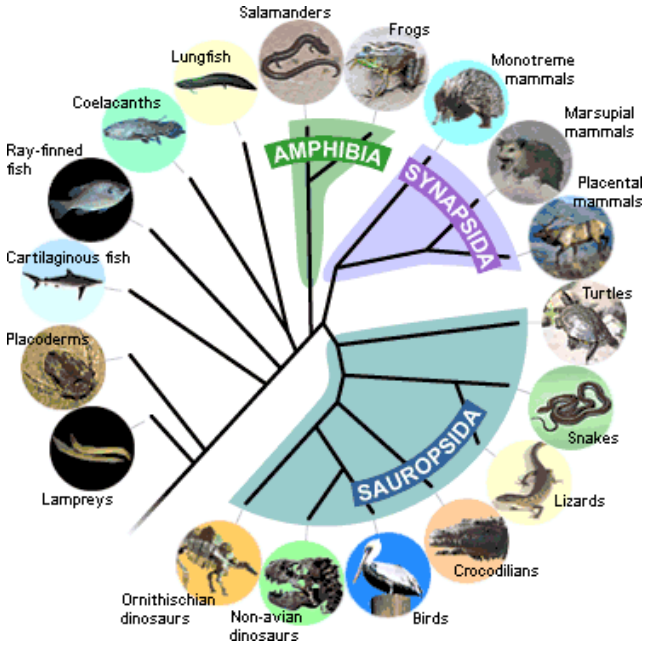
Contig N50 for ~60 bird genomes



Genome Assembly – QC: Approach

- How *complete* is your assembly?
  - Contiguity
  - Gappiness
- How *accurate* is your assembly?
  - Correctness

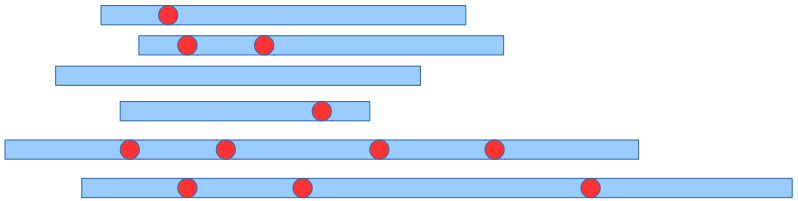
Genome Assembly – QC: Completeness



Select gene families with a single copy in all sequenced vertebrate genomes (BUSCOs from OrthoDB)

