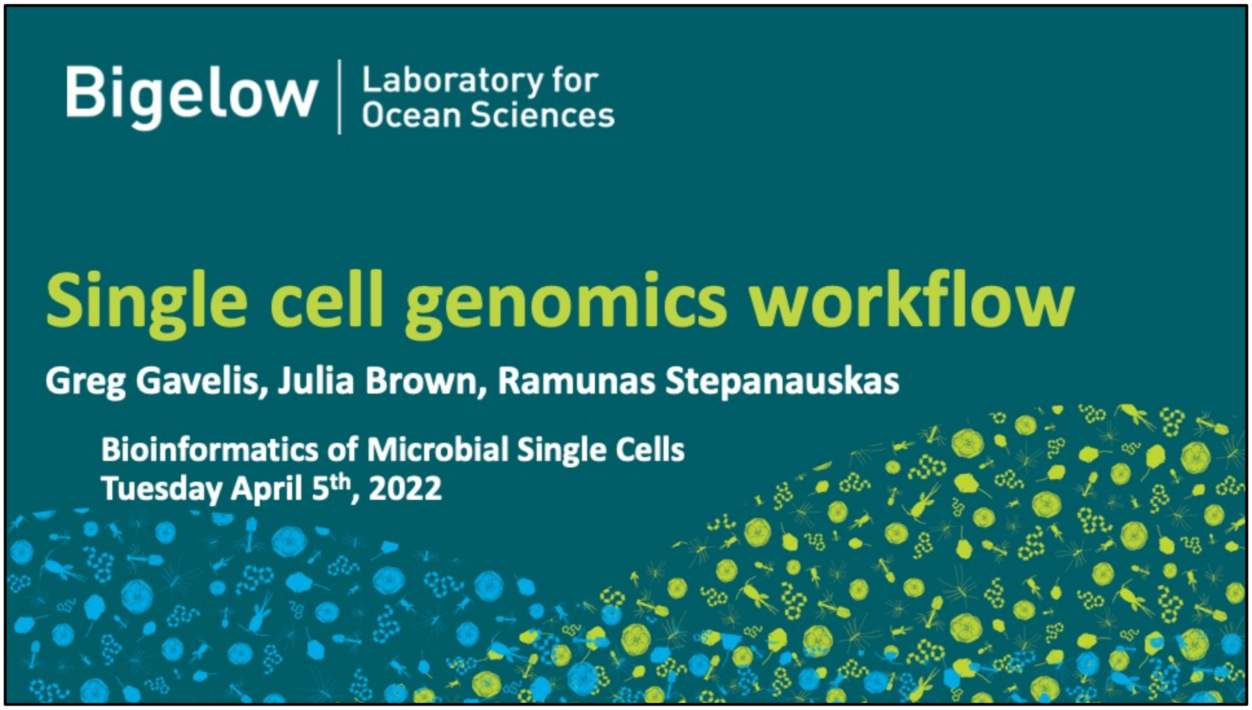
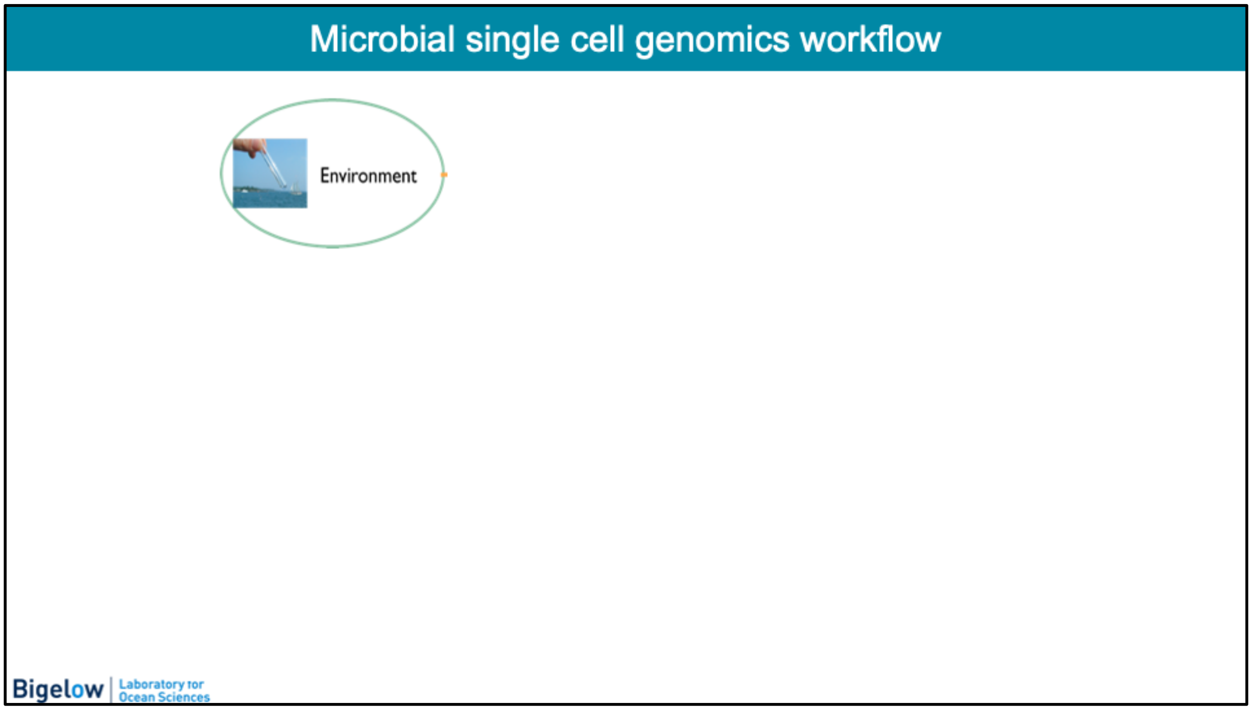


Bigelow | Laboratory for Ocean Sciences

Single cell genomics workflow

Greg Gavelis, Julia Brown, Ramunas Stepanauskas

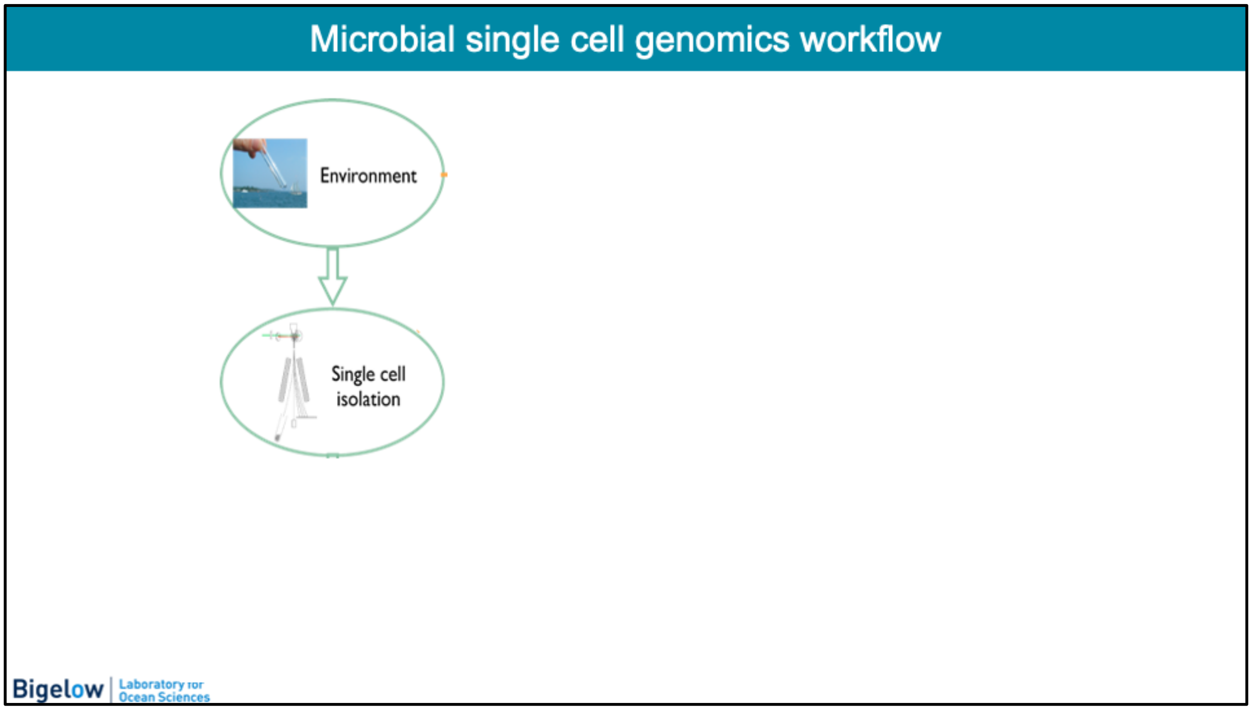
Bioinformatics of Microbial Single Cells
Tuesday April 5th, 2022



Microbial single cell genomics workflow

Bigelow | Laboratory for Ocean Sciences

As Ramunas explained this morning, the SCGC process begins with sample collection and lysis, and results in terabytes of data

