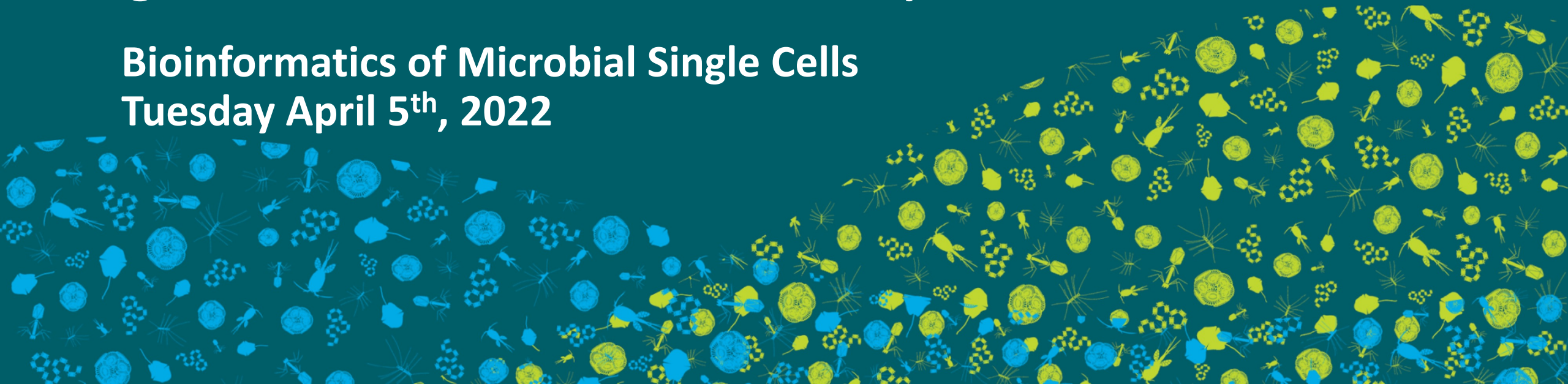
Bigelow | Laboratory for Ocean Sciences

Single cell genomics workflow

Greg Gavelis, Julia Brown, Ramunas Stepanauskas

Bioinformatics of Microbial Single Cells
Tuesday April 5th, 2022



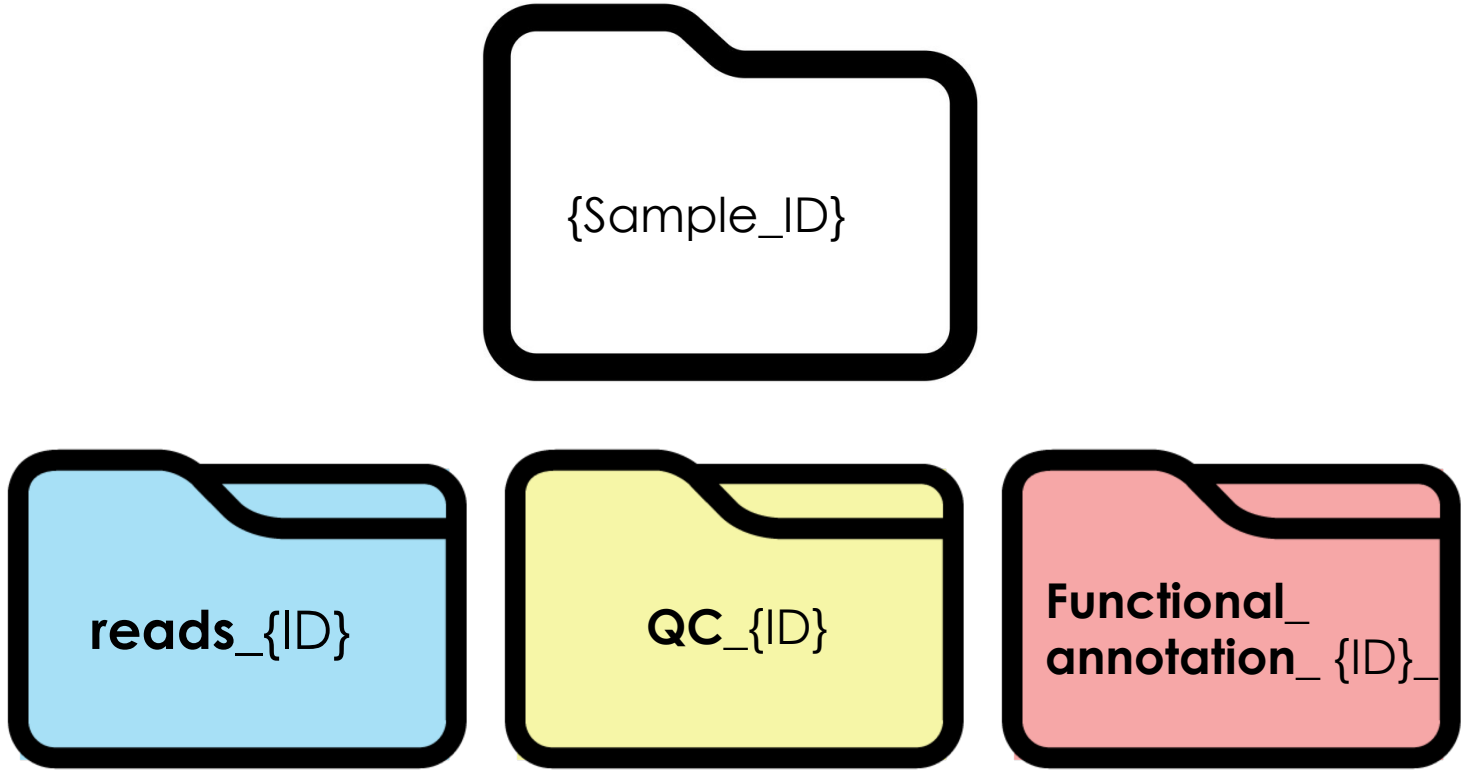
How we generate the final products

& where to find them

GORG_dark has >9000 samples,
with names like:

AH-141-C18
AH-141-D10
AH-141-I04

etc ….