Extract copy from closest relative to target genome

TBLASTN

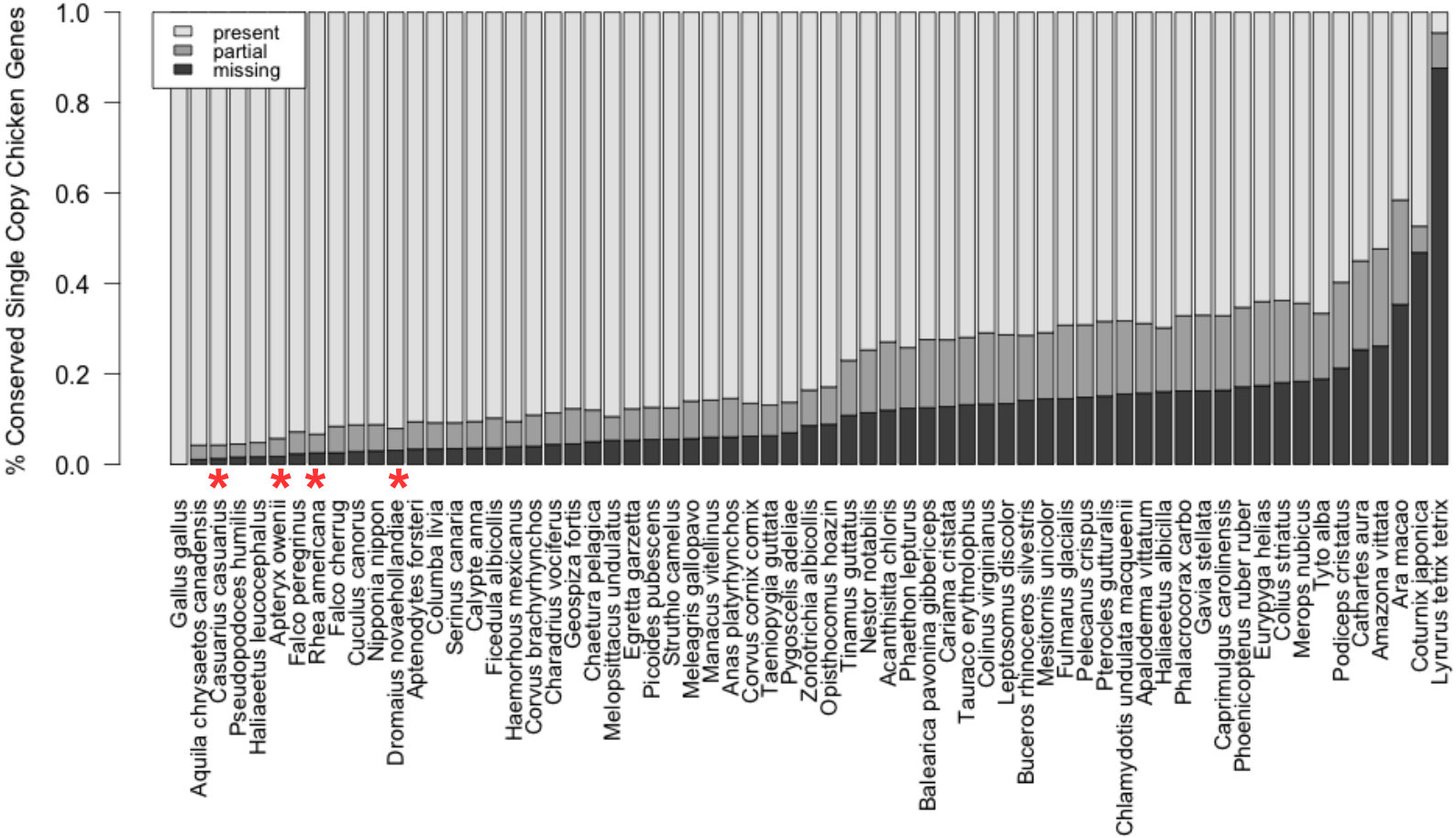


Identify hits and parse

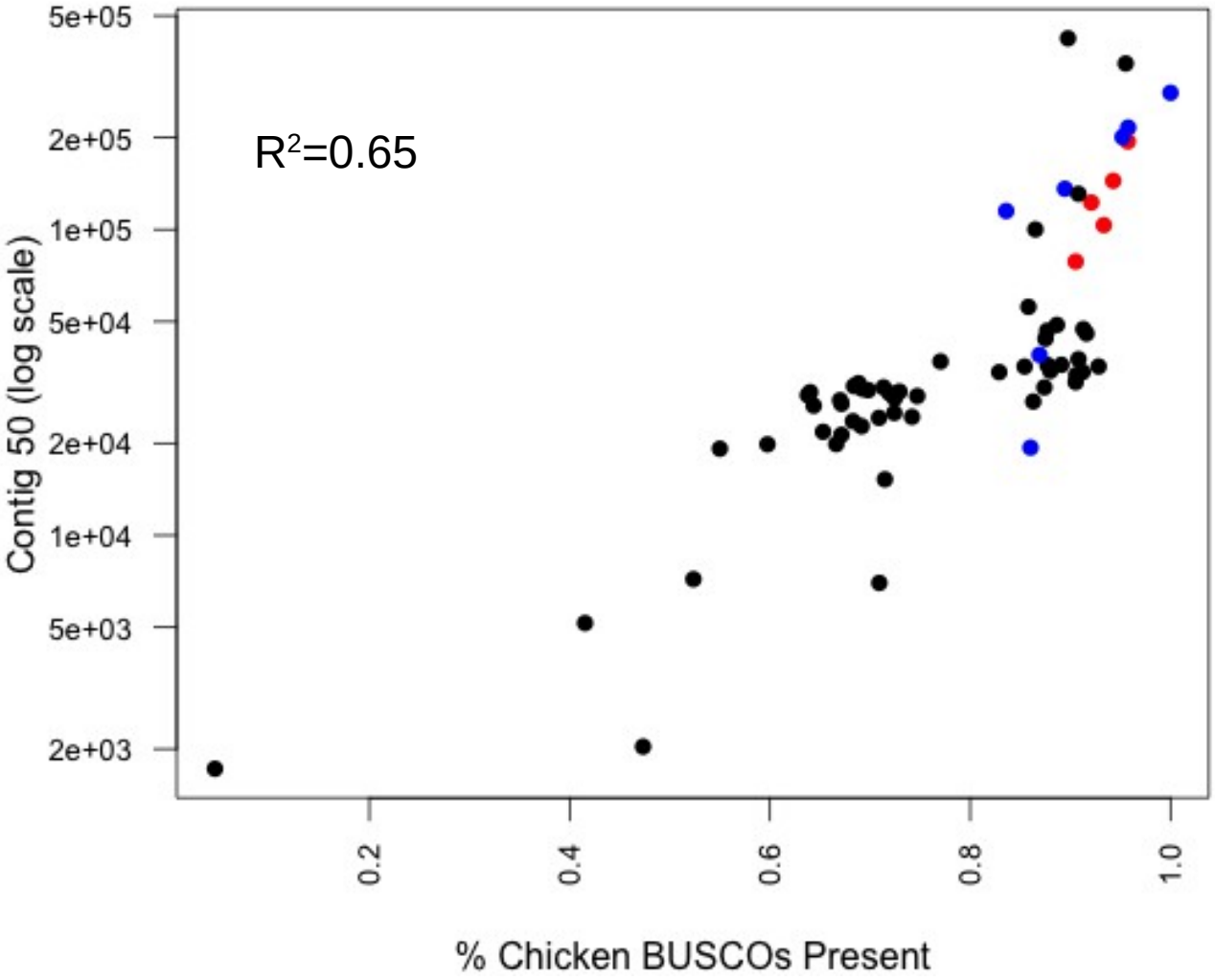
Phylogeny image:University of California Museum of Paleontology's Understanding Evolution (http://evolution.berkeley.edu)
Photo: Tim Sackton