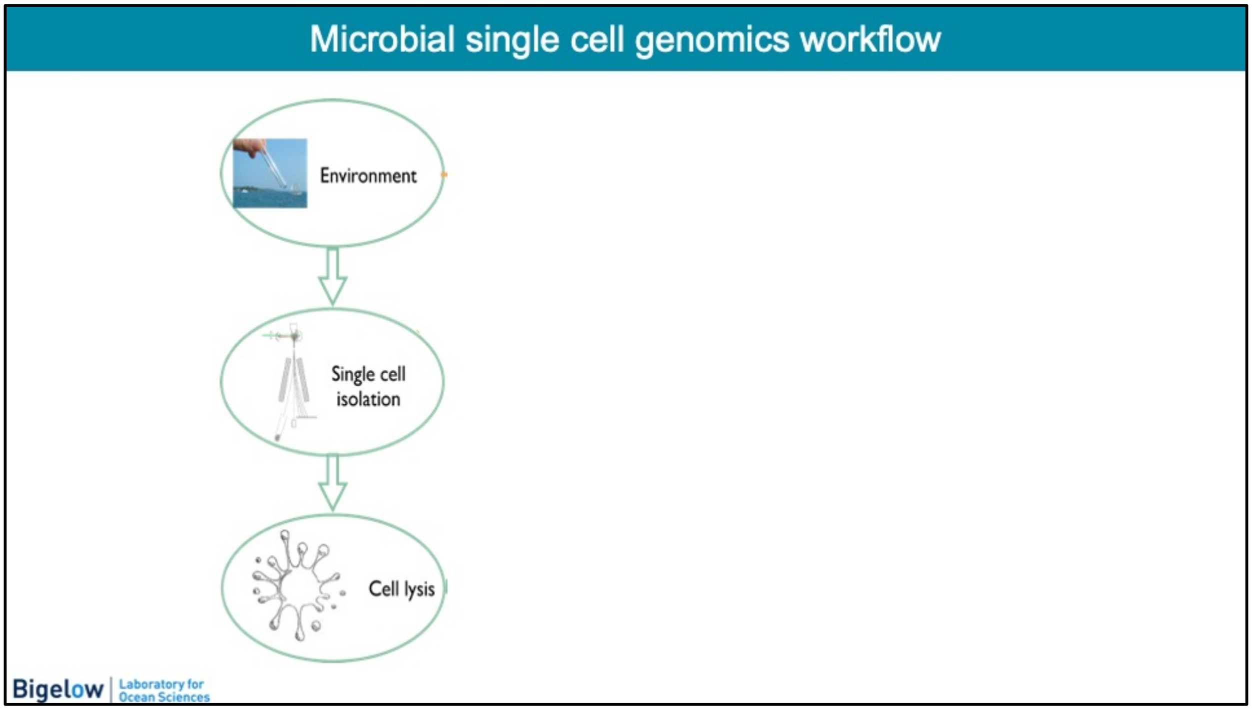


Microbial single cell genomics workflow

Environment

Single cell isolation

As Ramunas explained this morning, the SCGC process begins with sample collection and lysis, and results in terabytes of data



Microbial single cell genomics workflow

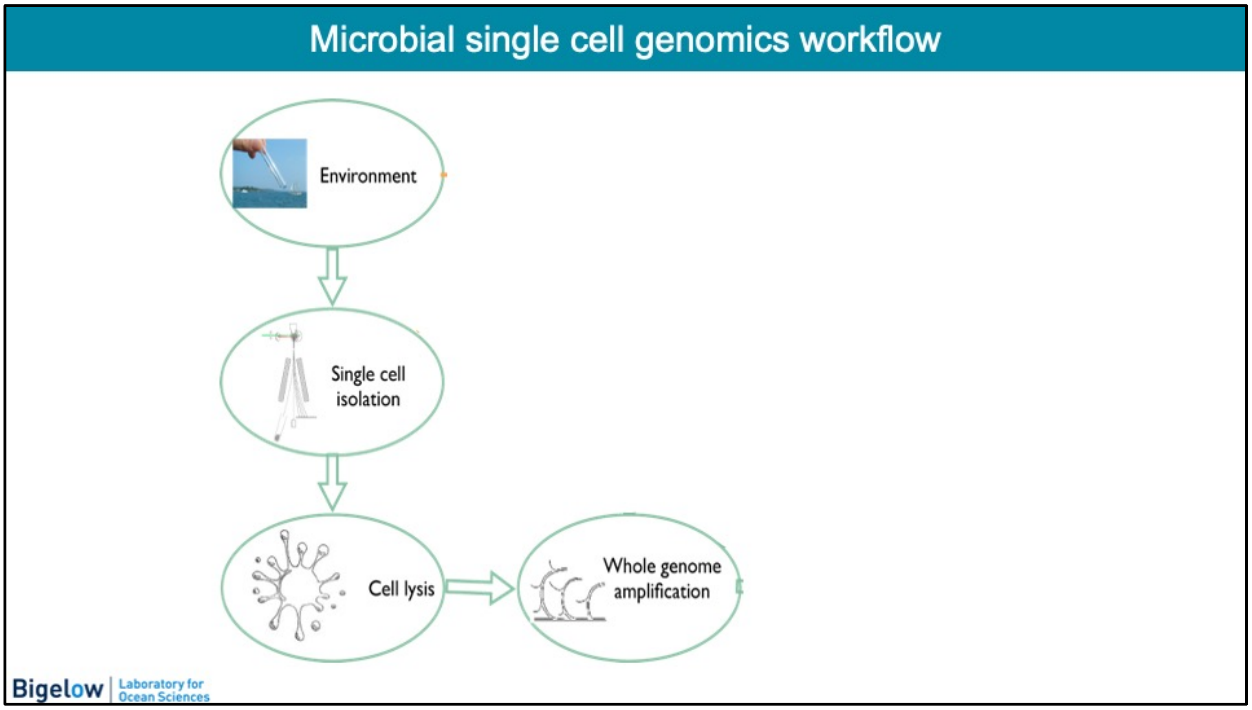
Environment

Single cell isolation

Cell lysis

Bigelow | Laboratory for Ocean Sciences

As Ramunas explained this morning, the SCGC process begins with sample collection and lysis, and results in terabytes of data



Microbial single cell genomics workflow

Environment

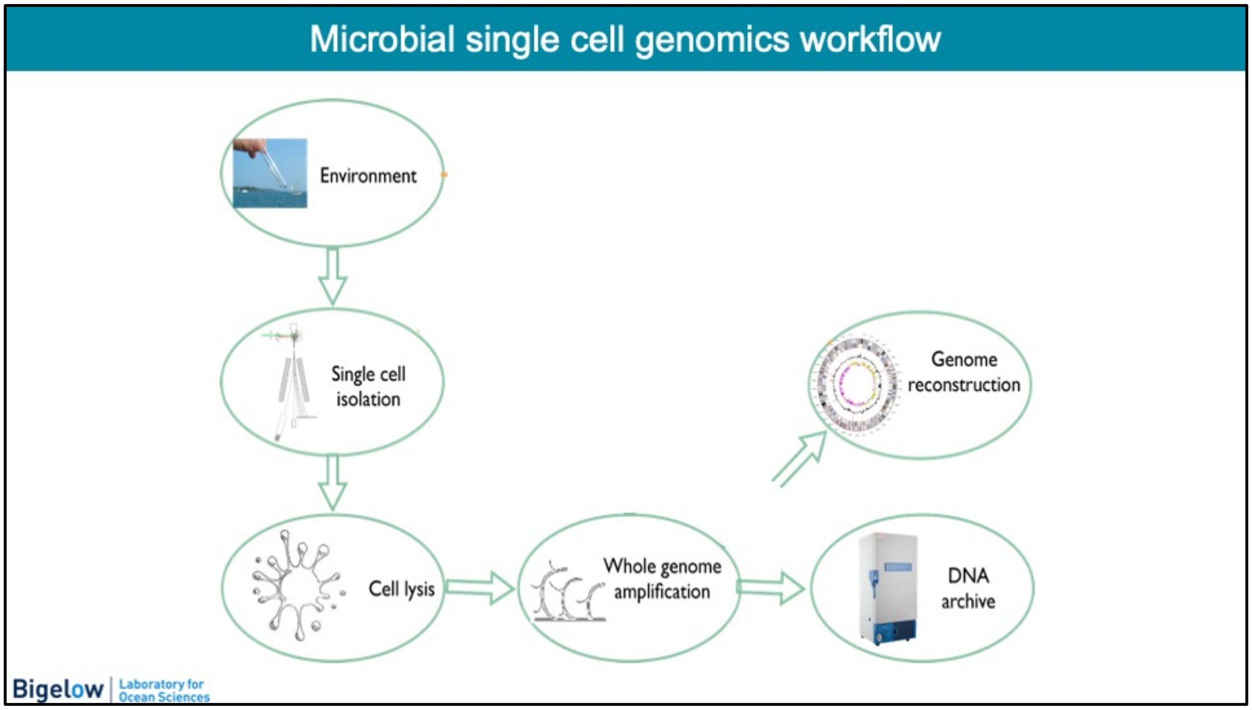
Single cell isolation

Cell lysis

Whole genome amplification

Bigelow | Laboratory for Ocean Sciences

As Ramunas explained this morning, the SCGC process begins with sample collection and lysis, and results in terabytes of data



Microbial single cell genomics workflow

Environment

Single cell isolation

Cell lysis

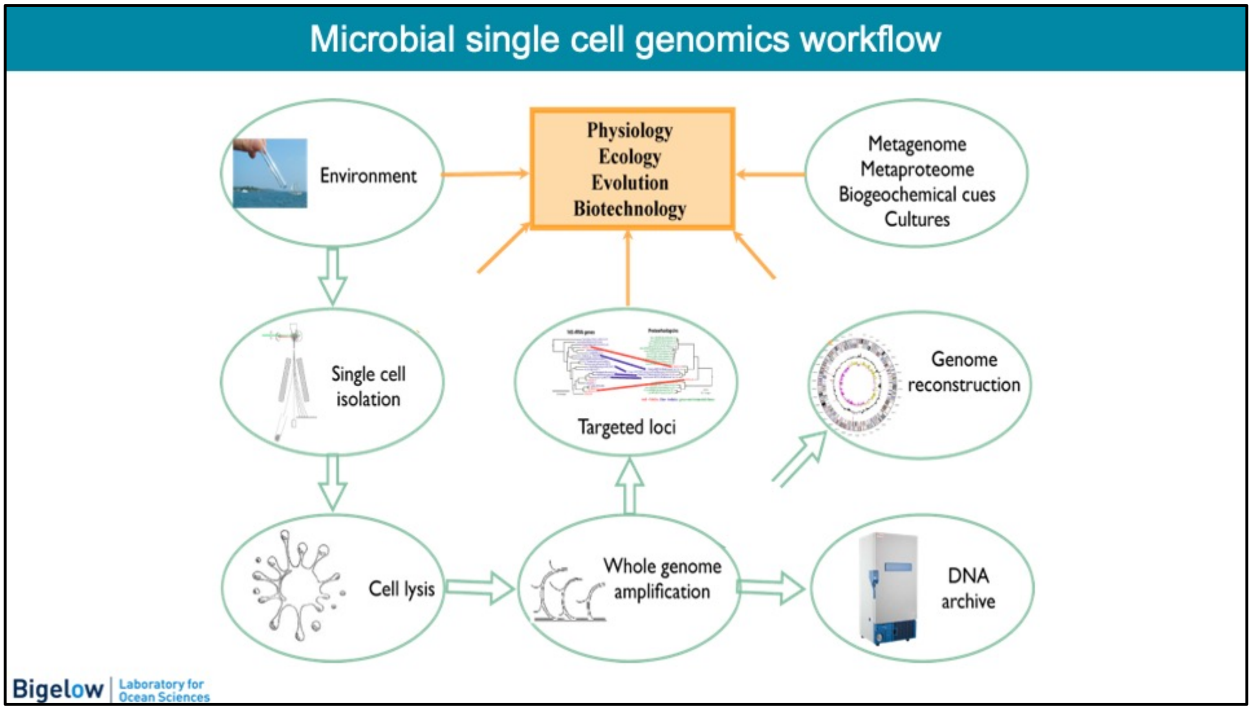
Whole genome amplification

DNA archive

Genome reconstruction

Bigelow | Laboratory for Ocean Sciences

As Ramunas explained this morning, the SCGC process begins with sample collection and lysis, and results in terabytes of data