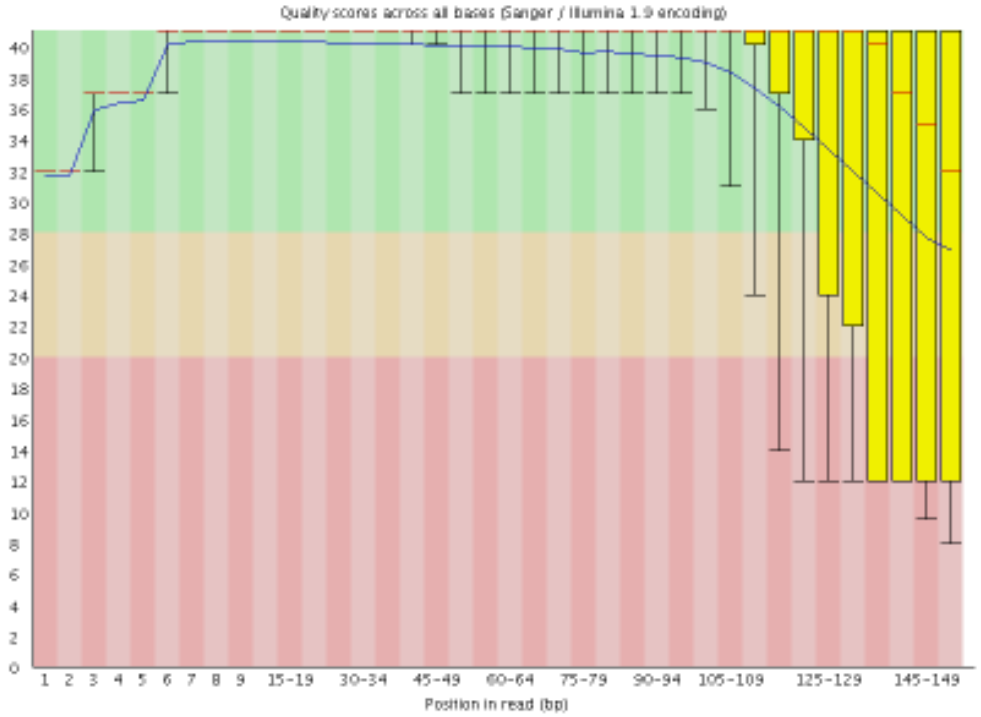
PIPELINE

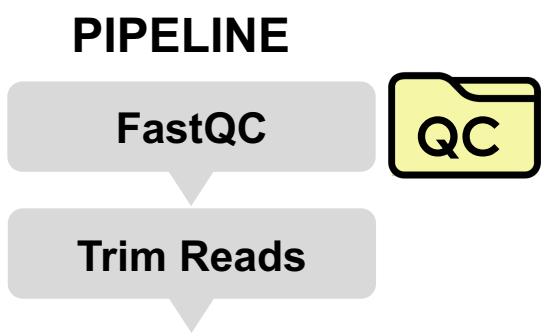
FastQC

QC

1. Report Read Quality

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





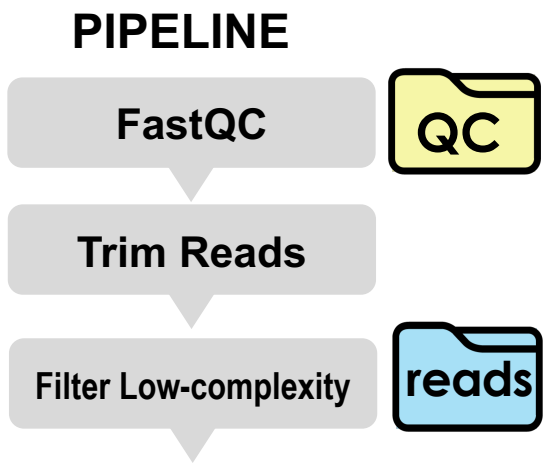
2. Trim Reads

Remove low quality 5’ and 3’ bases.

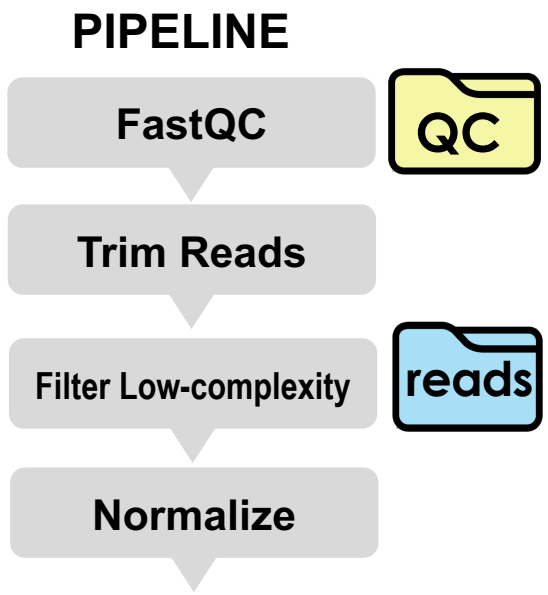
```
trimmomatic PE -phred33 \
        {Forward.fastq} \
        {Reverse.fastq} \
        LEADING:0 TRAILING:5 SLIDINGWINDOW:4:15 MINLEN:36
```

3. Remove low-complexity reads

- Custom script in python



4. Normalize reads



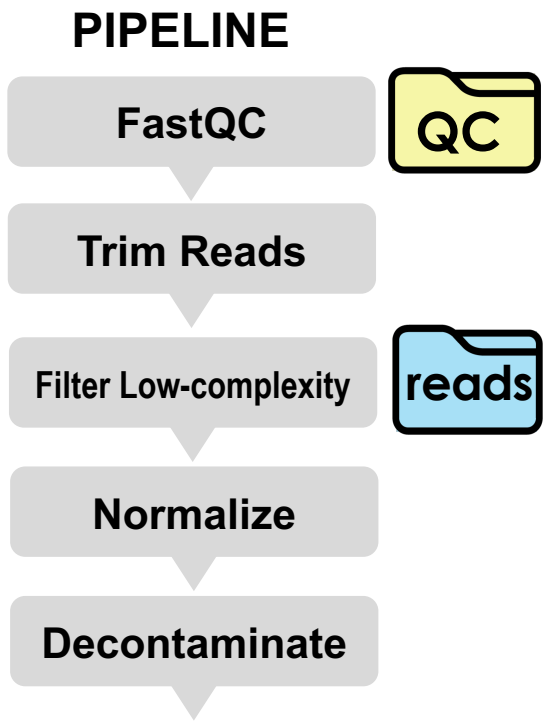
For computational efficiency, downsample readpairs with over-represented kmers.

- (our kmers are 21bp subsequences.)