Software: ftp://cegg.unige.ch/OrthoDB7/BUSCO

Genome Assembly – QC: Completeness



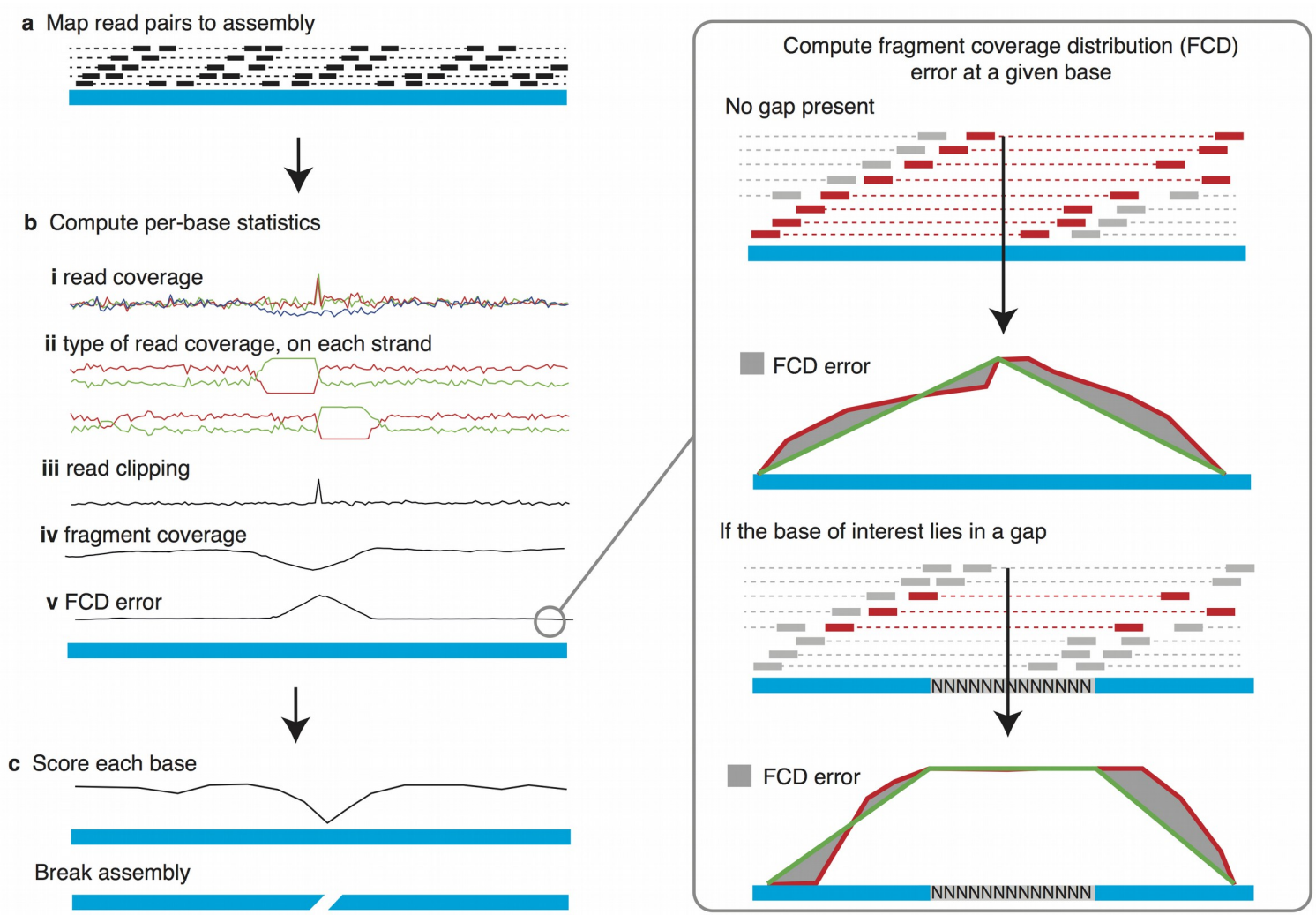
Genome Assembly – QC: Completeness



Genome Assembly – QC: Approach

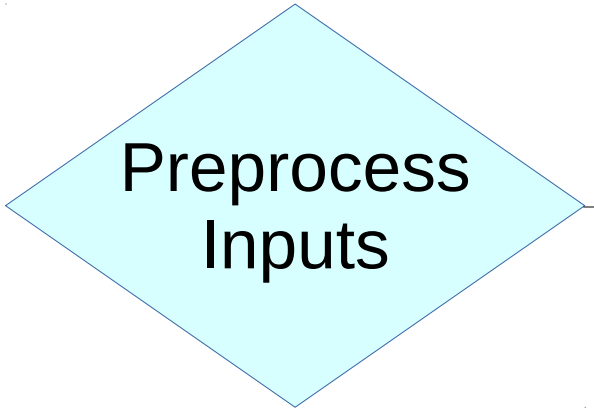
- How *complete* is your assembly?
  - Contiguity
  - Gappiness
- How *accurate* is your assembly?
  - Correctness

Genome Assembly – QC: Correctness



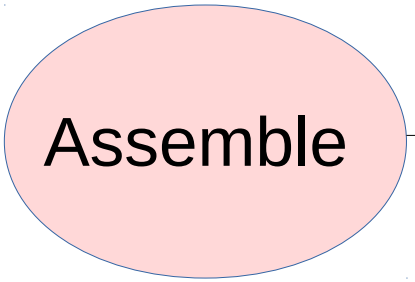
Hunt et al. Genome Biology 2013 14:R47

Genome Assembly: Summary



Always remove adapter sequence
Always visualize quality (FastQC)
Trim low quality sequence cautiously