Microbial single cell genomics workflow

Environment

Physiology
Ecology
Evolution
Biotechnology

Metagenome
Metaproteome
Biogeochemical cues
Cultures

Single cell isolation

Targeted loci

Genome reconstruction

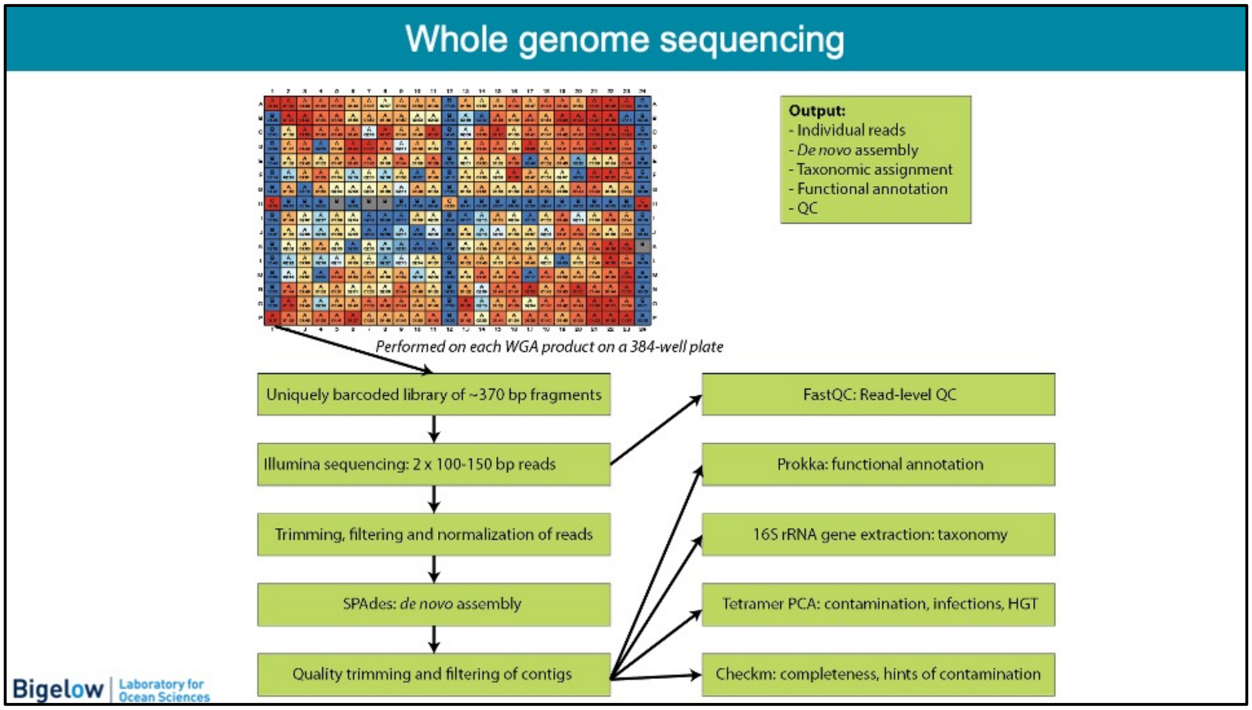
Cell lysis

Whole genome amplification

DNA archive

Bigelow | Laboratory for Ocean Sciences

As Ramunas explained this morning, the SCGC process begins with sample collection and lysis, and results in terabytes of data



Whole genome sequencing

Output:
- Individual reads
- *De novo* assembly
- Taxonomic assignment
- Functional annotation
- QC

Performed on each WGA product on a 384-well plate

Uniquely barcoded library of ~370 bp fragments

Illumina sequencing: 2 x 100-150 bp reads

Trimming, filtering and normalization of reads

SPAdes: *de novo* assembly

Quality trimming and filtering of contigs

FastQC: Read-level QC

Prokka: functional annotation

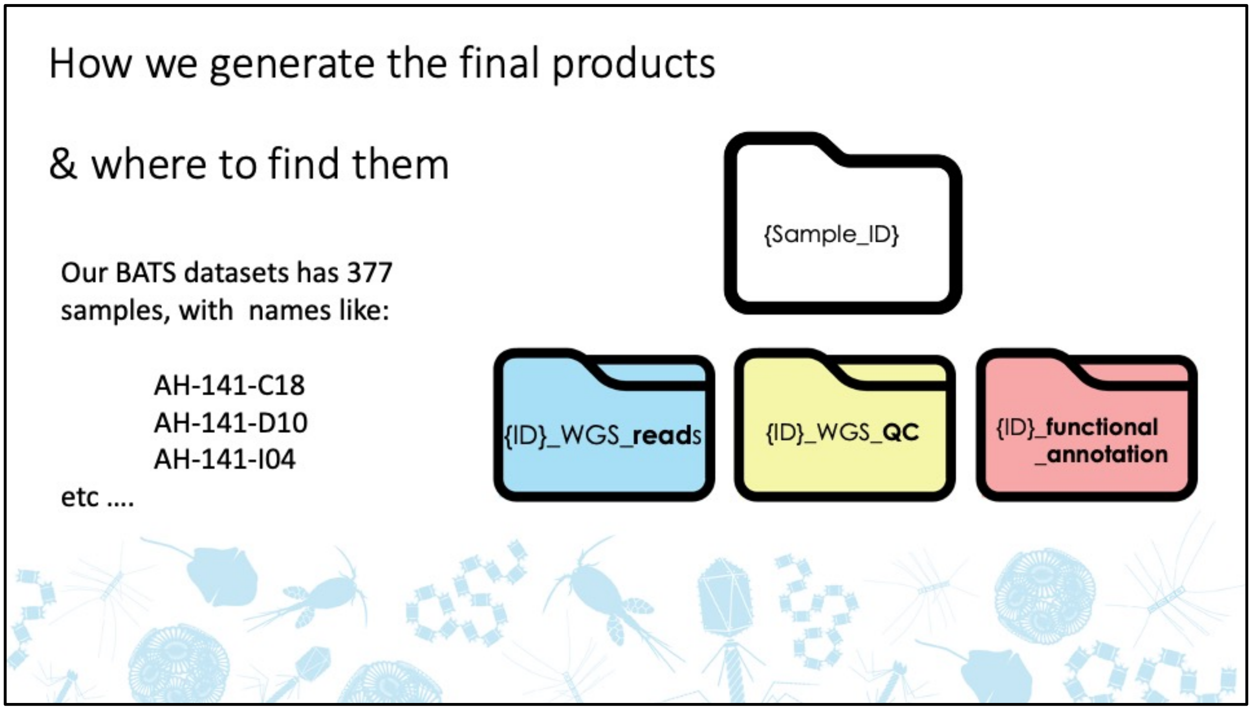
16S rRNA gene extraction: taxonomy

Tetramer PCA: contamination, infections, HGT

Checkm: completeness, hints of contamination

Bigelow | Laboratory for Ocean Sciences

Many bioinformatic steps are involved in cleaning, assembling, and annotating the genomes. This presentation will be a brief overview of the different steps involved. When space allows, I include the actual commands that we use at each step of the pipeline.



How we generate the final products

& where to find them

Our BATS datasets has 377 samples, with names like:

AH-141-C18
AH-141-D10
AH-141-I04
etc ….

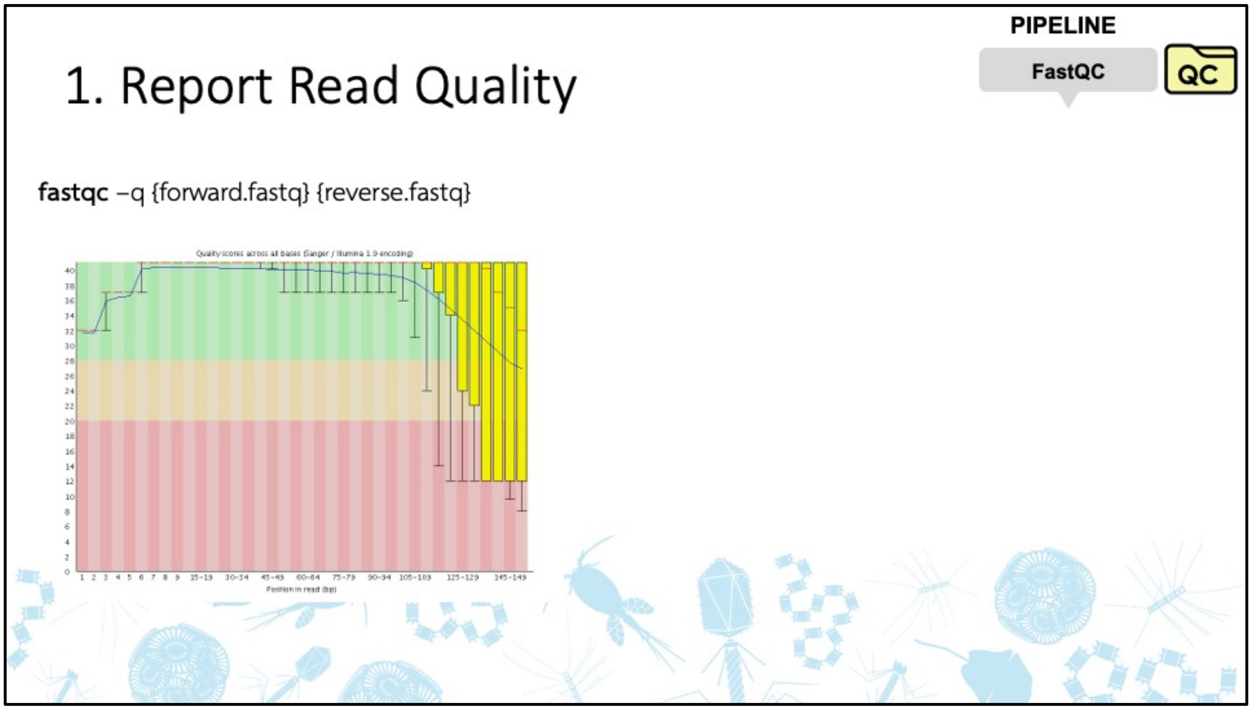
{Sample_ID}

{ID}_WGS_**read**s

{ID}_WGS_**QC**

{ID}_**functional _annotation**

Each sample gets its own namesake folder. That folder contains the final genome assembly (e.g. AH-141-C18_contigs.fasta), and also three directories, where the products of varies pipeline steps are deposited. (Note that we don’t include all of the intermediates from read processing, since fastq files are very large.)