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

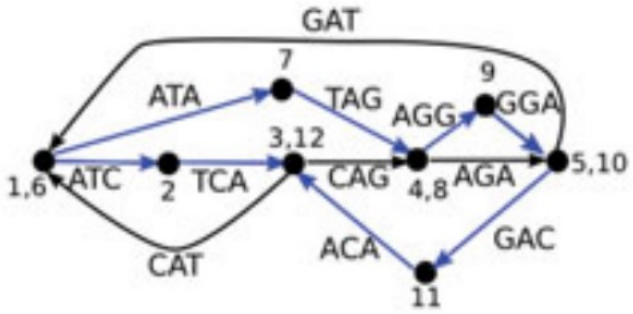
5. Remove contaminant reads

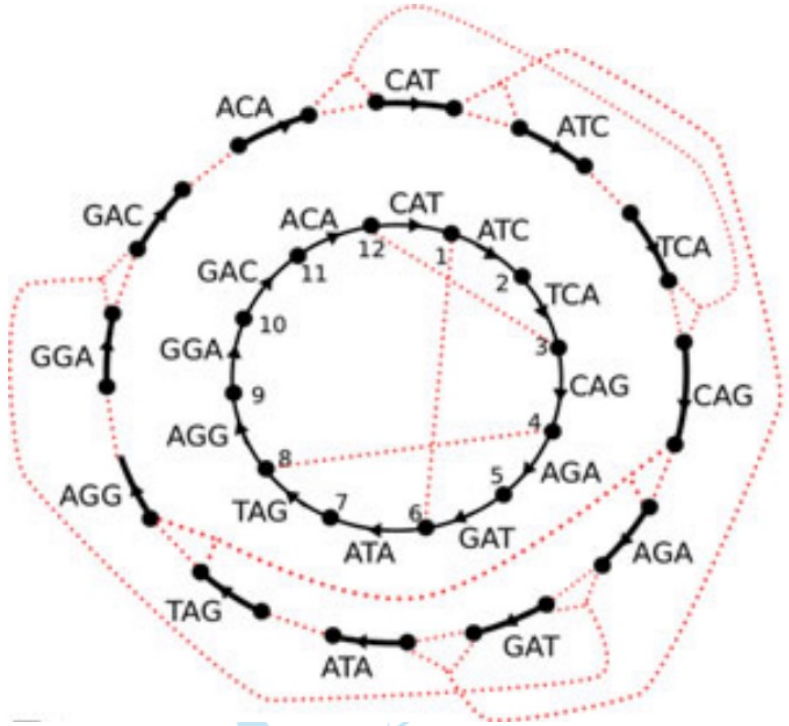
- Custom python script
- BWA (Burroughs-Wheeler Aligner)
- Align to known reagent contaminant
  - Mouse, human, etc.

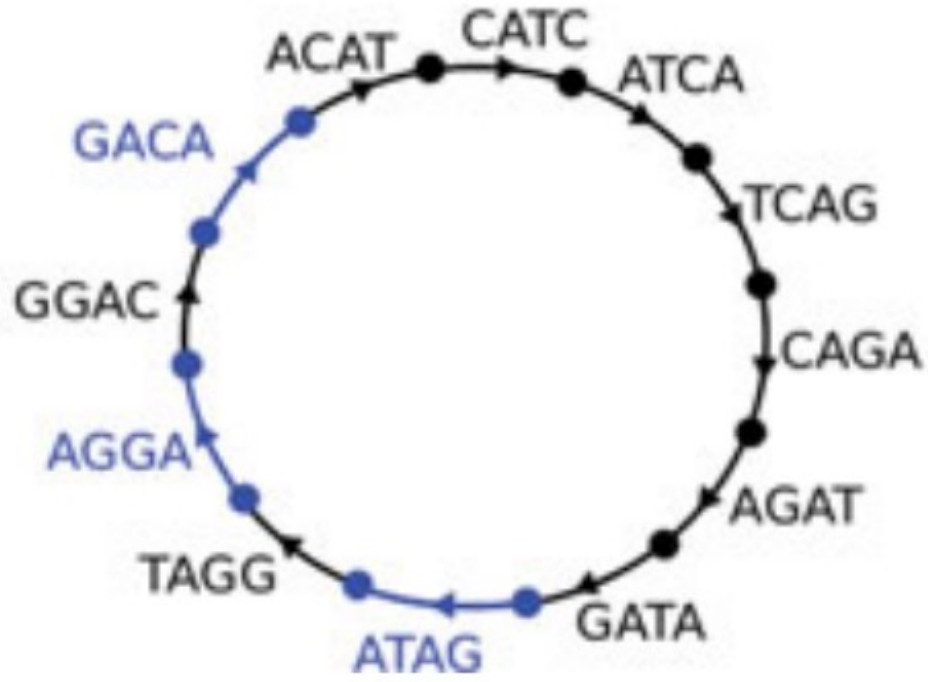


6. Assemble Genome

- **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)

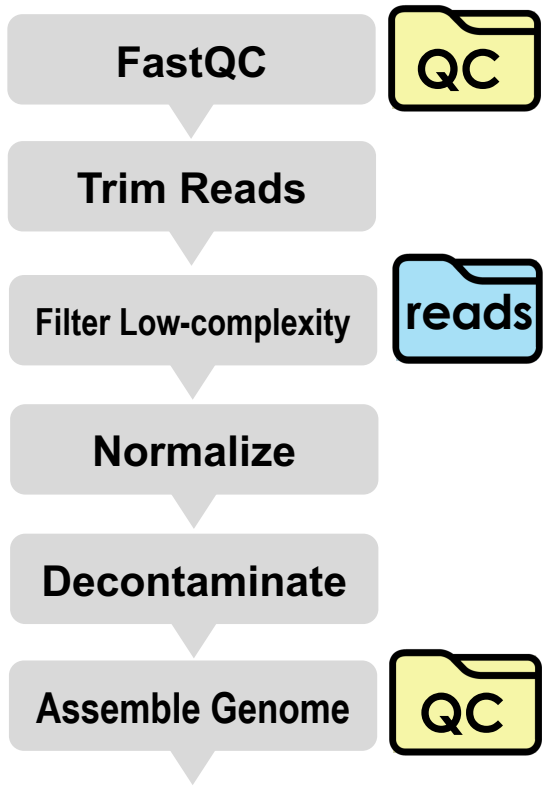






6. Assemble Genome

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



7. BLAST raw contigs against NCBI nr/

- -> To file named “BLAST_raw_contigs_{ID}.tsv”