Software: Trimmomatic, FastQC



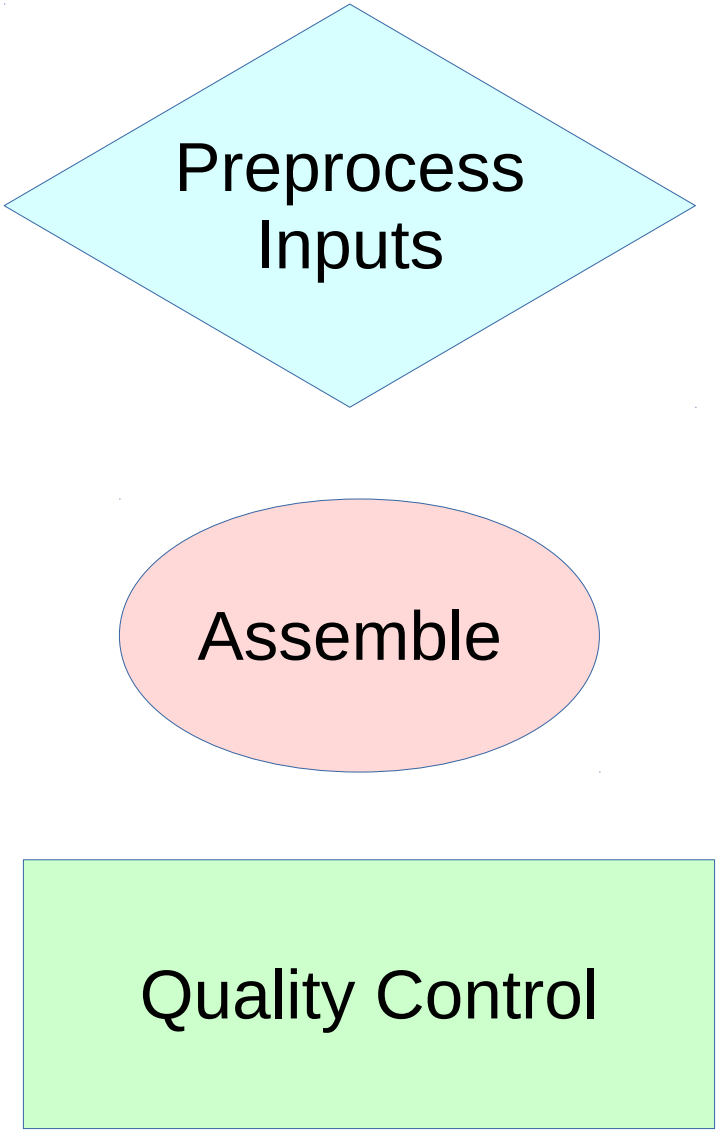
ALLPATHS recipe works very well
Best tool is data-dependent