PIPELINE

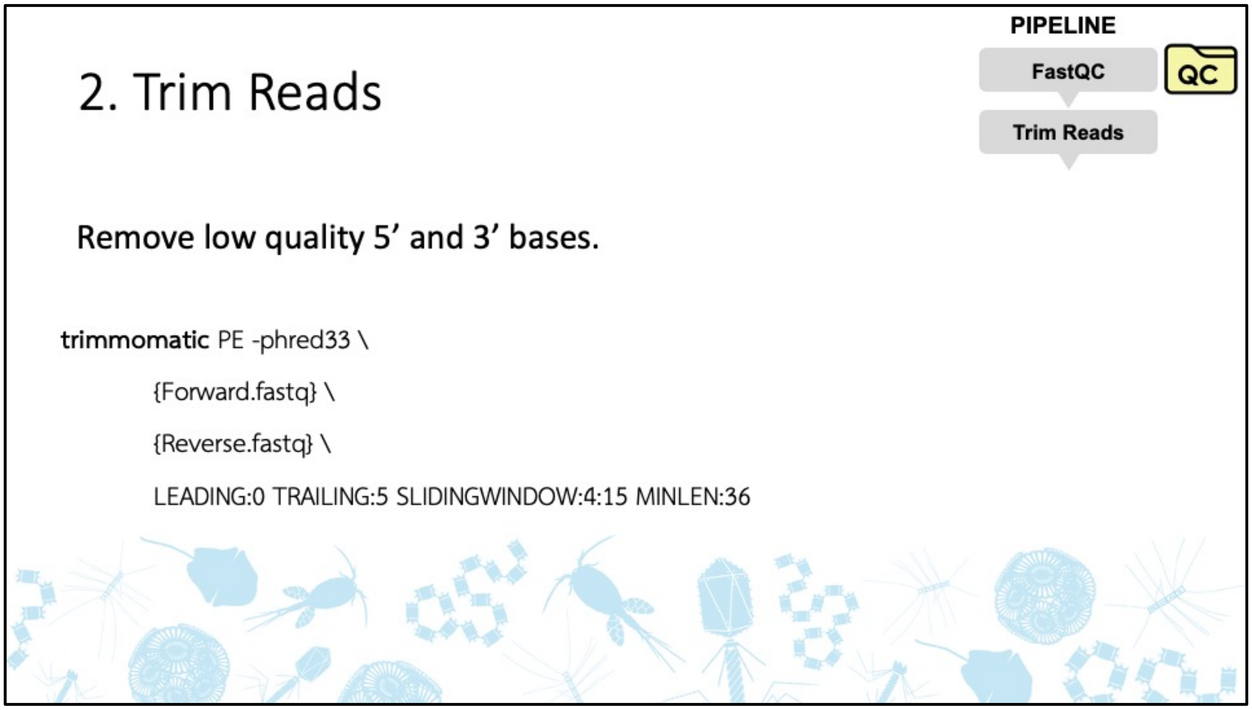
FastQC

QC

1. Report Read Quality

```
fastqc –q {forward.fastq} {reverse.fastq}
```

Before processing, we review the quality of the raw reads, using the program fastqc to output html report files, like above. These are included in the QC subfolder. Note that the quality of the basecalls tends to decline as we reach the end of the 150 bp read.



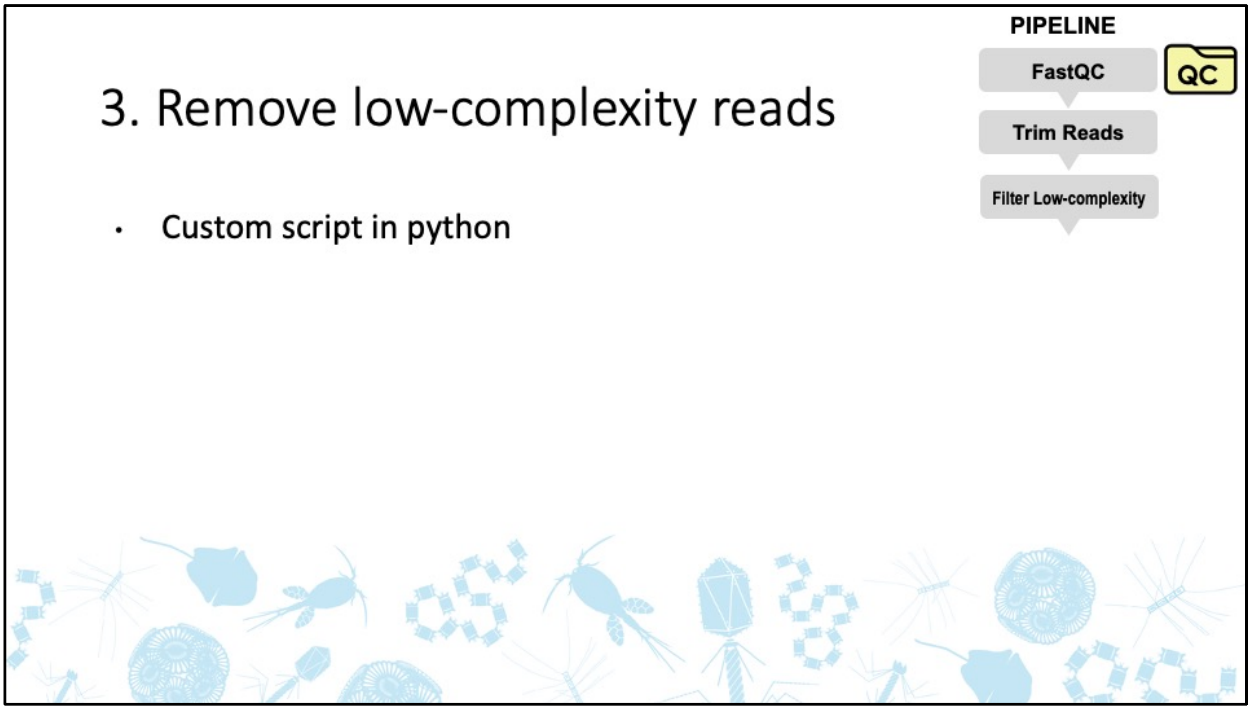
2. Trim Reads

PIPELINE
FastQC
Trim Reads
QC

Remove low quality 5’ and 3’ bases.

```
trimmomatic PE -phred33 \
        {Forward.fastq} \
        {Reverse.fastq} \
        LEADING:0 TRAILING:5 SLIDINGWINDOW:4:15 MINLEN:36
```

For our next step, we use the program trimmomatic to remove these low quality reads. Reads under 36 bp are discarded.



PIPELINE

FastQC

Trim Reads

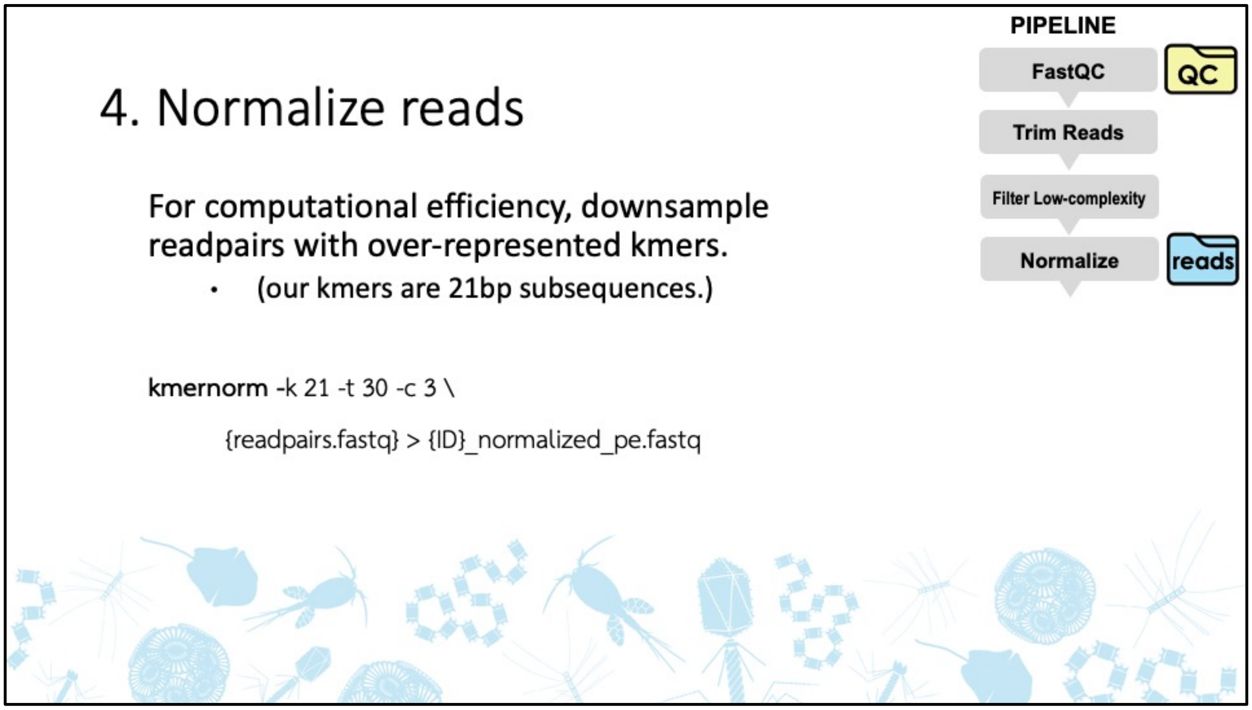
Filter Low-complexity

QC

3. Remove low-complexity reads

- Custom script in python

Step 3, we remove low-complexity reads include (e.g. homopolymer runs ”AAAAAA” and repeats “ATATATA’). If not removed, low complexity sequences can cause misassemblies