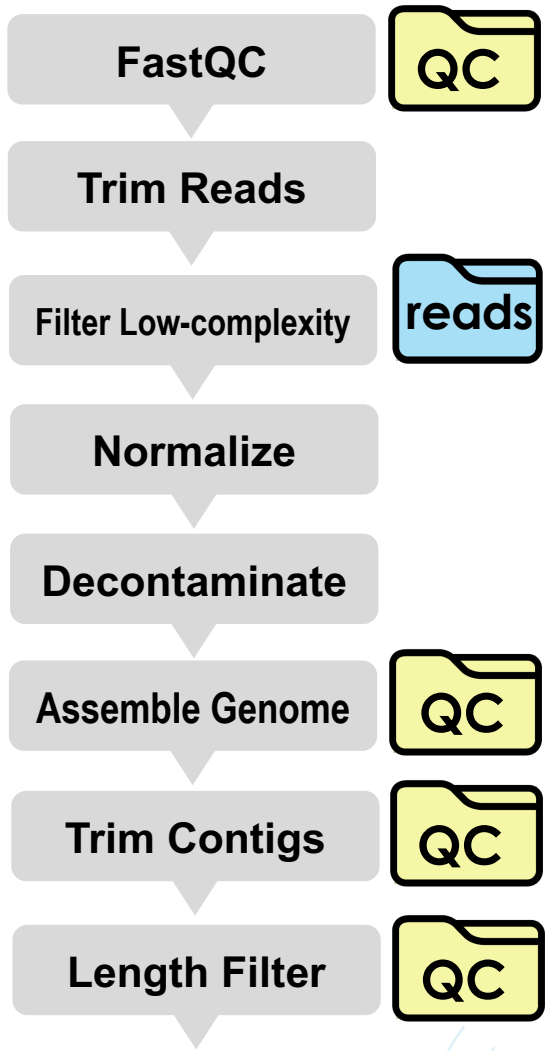
- A good place to look if you’re wondering about possible contaminants in your assembly.

8. Trim contigs

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

9. Length filtering

- Discard assemblies under 2000 bp.

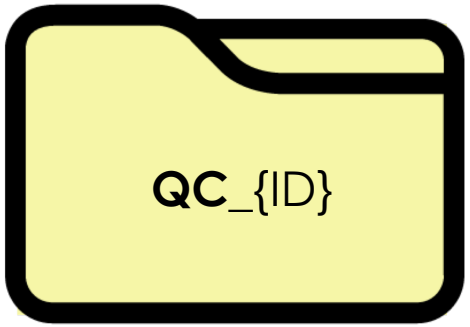


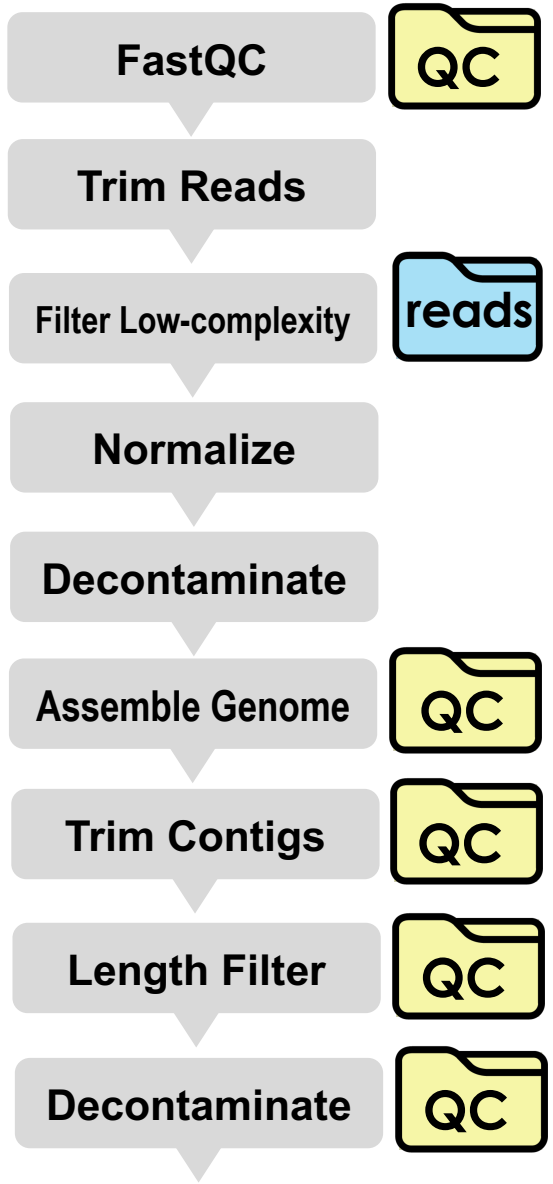
10. Filter contaminant contigs

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

Contaminant contigs are in:

{ID}_contaminated_contigs.fasta



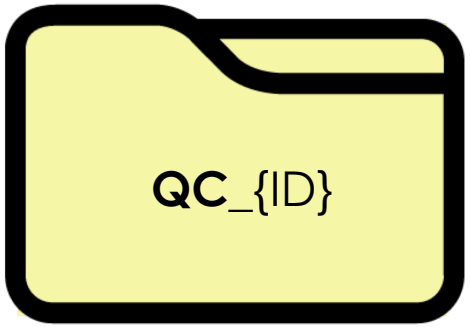


11. Filter contaminant contigs

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

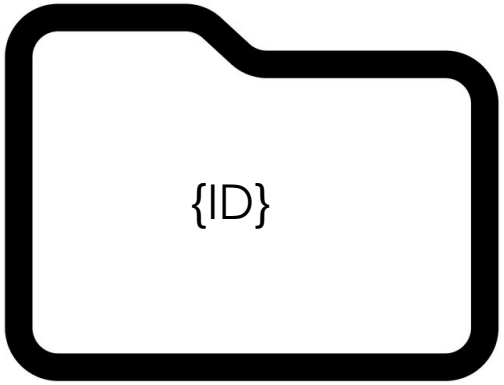
Contaminant contigs are in:

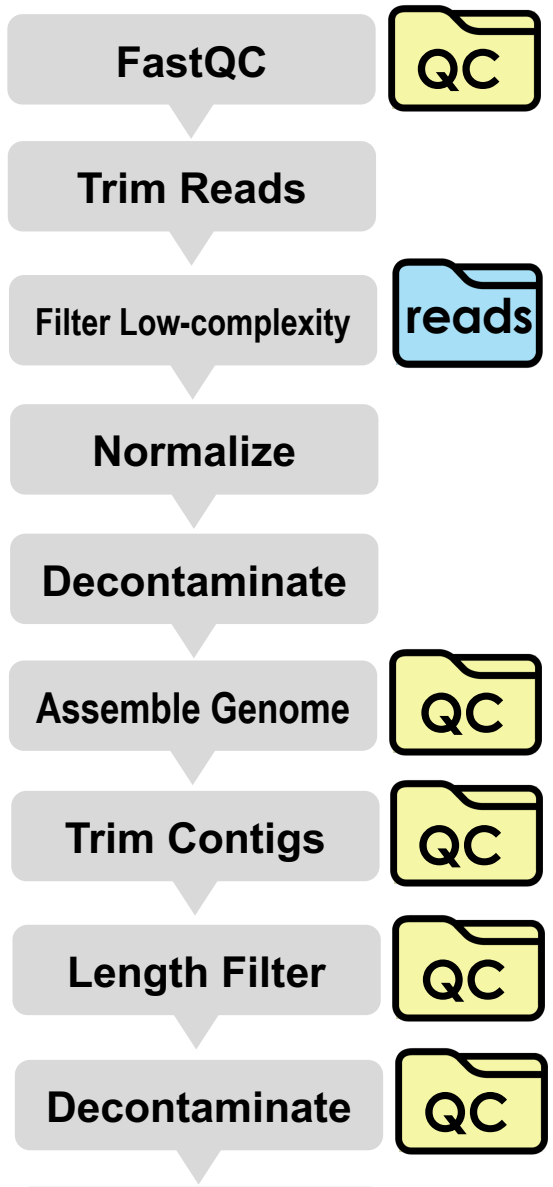
{ID}_contaminated_contigs.fasta



- The final cleaned assembly is in:

{ID}_contigs.fasta



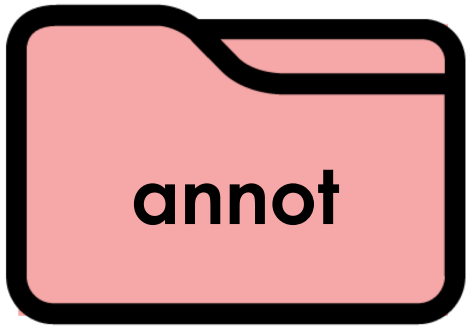


QC

12. CheckM1 to estimate SAG completeness

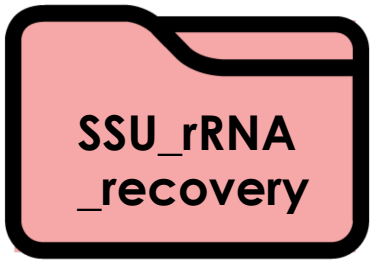
- We've found both CheckM1 (marker gene-based) and CheckM2 (neural network based) to have similar results.
- However, we still use CheckM1 because it is more transparent.
- Important CheckM1 fields:
  - "Completeness" (%)
  - "Multicopy marker genes" (#)
    - Can be useful for inferring contamination.

13. 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.

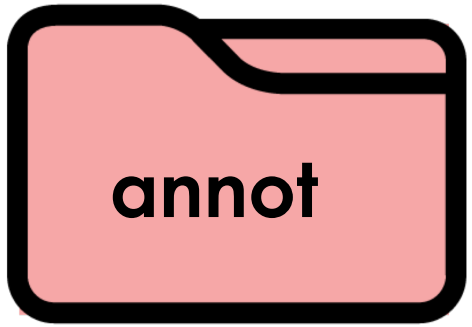
Custom python script (CREST classifier) assigns Linnean classification.

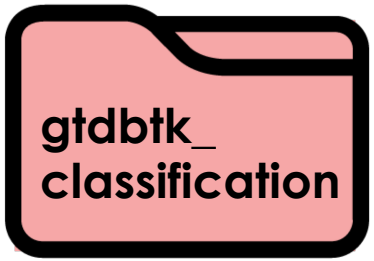
  -> {ID}_**SSU.tsv**

Example:

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

14. GTDB-Tk for multigene classification

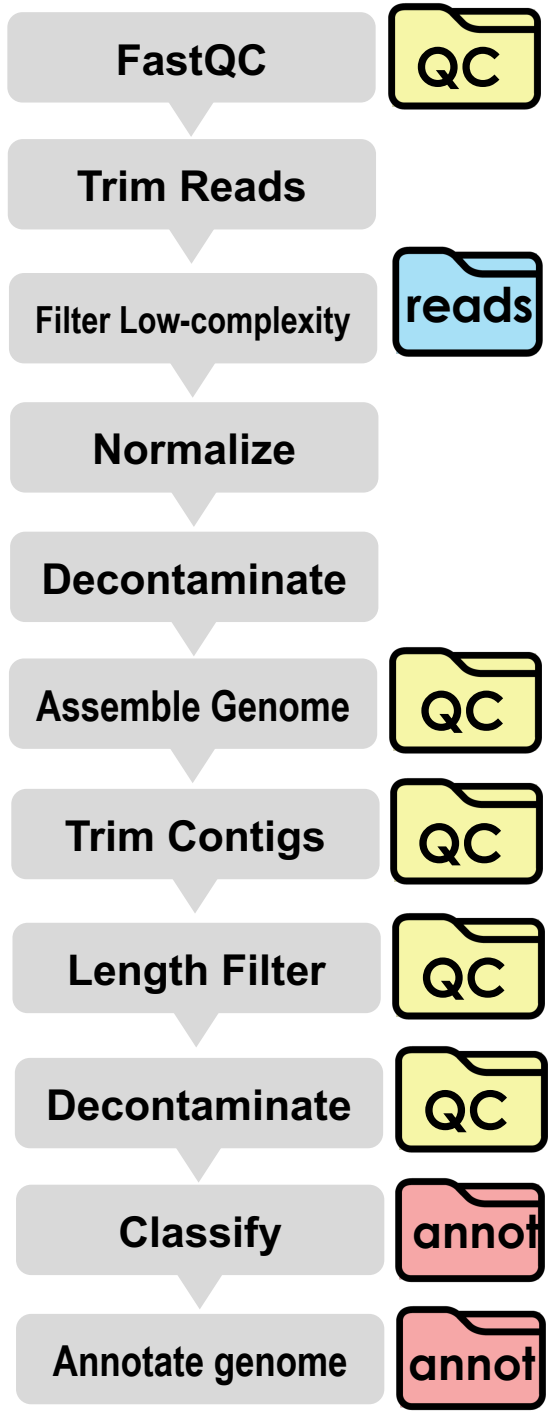




15. Genome annotation with Prokka

- **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)