Software: ABySS, ALLPATHS, Velvet, SOAPdenovo, etc

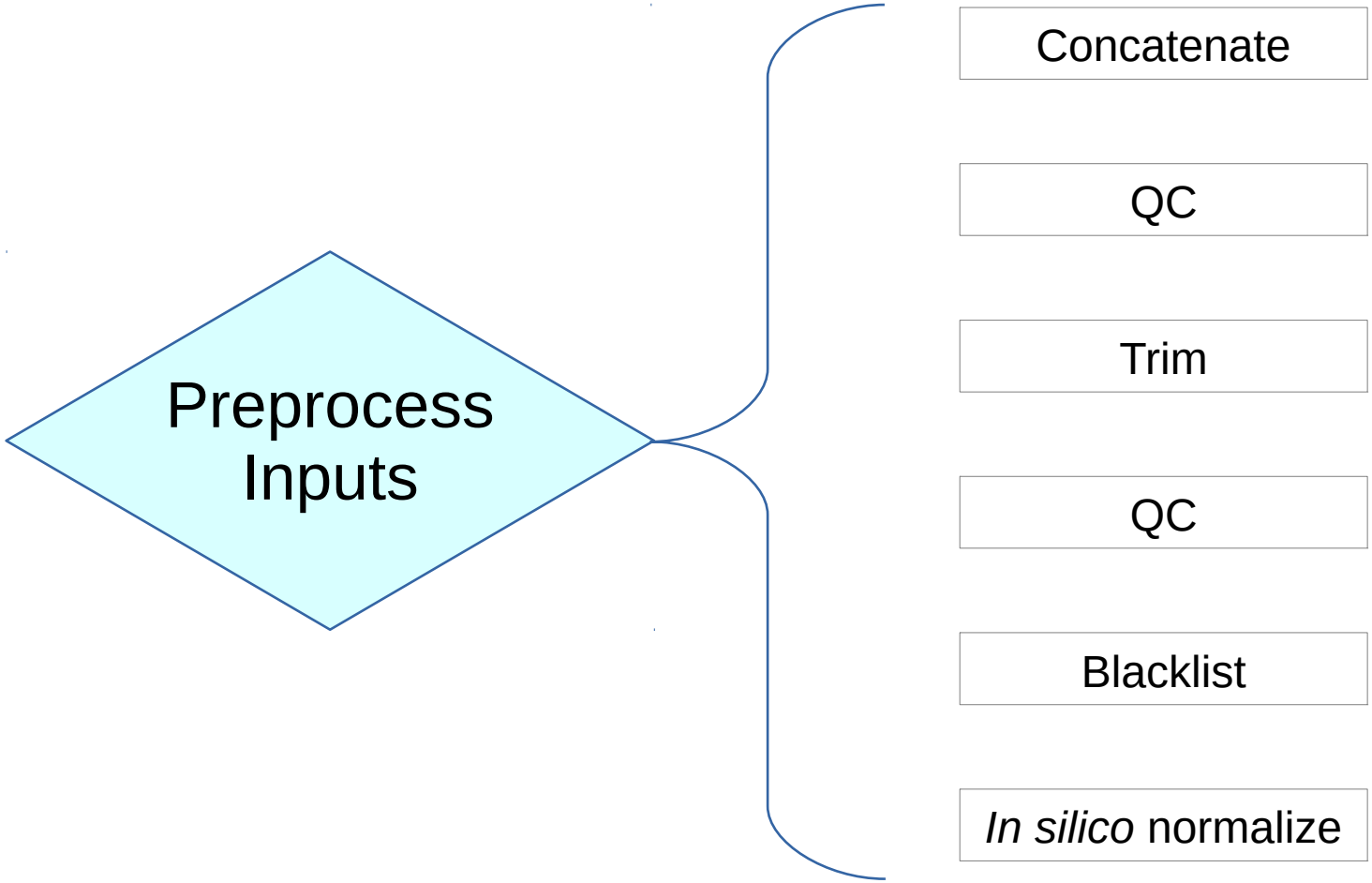
N50 is useful but not the final word

BUSCO: completeness
REAPR: correctness

Transcriptome Assembly



Transcriptome Assembly



Concatenate & QC

create file concat_n_fastqc.slurm in your favorite **text** editor

```
-----
#!/bin/bash

#SBATCH -p serial_requeue             # Partition to submit to
#SBATCH -n 1                          # Number of cores
#SBATCH -N 1                          # Ensure that all cores are on one machine
#SBATCH -t 0-2:00                     # Runtime in days-hours:minutes
#SBATCH --mem 500                     # Memory in MB
#SBATCH –J A1_fastqc                  # job name
#SBATCH -o A1_fastqc.out              # File to which standard out will be written
#SBATCH –e A1_fastqc.err              # File to which standard err will be written
#SBATCH --mail-type=ALL               # Type of email notification- BEGIN,END,FAIL,ALL
#SBATCH --mail-user=name@harvard.edu  # Email to which notifications will be sent

source new-modules.sh; module load fastqc

# concat first
cat raw/A1_TTAGGC_L006_R1_0{01,02,03,04,05}.fastq.gz > raw/A1_R1.fastq.gz
cat raw/A1_TTAGGC_L006_R2_0{01,02,03,04,05}.fastq.gz > raw/A1_R2.fastq.gz

# and now QC
fastqc --casava -o fastqc_reports raw/A1_R1.fastq.gz
fastqc --casava -o fastqc_reports raw/A1_R2.fastq.gz
-------

sbatch concat_n_fastqc.slurm
```

Trim

create file trim.slurm in your favorite **text** editor

```
-----
#!/bin/bash

#SBATCH -p serial_requeue             # Partition to submit to
#SBATCH -n 4                          # Number of cores
#SBATCH -N 1                          # Ensure that all cores are on one machine
#SBATCH -t 0-6:00                     # Runtime in days-hours:minutes
#SBATCH --mem 2000                    # Memory in MB
#SBATCH –J A1_trim                    # job name
#SBATCH -o A1_trim.out                # File to which standard out will be written
#SBATCH –e A1_trim.err                # File to which standard err will be written
#SBATCH --mail-type=ALL               # Type of email notification- BEGIN,END,FAIL,ALL
#SBATCH --mail-user=name@harvard.edu   # Email to which notifications will be sent

module load centos6/Trimmomatic-0.32

mkdir trimmed
java -jar $TRIMMOMATIC/trimmomatic-0.32.jar PE \
  -threads 4 \
  raw/A1_R1.fastq.gz raw/A1_R2.fastq.gz \
  trimmed/A1_R1.pair.fastq trimmed/A1_R1.single.fastq \
  trimmed/A1_R2.pair.fastq trimmed/A1_R2.single.fastq \
  ILLUMINACLIP:illuminaClipping_main.fa:2:40:15 \
  LEADING:3 TRAILING:3 \
  SLIDINGWINDOW:4:20 MINLEN:25

-------

sbatch trim.slurm
```

QC 2nd time

create file fastqc2.slurm in your favorite **text** editor

```
-----
#!/bin/bash

#SBATCH -p serial_requeue             # Partition to submit to
#SBATCH -n 1                          # Number of cores
#SBATCH -N 1                          # Ensure that all cores are on one machine
#SBATCH -t 0-0:30                     # Runtime in days-hours:minutes
#SBATCH --mem 500                     # Memory in MB
#SBATCH –J A1_fastqc2                 # job name
#SBATCH -o A1_fastqc2.out             # File to which standard out will be written
#SBATCH –e A1_fastqc2.err             # File to which standard err will be written
#SBATCH --mail-type=ALL               # Type of email notification- BEGIN,END,FAIL,ALL
#SBATCH --mail-user=name@harvard.edu  # Email to which notifications will be sent

source new-modules.sh; module load fastqc

# and now QC
fastqc -o fastqc_reports trimmed/A1_R1.pair.fastq
fastqc -o fastqc_reports trimmed/A1_R2.pair.fastq
-------

sbatch fastqc2.slurm
```