PIPELINE

FastQC

QC

Trim Reads

Filter Low-complexity

Normalize

reads

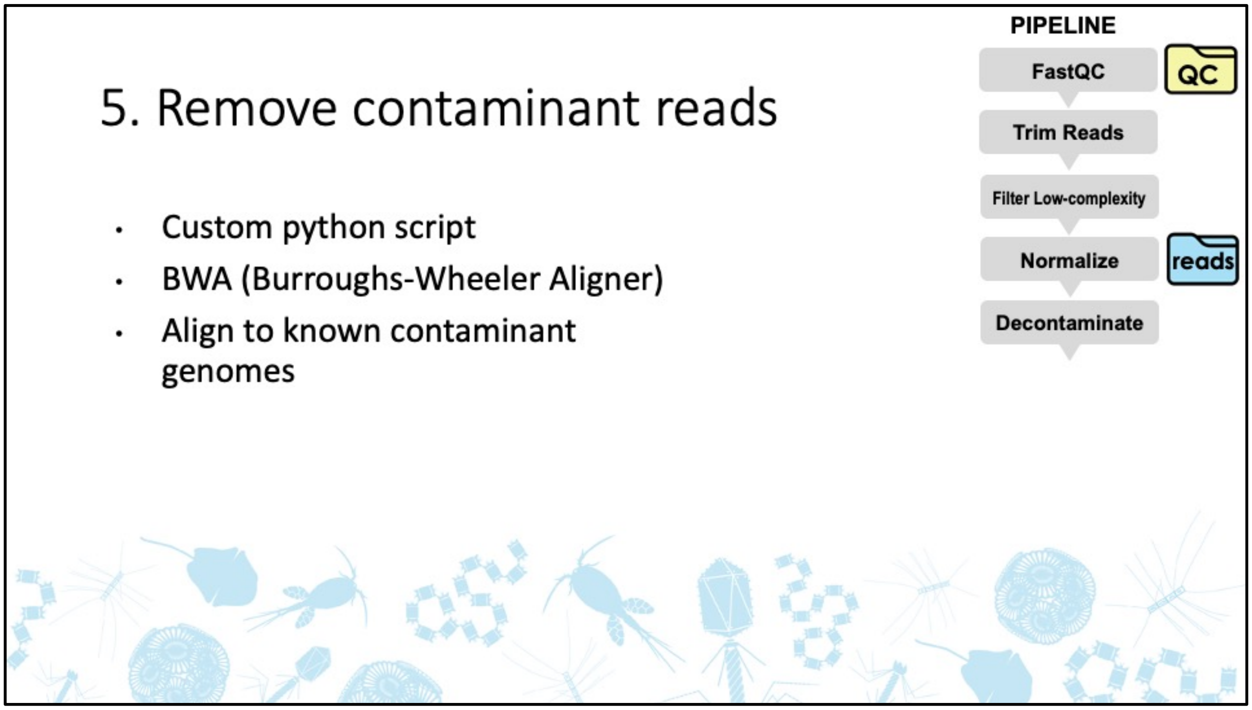
4. Normalize reads

For computational efficiency, downsample readpairs with over-represented kmers.

- (our kmers are 21bp subsequences.)

```
kmernorm -k 21 -t 30 -c 3 \
        {readpairs.fastq} > {ID}_normalized_pe.fastq
```

Kmernorm is much faster than the the read normalization program that comes built into the assembly software. Normalization allows assembly to run ~10X faster and with no significant decline in assembly quality.



5. Remove contaminant reads

- Custom python script
- BWA (Burroughs-Wheeler Aligner)
- Align to known contaminant genomes

PIPELINE

FastQC

QC

Trim Reads

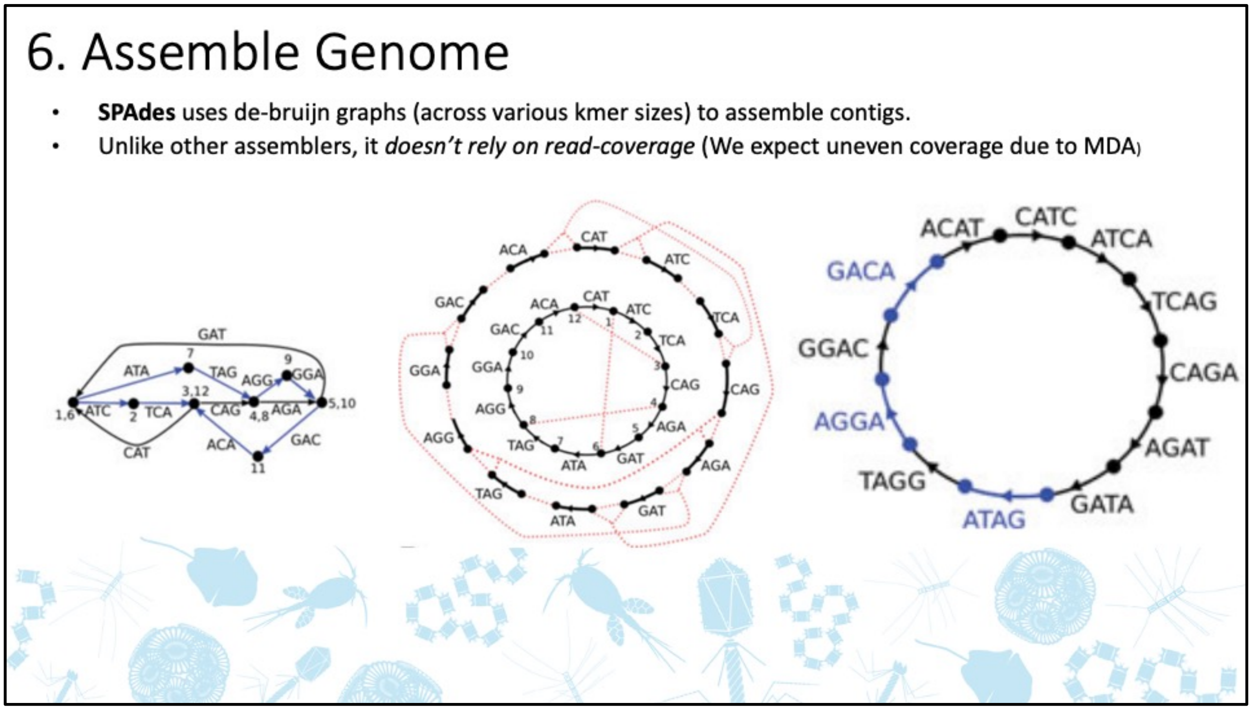
Filter Low-complexity

Normalize

reads

Decontaminate

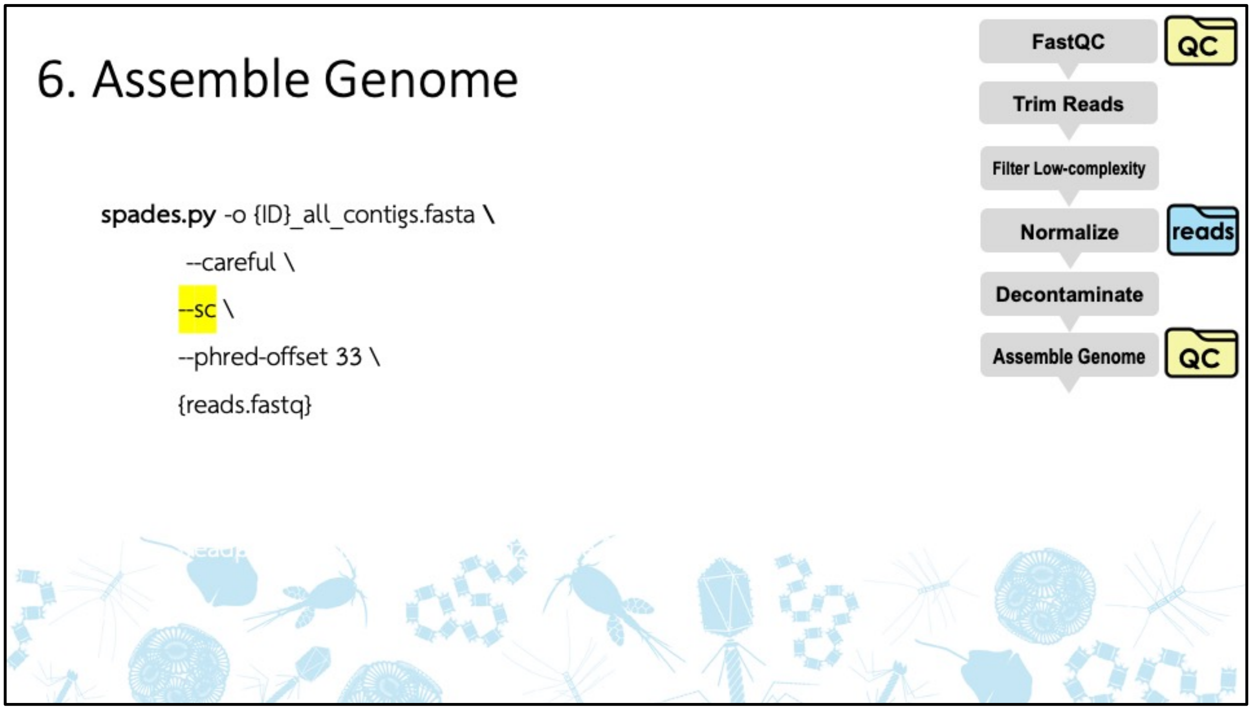
For the first of three decontamination steps, we remove reads that align to nucleotides in a custom database of genomic contaminants. We obtained these contaminants by sequencing ‘empty’ (i.e. negative control) plates in the clean lab.