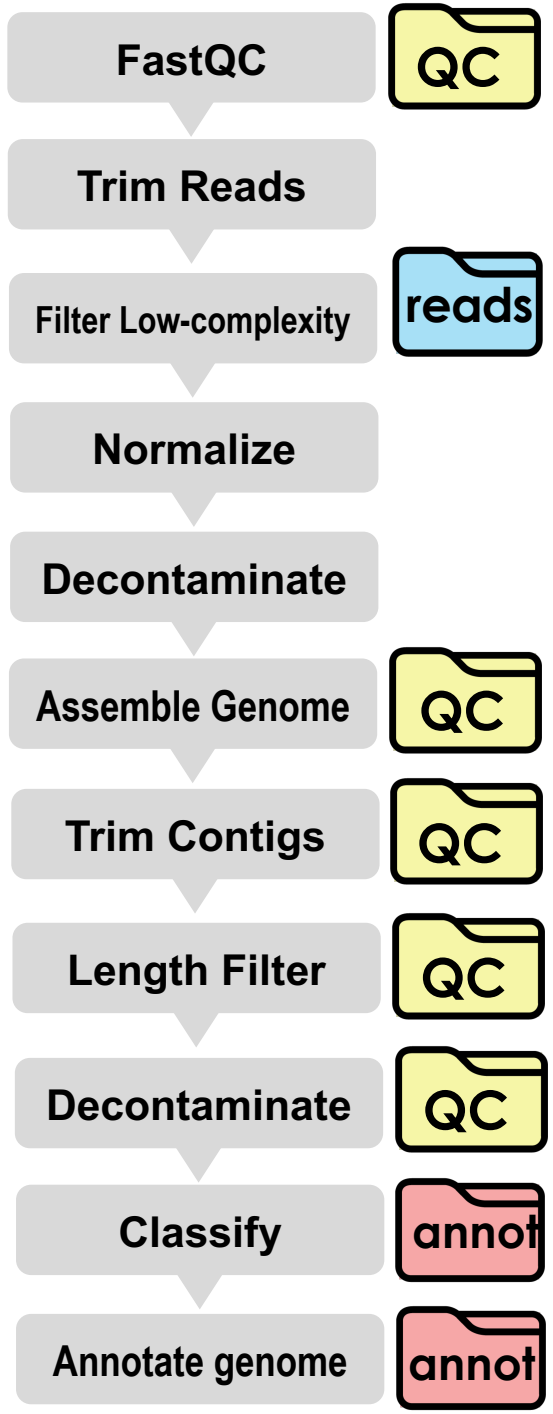


15. Prokka outputs

- Our summary (derived from prokka gff)
  - “comprehensive_prokka”

- Prokka Output files:

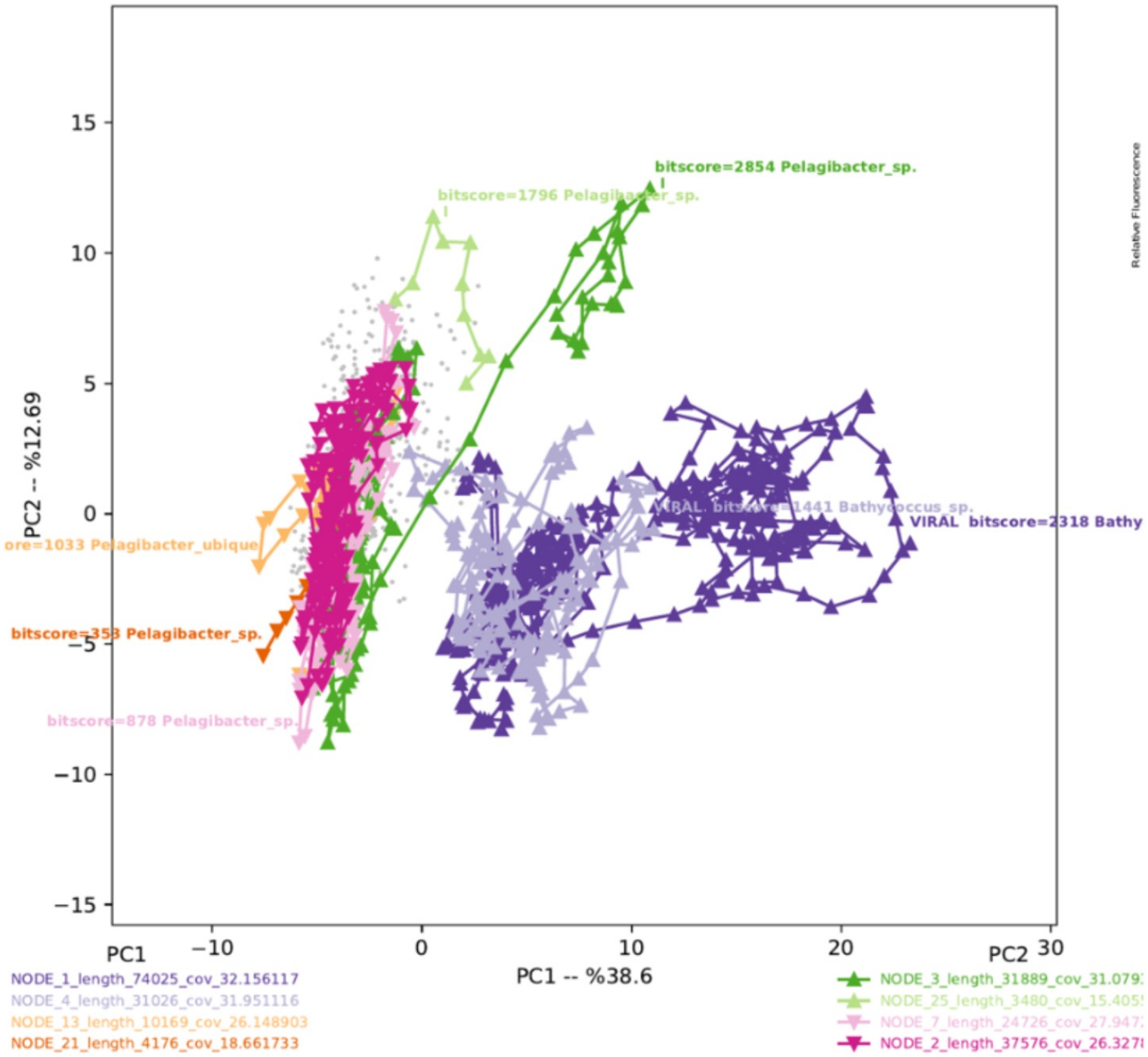
  - **.faa** – Proteins, i.e. translated CDS
  - **.ffn** – Fasta of all genomic features
  - **.gbk**/**.gff** – Files of seqs + annotations
  - . /**tsv**.**tbl** – Feature tables
  - **.txt** – Counts of each feature type



15: Tetramer PCA

QC

Purpose: Flag divergent areas of the genome (potential contaminants / HGT / viruses)