Blacklist: Remove unwanted reads (e.g. rRNA, spike-ins)

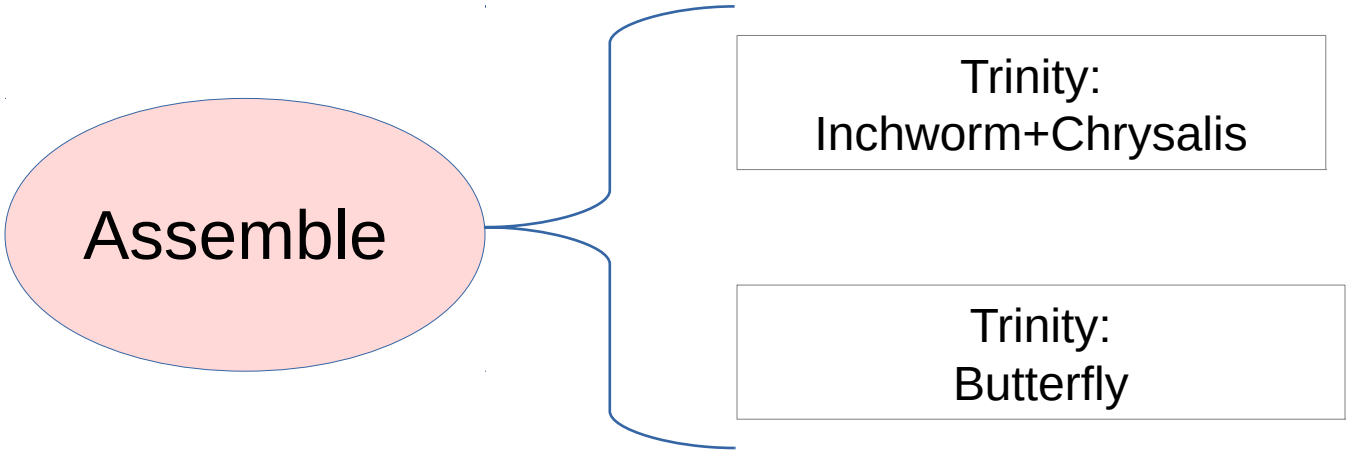
```
download/create FASTA file with blacklist reference sequences
create file blacklist.slurm in your favorite text editor
(may need to be done separately for singles & pairs)

-----
#!/bin/bash

#SBATCH -p serial_requeue                # Partition to submit to
#SBATCH -n 4                             # Number of cores
#SBATCH -N 1                             # Ensure that all cores are on one machine
#SBATCH -t 1-0:00                        # Runtime in days-hours:minutes
#SBATCH --mem 2000                       # Memory in MB
#SBATCH –J A1_blacklist                  # job name
#SBATCH -o A1_blacklist.out              # File to which standard out will be written
#SBATCH –e A1_blacklist.err              # File to which standard err will be written
#SBATCH --mail-type=ALL                  # Type of email notification- BEGIN,END,FAIL,ALL
#SBATCH --mail-user=name@harvard.edu     # Email to which notifications will be sent

source new-modules.sh; module load bowtie

#index for bowtie
bowtie-build blacklist_seqs.fasta blacklist_seqs

# blacklist
mkdir blacklisted cleaned
bowtie blacklist_seqs -q \
  -1 trimmed/A1_R1.pair.fastq -2 trimmed/A1_R2.pair.fastq \
  --al blacklisted/A1_aligned.fastq \
  --un cleaned/A1_unaligned.fastq \
  --max blacklisted/A1_max.fastq \
  -p 4 \
  > blacklisted/A1_bowtie_out.txt
-------

sbatch blacklist.slurm
```

in silico normalize

may need to run this on bigmem partition if tons of reads (here assuming 150m reads)
make list files if more than 1 sample to normalize/combine
create file normalize.slurm in your favorite **text** editor

```
-----
#!/bin/bash

#SBATCH -p general                      # Partition to submit to
#SBATCH –n 8                            # Number of cores
#SBATCH -N 1                            # Ensure that all cores are on one machine
#SBATCH –t 3-0:00                       # Runtime in days-hours:minutes
#SBATCH --mem 155000                    # Memory in MB
#SBATCH –J A1_normalize                 # job name
#SBATCH -o A1_normalize.out             # File to which standard out will be written
#SBATCH –e A1_normalize.err             # File to which standard err will be written
#SBATCH --mail-type=ALL                 # Type of email notification- BEGIN,END,FAIL,ALL
#SBATCH --mail-user=name@harvard.edu    # Email to which notifications will be sent

source new-modules.sh; module load trinityrnaseq

# rename and compress again
mkdir trinity
mv cleaned/A1_unaligned_1.fastq cleaned/A1_R1.pair.clean.fastq
gzip cleaned/A1_R1.pair.clean.fastq
echo “/fullpath_to_file/cleaned/A1_R1.pair.clean.fastq.gz” > trinity/left_list.txt

mv cleaned/A1_unaligned_2.fastq cleaned/A1_R2.pair.clean.fastq
gzip cleaned/A1_R2.pair.clean.fastq
echo “/fullpath_to_file/cleaned/A1_R2.pair.clean.fastq.gz” > trinity/right_list.txt

# normalize
$TRINITY_RNASEQ_ROOT/util/insilico_read_normalization.pl \
  --seqType fq \
  --JM 150G \
  --max_cov 30 \
  --left_list left_list.txt --right_list right_list.txt \
  --pairs_together \
  --PARALLEL_STATS --CPU 8
-------

sbatch normalize.slurm
```