6. Assemble Genome

- **SPAdes** uses de-bruijn graphs (across various kmer sizes) to assemble contigs.
- Unlike other assemblers, it *doesn't rely on read-coverage* (We expect uneven coverage due to MDA)

SPAdes is unparalleled for the “de-novo” (i.e. reference-free) assembly of DNA obtained from single cells. Critically, it assumes that coverage of input reads will be highly uneven (due to amplification bias introduced during Multiple Displacement Amplificiation).

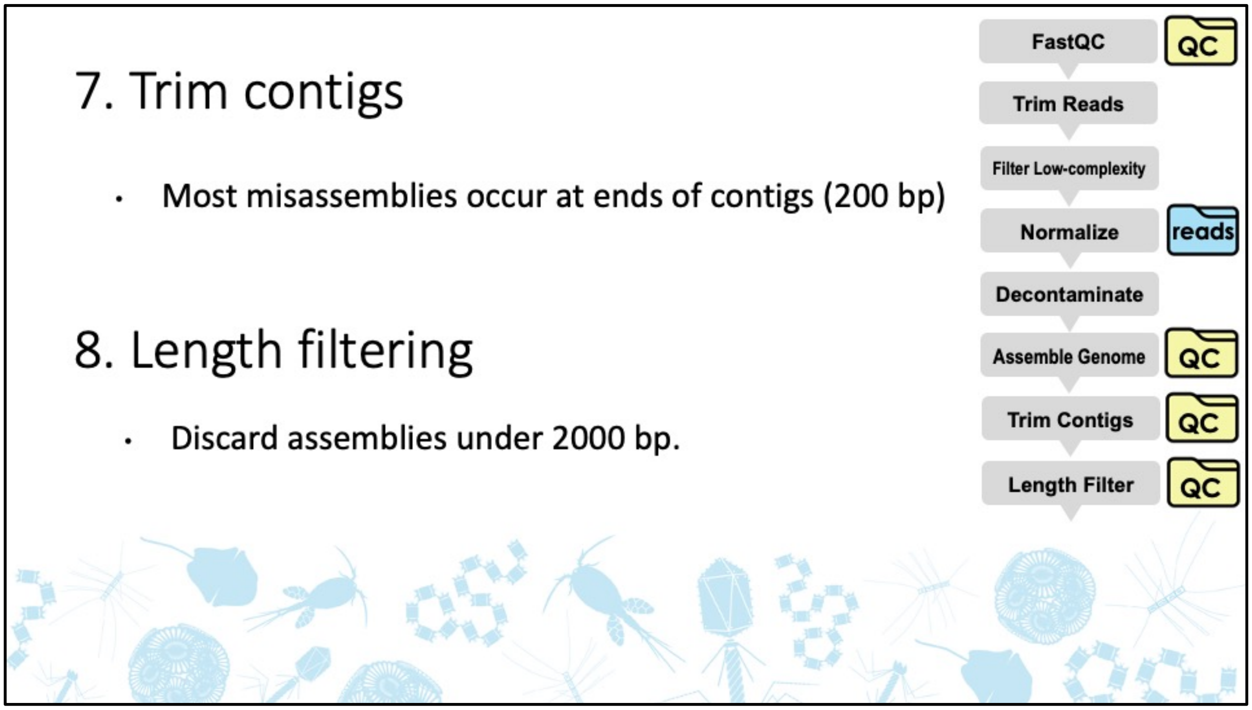


6. Assemble Genome

```
spades.py -o {ID}_all_contigs.fasta \
            --careful \
           --sc \
           --phred-offset 33 \
           {reads.fastq}
```

FastQC
QC
Trim Reads
Filter Low-complexity
Normalize
reads
Decontaminate
Assemble Genome
QC

We make sure to run spades in single-cell mode, so that it does not take read coverage into account. This output assembly is considered raw and likely has some areas that need to be processed. The next several slides deal with matters of QC, that have been carefully optimized.



7. Trim contigs

- Most misassemblies occur at ends of contigs (200 bp)

8. Length filtering

- Discard assemblies under 2000 bp.

FastQC — QC

Trim Reads

Filter Low-complexity

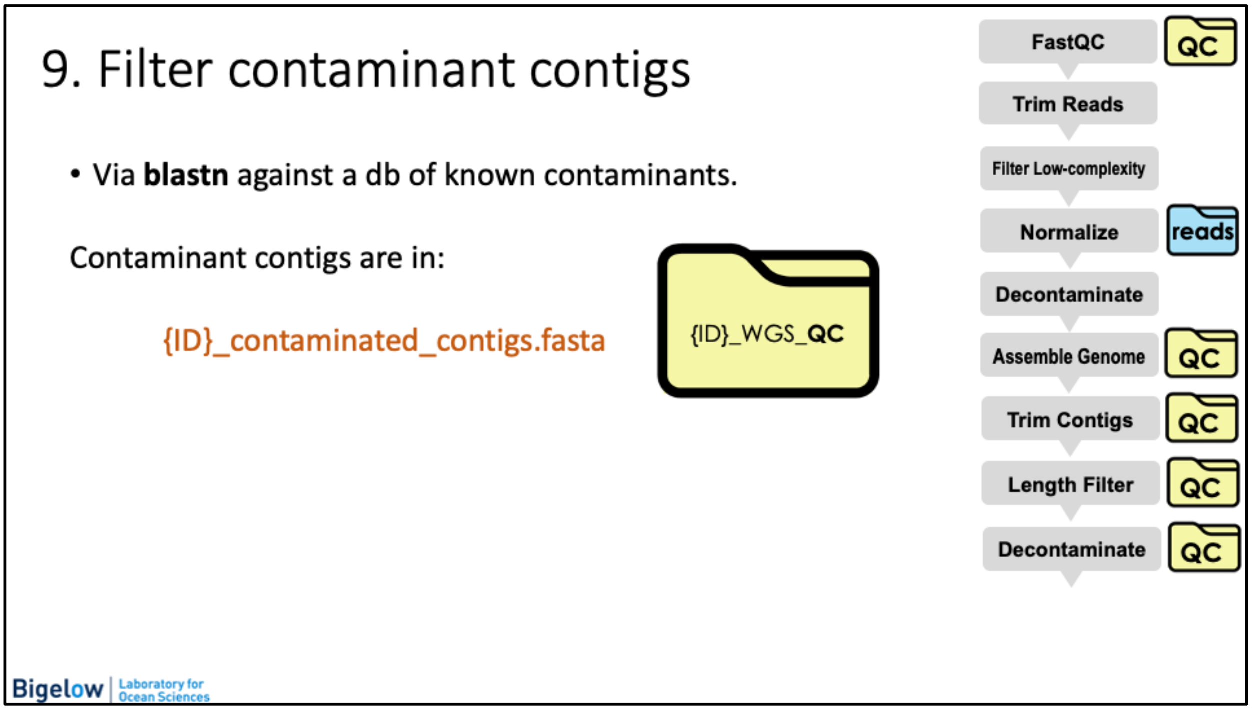
Normalize — reads

Decontaminate

Assemble Genome — QC

Trim Contigs — QC

Length Filter — QC



9. Filter contaminant contigs

- Via **blastn** against a db of known contaminants.

Contaminant contigs are in:

{ID}_contaminated_contigs.fasta

{ID}_WGS_**QC**

FastQC — QC

Trim Reads

Filter Low-complexity

Normalize — reads

Decontaminate

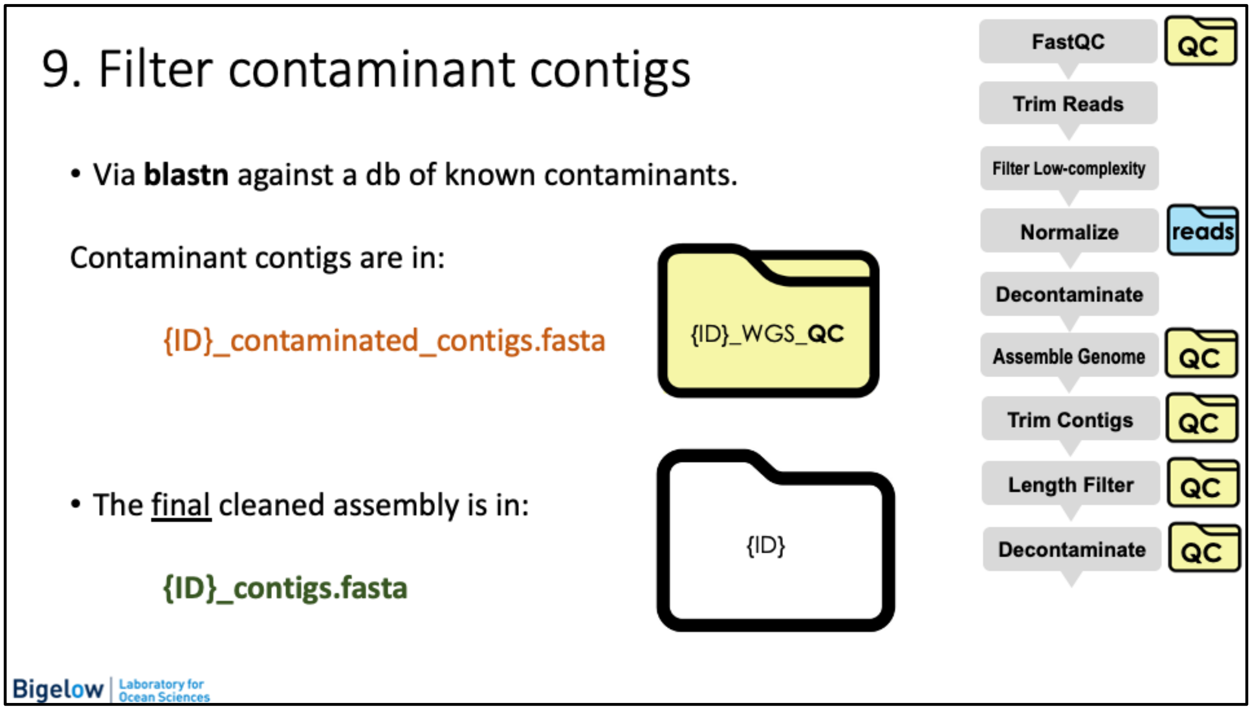
Assemble Genome — QC

Trim Contigs — QC

Length Filter — QC

Decontaminate — QC

Again, sample nucleotides are aligned to known contaminants (this time with blastn.) All these potential contaminants are retained for the user to review, if desired.