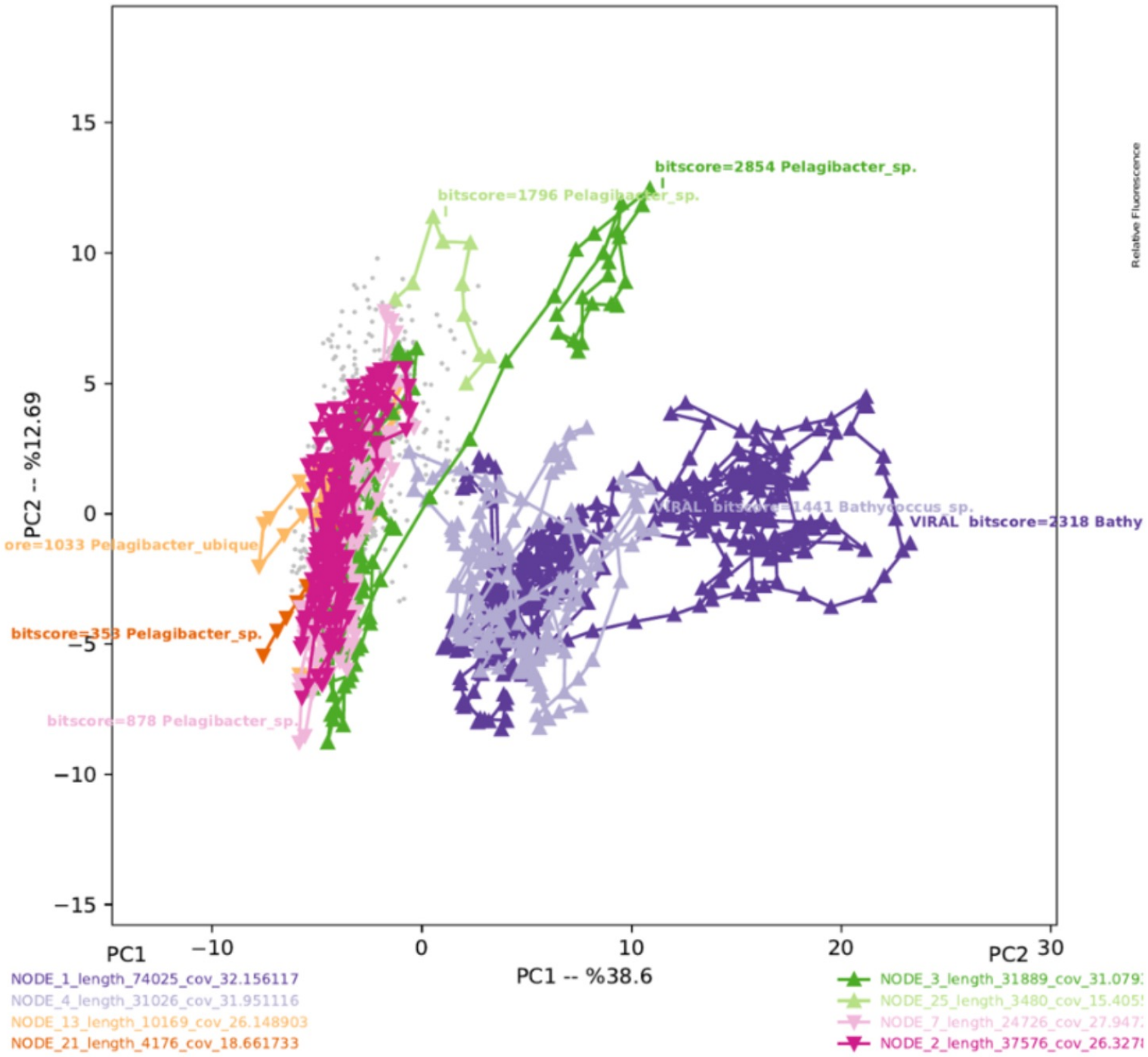


15: Tetramer PCA

QC

Purpose: Flag divergent areas of the genome (potential contaminants / HGT / viruses)

How it works:
1. Calculates tetramer usage for 1600 bp windows of the assembly (sliding along every 200 bp)
2. Flags outliers using Principal Component Analysis (up to 8 outlier contigs are colorized)