Transcriptome Assembly



Trinity: Inchworm + Chrysalis

create file trinity_ic.slurm in your favorite **text** editor
you may need to run this on the bigmem partition

```
-----
#!/bin/bash

#SBATCH -p general                      # Partition to submit to
#SBATCH –n 16                           # Number of cores
#SBATCH -N 1                            # Ensure that all cores are on one machine
#SBATCH –t 3-0:00                       # Runtime in days-hours:minutes
#SBATCH --mem 155000                    # Memory in MB
#SBATCH –J trinity_ic                   # job name
#SBATCH -o A1_trinity_ic.out            # File to which standard out will be written
#SBATCH –e A1_trinity_ic.err            # File to which standard err will be written
#SBATCH --mail-type=ALL                 # Type of email notification- BEGIN,END,FAIL,ALL
#SBATCH --mail-user=name@harvard.edu    # Email to which notifications will be sent

source new-modules.sh; module load trinityrnaseq

# cat R1 singles onto L pair; and R2 singles onto R pair
gunzip trimmed/A1_R1.single.fastq.gz; gunzip trimmed/A1_R2.single.fastq.gz;
cat trinity/R1_normalized.fq trimmed/A1_R1.single.fastq > trinity/A1_R1.p+s.clean.norm.fastq
cat trinity/R2_normalized.fq trimmed/A1_R2.single.fastq > trinity/A1_R2.p+s.clean.norm.fastq

Trinity --seqType fq \
   --JM 150G \
   --left trinity/A1_R1.p+s.clean.norm.fastq --right trinity/A1_R2.p+s.clean.norm.fastq \
   --SS_lib_type FR \
   --output trinity_output \
   --CPU 16 \
   --min_kmer_cov 2 \
   --max_reads_per_loop 5000000 \
   --group_pairs_distance 800 \
   --no_butterfly
-------

sbatch trininty_ic.slurm
```

*Submitted batch job **22855027***

Trinity: Butterfly

you may need to run this on the bigmem partition

create file trinity_SLURM_conf.txt in your favorite **text** editor

```
-----
#-----------------------------------------------------------------------------------------------
# grid type:
grid=SLURM
# template for a grid submission
cmd=sbatch -p serial_requeue --mem=10000 --time=02:00:00
# number of grid submissions to be maintained at steady state by the Trinity submission system
max_nodes=1000
# number of commands that are batched into a single grid submission job.
cmds_per_node=60
#-----------------------------------------------------------------------------------------------
-----
```

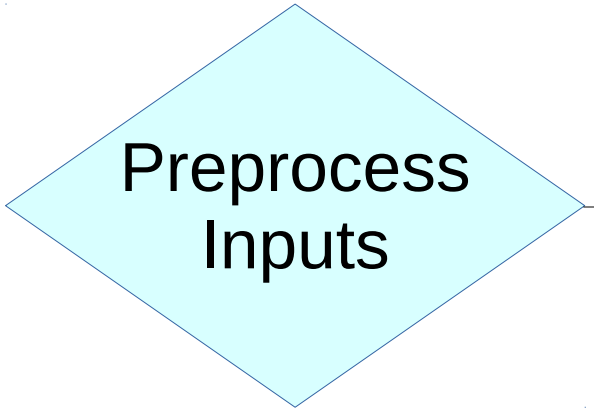
create file trinity_b.slurm in your favorite **text** editor

```
-----
#!/bin/bash

#SBATCH -p general                      # Partition to submit to
#SBATCH –n 1                            # Number of cores
#SBATCH -N 1                            # Ensure that all cores are on one machine
#SBATCH –t 1-0:00                       # Runtime in days-hours:minutes
#SBATCH --mem 4000                      # Memory in MB
#SBATCH –J trinity_b                    # job name
#SBATCH -o A1_trinity_b.out             # File to which standard out will be written
#SBATCH –e A1_trinity_b.err             # File to which standard err will be written
#SBATCH --mail-type=ALL                 # Type of email notification- BEGIN,END,FAIL,ALL
#SBATCH --mail-user=name@harvard.edu    # Email to which notifications will be sent

source new-modules.sh; module load trinityrnaseq

Trinity --seqType fq \
  --left trinity/A1_R1.p+s.clean.norm.fastq --right trinity/A1_R2.p+s.clean.norm.fastq \
  --SS_lib_type FR \
  --output trinity_output \
  --grid_conf trinity_SLURM_conf.txt

-------

sbatch --dependency=afterok:22855027 trininty_b.slurm
```