9. Filter contaminant contigs

- Via **blastn** against a db of known contaminants.

Contaminant contigs are in:

{ID}_contaminated_contigs.fasta

{ID}_WGS_**QC**

- The final cleaned assembly is in:

{ID}_contigs.fasta

{ID}

FastQC | QC

Trim Reads

Filter Low-complexity

Normalize | reads

Decontaminate

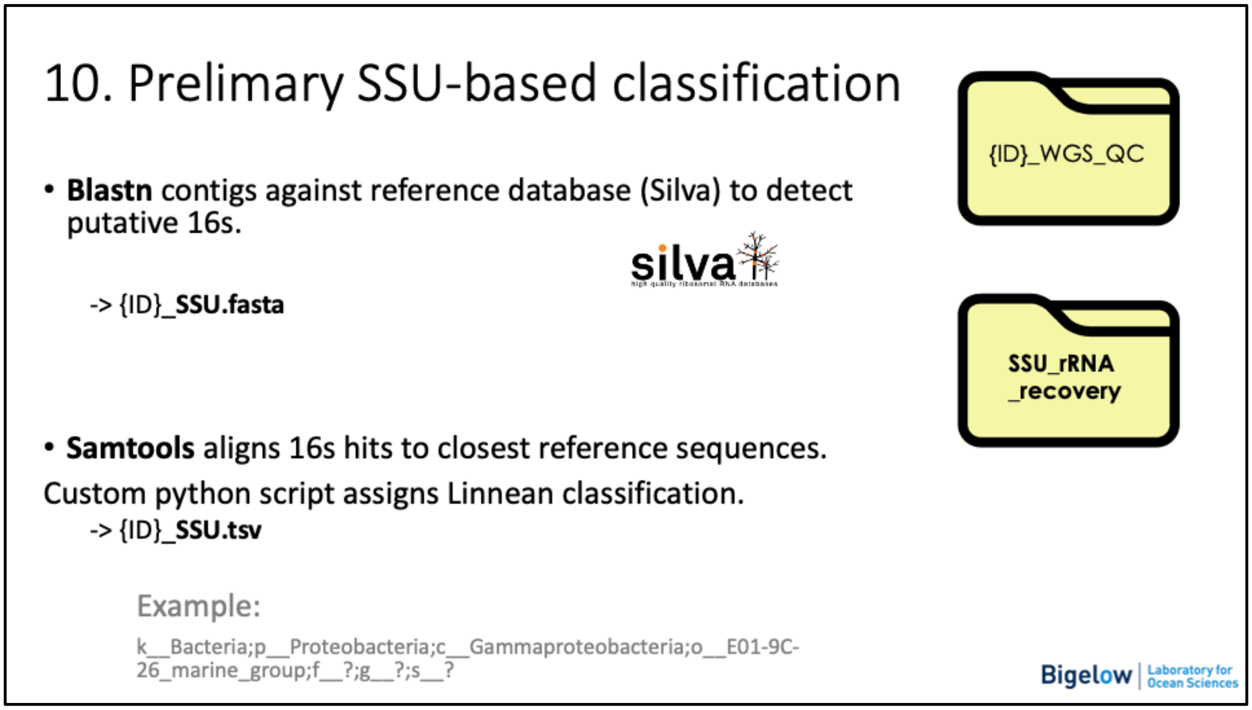
Assemble Genome | QC

Trim Contigs | QC

Length Filter | QC

Decontaminate | QC

Bigelow | Laboratory for Ocean Sciences



10. Prelimary SSU-based classification

- **Blastn** contigs against reference database (Silva) to detect putative 16s.

  -> {ID}**_SSU.fasta**

- **Samtools** aligns 16s hits to closest reference sequences.

Custom python script assigns Linnean classification.

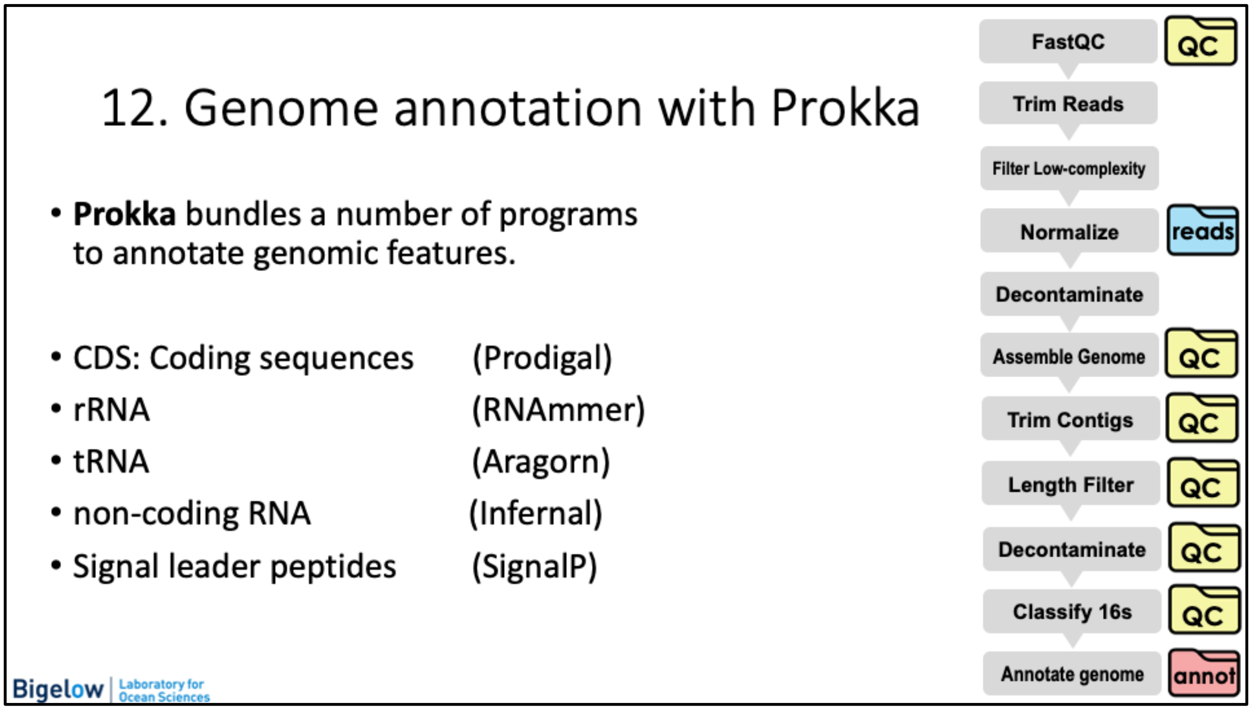
  -> {ID}**_SSU.tsv**

Example:

k__Bacteria;p__Proteobacteria;c__Gammaproteobacteria;o__E01-9C-26_marine_group;f__?;g__?;s__?

{ID}_WGS_QC
SSU_rRNA _recovery
silva
high quality ribosomal RNA databases
Bigelow | Laboratory for Ocean Sciences

This is only one classification approach. In coming lectures we'll talk about how multigene approaches are also used.



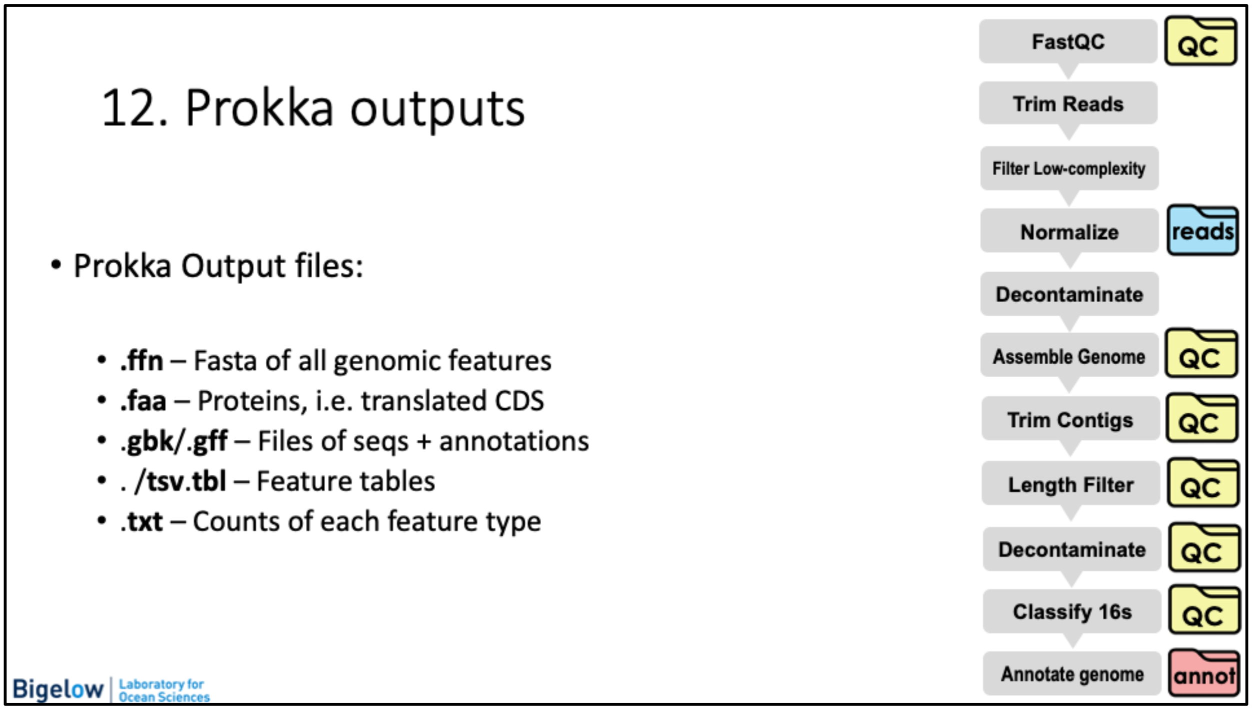
12. Genome annotation with Prokka

- **Prokka** bundles a number of programs to annotate genomic features.