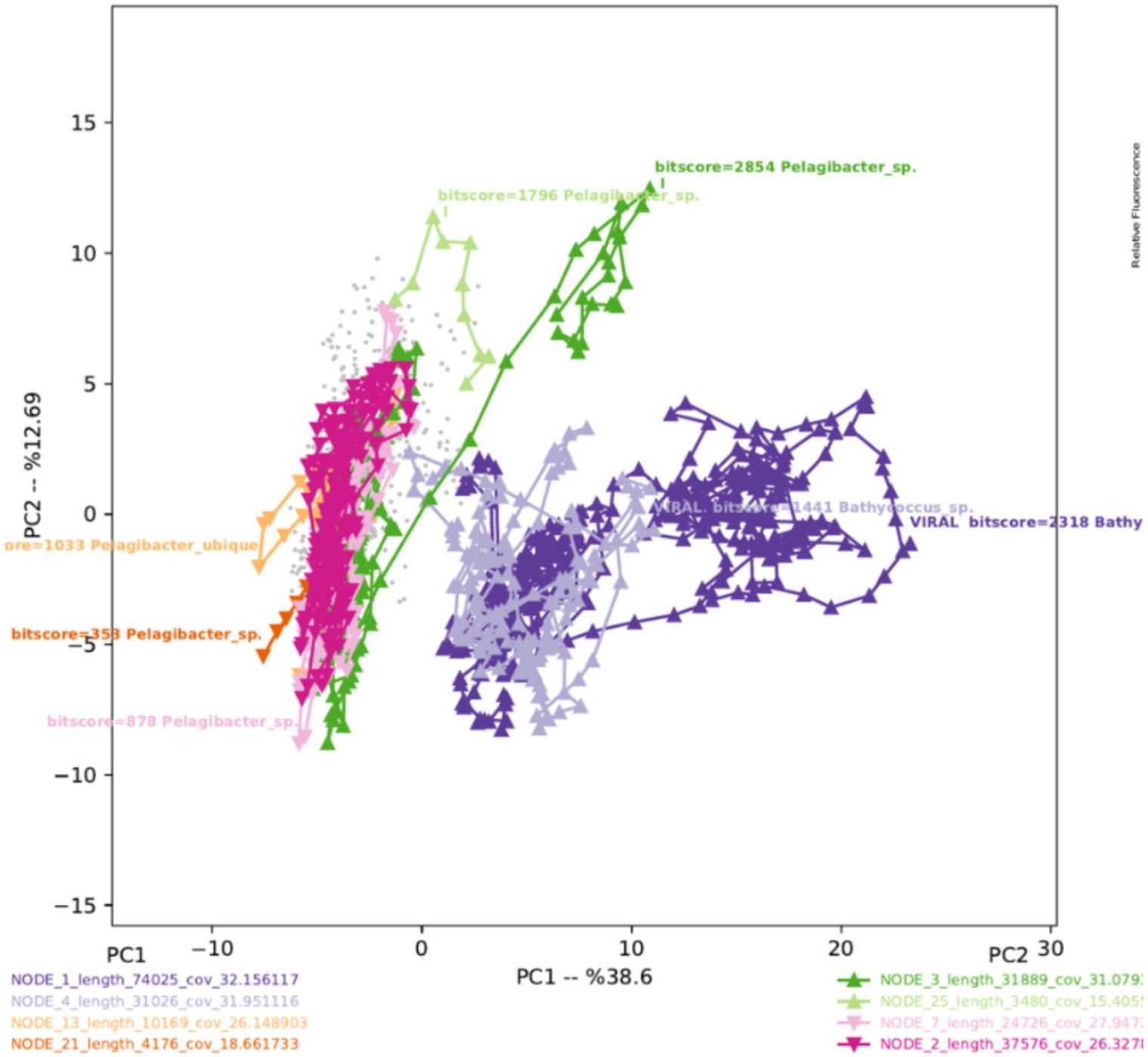
15: Tetramer PCA

QC

Purpose: Flag divergent areas of the genome (potential contaminants / HGT / viruses)

How it works:
1. Calculates tetramer usage for 1600 bp windows of the assembly (sliding along every 200 bp)
2. Flags outliers using Principal Component Analysis (up to 8 outlier contigs are colorized)

Understanding the plot:
- Axes = Principal components 1 & 2
- Gray dots = all windows
- Colors = contigs w/ outlier windows
- In-plot labels: BLASTn hits of those outliers
  - (“VIRAL” flag added based on ncbi-taxid and text terms like “phage”.)
- Legend: Sequence IDs of outlier contigs



16. Lastly, summary table

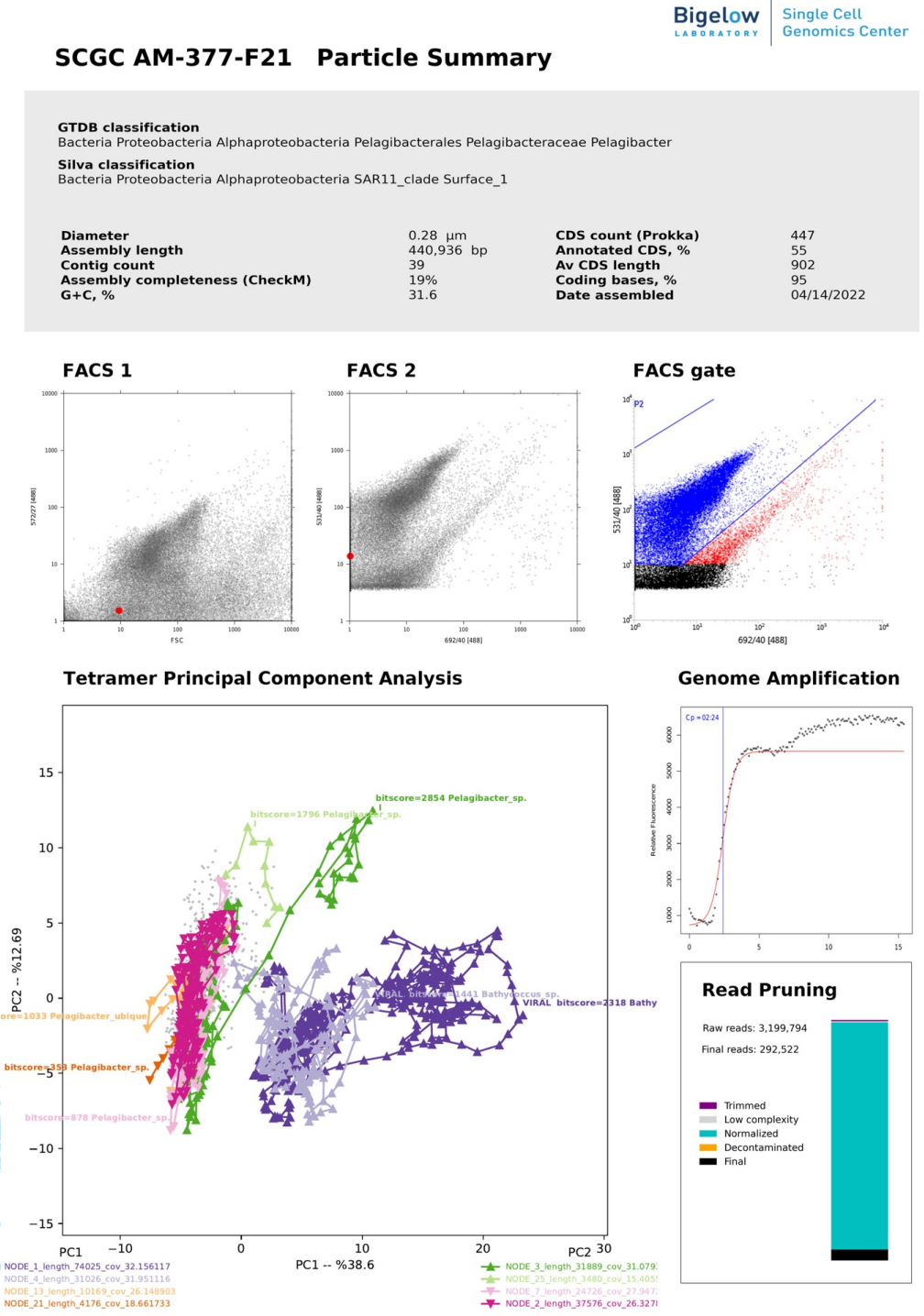
{ID}

Aggregates metrics from FACS, QC & annotation

well	AM-377-F21
well_type	1 cell
diameter	0.28
wga_cp	2.41
raw_read_count	3199794
final_assembly_length	440936
contig_count	39
max_contig_length	74025
gc_content	31.63
coding_density	95.442
rRNA	3
tRNA	11
CDS	447
percent_CDS_annotated	55.26
average_CDS_length	902.24
checkm1_est_genome_completeness	19.04
multicopy_marker_genes	0
classification_via_GTDBTk	d__Bacteria;p__Proteobacteria;c__Alphaproteoba...
1_SSU_classification	k__Bacteria;p__Proteobacteria;c__Alphaproteoba...
2_SSU_classification	NaN
3_SSU_classification	NaN
531/40_488	13.89
572/27_488	1.53
692/40_488	1.02
trigger_pulse_width	9424
side_scatter	9.2
forward_scatter	9.37
probe	Baclight Redox Sensor Green
run_id	211102_VH00511_31_AAAJ5NCHV
notes	NaN

New from SCGC:
"Particle Summary"

- Visual summary of SAG
- Developed after GORG Dark (sorry!)



Future Improvements:

- Please let us know what tools could be used to improve our pipeline

- Thanks!