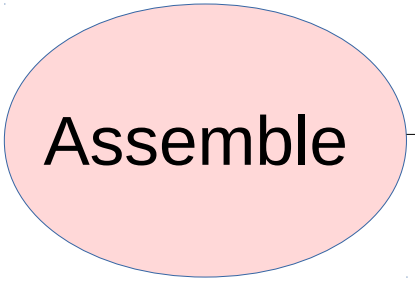
Transcriptome Assembly – QC

- Conceptually similar to genome QC with a few catches
- Assembly completeness:
  - What fraction of assembled transcripts are “full length”?
- Assembly correctness:
  - How to measure error?
- RSEQC: package for RNA-seq QC with lots of potentially useful tools

Transcriptome Assembly: Summary



Concatenate reads
Trim and QC
Blacklist
In silico normalize



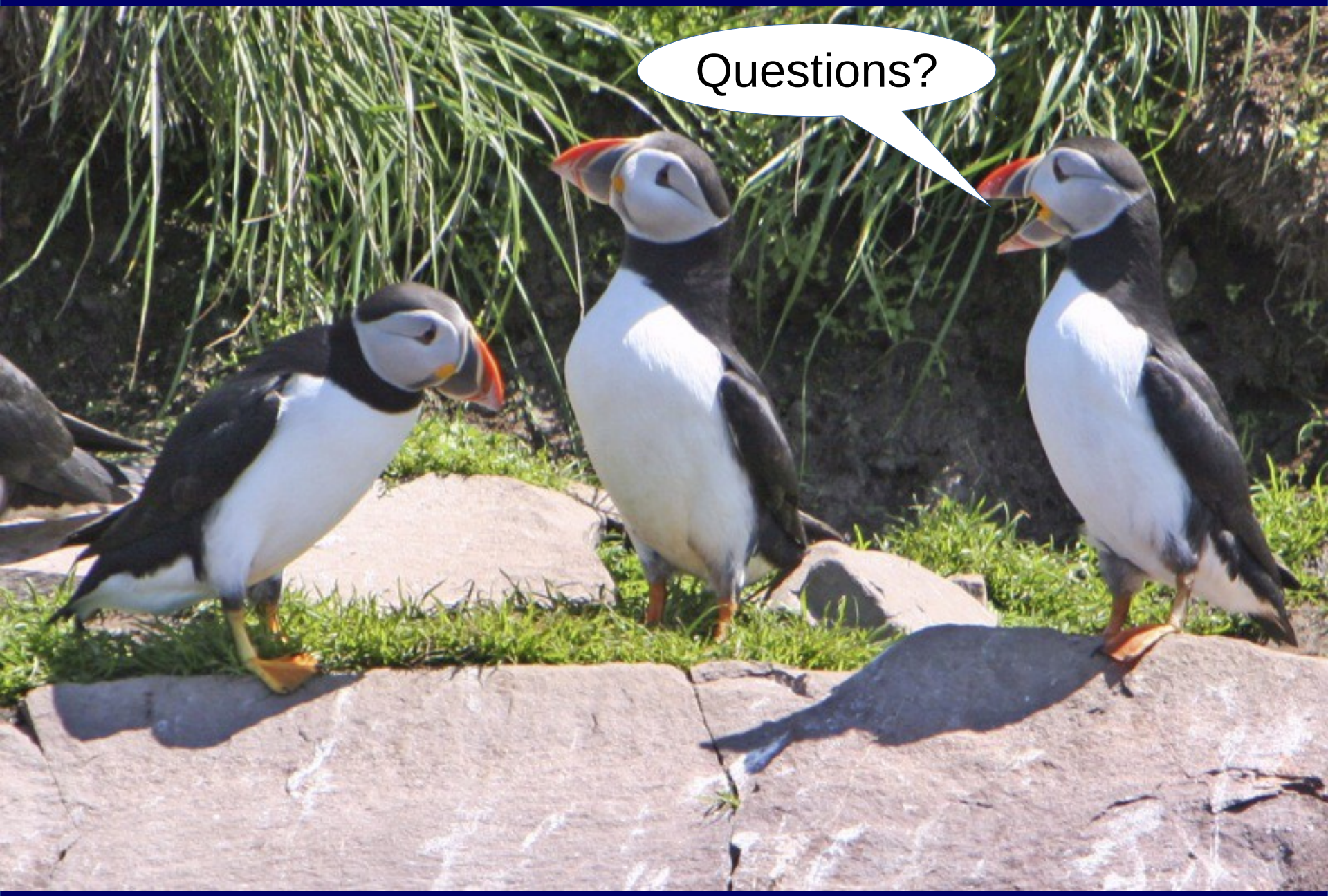
Trinity (*de novo*)
Reference-based assemblies:
(Trinity genome-guided)
(Cufflinks)

Mapping reads to assembly always useful
BUSCO-type approaches can help
Not many good prepackaged solutions
RSEQC: http://rseqc.sourceforge.net/

More code examples and best practices coming soon to the informatics website: http://informatics.fas.harvard.edu/

RC/Informatics Resources

- Training and courses
  - https://rc.fas.harvard.edu/education/training/
  - Next Intro to Odyssey class is Wed Nov 12th at 9:00 AM. RSVP: http://goo.gl/FQ6z0T
- Weekly RC office hours for cluster-related questions
  - Wednesdays 12:00 – 3:00 pm, 38 Oxford St Room 206
- Informatics Online Tutorials:
  - http://informatics.fas.harvard.edu/


Questions?