• CDS: Coding sequences	(Prodigal)
• rRNA	(RNAmmer)
• tRNA	(Aragorn)
• non-coding RNA	(Infernal)
• Signal leader peptides	(SignalP)

FastQC
QC
Trim Reads
Filter Low-complexity
Normalize
reads
Decontaminate
Assemble Genome
QC
Trim Contigs
QC
Length Filter
QC
Decontaminate
QC
Classify 16s
QC
Annotate genome
annot
Bigelow Laboratory for Ocean Sciences

In coming lectures, we'll also discuss other annotation programs, like DRAM.



12. Prokka outputs

- Prokka Output files:
  - **.ffn** – Fasta of all genomic features
  - **.faa** – Proteins, i.e. translated CDS
  - **.gbk**/**.gff** – Files of seqs + annotations
  - . /**tsv.tbl** – Feature tables
  - **.txt** – Counts of each feature type

FastQC QC

Trim Reads

Filter Low-complexity

Normalize reads

Decontaminate

Assemble Genome QC

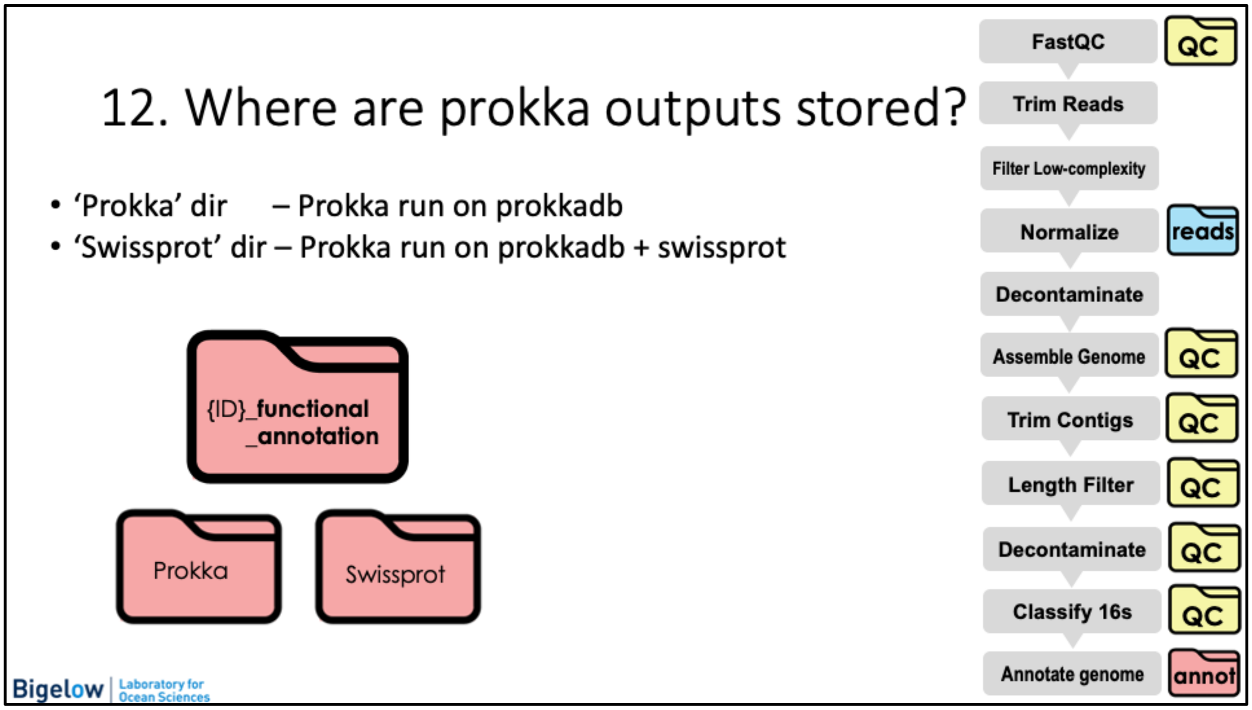
Trim Contigs QC

Length Filter QC

Decontaminate QC

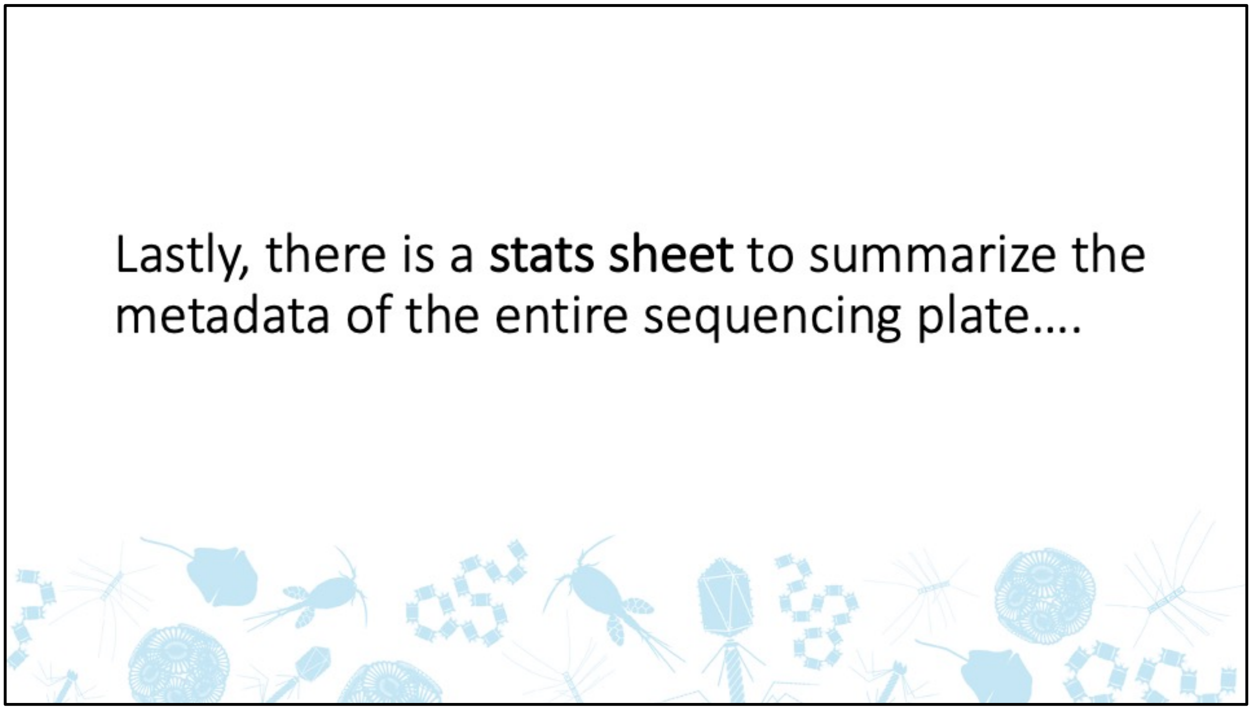
Classify 16s QC

Annotate genome annot



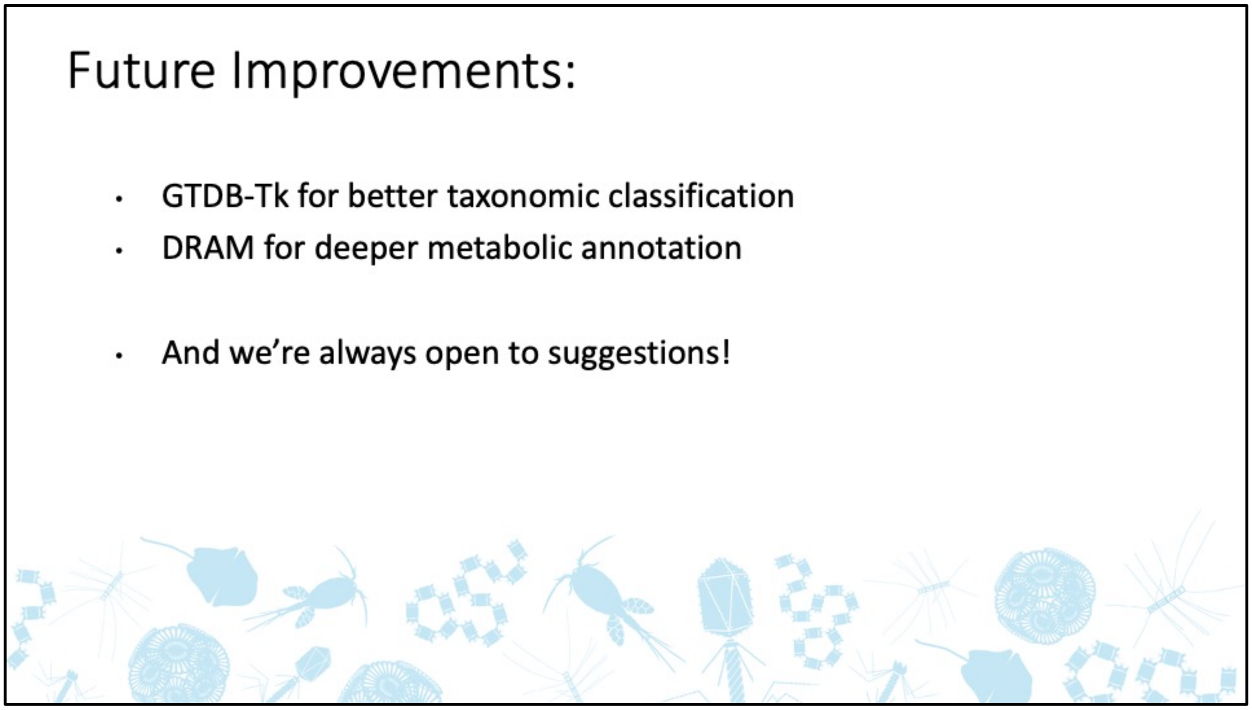
12. Where are prokka outputs stored?

- ‘Prokka’ dir – Prokka run on prokkadb
- ‘Swissprot’ dir – Prokka run on prokkadb + swissprot



Lastly, there is a **stats sheet** to summarize the metadata of the entire sequencing plate….

You'll take a look at that sheet this afternoon, during the interactive session.



Future Improvements:

- GTDB-Tk for better taxonomic classification
- DRAM for deeper metabolic annotation

- And we’re always open to suggestions!