Bacteriophages - Walkthrough

Notebook:	Thesis Methods		
Created:	15/09/2020 16:39	**Updated:**	15/09/2020 17:36
Author:	Jess Friedersdorff		
URL:	https://www.frontiersin.org/articles/10.3389/fmicb.2020.01588/full		

Quality control with FastQC

FastQC version 0.11.8:

```
for i in *.fastq.gz;
    do fastqc $i;
done
```

Trimming with Sickle

As suggested or used by:

- Rihtman, B., Meaden, S., Clokie, M. R., Koskella, B., & Millard, A. D. (2016). Assessing Illumina technology for the high-throughput sequencing of bacteriophage genomes. *PeerJ, 4*, e2055. doi:10.7717/peerj.2055,
- Pavelas Sazinas, Tamsin Redgwell, Branko Rihtman, Aurelija Grigonyte, Slawomir Michniewski, David J Scanlan, Jon Hobman, Andrew Millard, Comparative Genomics of Bacteriophage of the genus *Seuratvirus*, *Genome Biology and Evolution*, Volume 10, Issue 1, January 2018, Pages 72–76, https://doi.org/10.1093/gbe/evx275,
- http://millardlab.org/lab-members/lucy-gannon/lucys-beginner-guide-to-bacteriophage-genome-assembly-2/)

Ran Sickle with default parameters on paired end setting (as per the Sazinas *et al* 2018 paper:

```
sickle pe -f C_S40_L001_R1_001.fastq.gz -r C_S40_L001_R2_001.fastq.gz -t sanger -o sickle_trimmed_C_S40_L001_R1 -p
sickle_trimmed_C_S40_L001_R2 -s sickle_singles_C_S40_L001


sickle pe -f D_S41_L001_R1_001.fastq.gz -r D_S41_L001_R2_001.fastq.gz -t sanger -o sickle_trimmed_D_S41_L001_R1 -p
sickle_trimmed_D_S41_L001_R2 -s sickle_singles_D_S41_L001


sickle pe -f J_S42_L001_R1_001.fastq.gz -r J_S42_L001_R2_001.fastq.gz -t sanger -o sickle_trimmed_J_S42_L001_R1 -p
sickle_trimmed_J_S42_L001_R2 -s sickle_singles_J_S42_L001


sickle pe -f M_S38_L001_R1_001.fastq.gz -r M_S38_L001_R2_001.fastq.gz -t sanger -o sickle_trimmed_M_S38_L001_R1 -p
sickle_trimmed_M_S38_L001_R2 -s sickle_singles_M_S38_L001


sickle pe -f P_S39_L001_R1_001.fastq.gz -r P_S39_L001_R2_001.fastq.gz -t sanger -o sickle_trimmed_P_S39_L001_R1 -p
sickle_trimmed_P_S39_L001_R2 -s sickle_singles_P_S39_L001
```

Sickle output:

```
PE forward file: C_S40_L001_R1_001.fastq.gz
PE reverse file: C_S40_L001_R2_001.fastq.gz


Total input FastQ records: 509514 (254757 pairs)


FastQ paired records kept: 507330 (253665 pairs)
FastQ single records kept: 1010 (from PE1: 966, from PE2: 44)
FastQ paired records discarded: 164 (82 pairs)
FastQ single records discarded: 1010 (from PE1: 44, from PE2: 966)




PE forward file: D_S41_L001_R1_001.fastq.gz
PE reverse file: D_S41_L001_R2_001.fastq.gz


Total input FastQ records: 589102 (294551 pairs)


FastQ paired records kept: 586454 (293227 pairs)
FastQ single records kept: 1234 (from PE1: 1196, from PE2: 38)
FastQ paired records discarded: 180 (90 pairs)
FastQ single records discarded: 1234 (from PE1: 38, from PE2: 1196)




PE forward file: J_S42_L001_R1_001.fastq.gz
PE reverse file: J_S42_L001_R2_001.fastq.gz


Total input FastQ records: 869112 (434556 pairs)


FastQ paired records kept: 865374 (432687 pairs)
FastQ single records kept: 1647 (from PE1: 1583, from PE2: 64)
FastQ paired records discarded: 444 (222 pairs)
FastQ single records discarded: 1647 (from PE1: 64, from PE2: 1583)
```

```
PE forward file: M_S38_L001_R1_001.fastq.gz
PE reverse file: M_S38_L001_R2_001.fastq.gz


Total input FastQ records: 773164 (386582 pairs)


FastQ paired records kept: 768066 (384033 pairs)
FastQ single records kept: 2103 (from PE1: 2053, from PE2: 50)
FastQ paired records discarded: 892 (446 pairs)
FastQ single records discarded: 2103 (from PE1: 50, from PE2: 2053)




PE forward file: P_S39_L001_R1_001.fastq.gz
PE reverse file: P_S39_L001_R2_001.fastq.gz


Total input FastQ records: 685160 (342580 pairs)


FastQ paired records kept: 680668 (340334 pairs)
FastQ single records kept: 1706 (from PE1: 1648, from PE2: 58)
FastQ paired records discarded: 1080 (540 pairs)
FastQ single records discarded: 1706 (from PE1: 58, from PE2: 1648)
```

Assembly with SPAdes

Used SPAdes to assemble the genomes from the trimmed reads after applying Sickle, using the only-assembler parameter as suggested in the Sazinas *et al* 2018 paper.

```
SPAdes genome assembler v3.13.0

spades.py -o phage_C_spades -1 ../sickle_trimmed_C_S40_L001_R1.fastq -2 ../sickle_trimmed_C_S40_L001_R2.fastq -s
../sickle_singles_C_S40_L001.fastq --only-assembler --threads 4


spades.py -o phage_D_spades -1 ../sickle_trimmed_D_S41_L001_R1.fastq -2 ../sickle_trimmed_D_S41_L001_R2.fastq -s
../sickle_singles_D_S41_L001.fastq --only-assembler --threads 4


spades.py -o phage_J_spades -1 ../sickle_trimmed_J_S42_L001_R1.fastq -2 ../sickle_trimmed_J_S42_L001_R2.fastq -s
../sickle_singles_J_S42_L001.fastq --only-assembler --threads 4


spades.py -o phage_P_spades -1 ../sickle_trimmed_P_S39_L001_R1.fastq -2 ../sickle_trimmed_P_S39_L001_R2.fastq -s
../sickle_singles_P_S39_L001.fastq --only-assembler --threads 4


spades.py -o phage_M_spades -1 ../sickle_trimmed_M_S38_L001_R1.fastq -2 ../sickle_trimmed_M_S38_L001_R2.fastq -s
../sickle_singles_M_S38_L001.fastq --only-assembler --threads 4
```

Finding the Phage Genomes

By looking at the contigs.fasta from the assembled reads for each of the five phages, SPAdes gives a read length and a coverage. The coverage in this case in K-mer coverage, how many times the last (longest) kmer used covers the contig. It can be less than one and it is always lower than the nucleotide coverage (coverage per base).

I can essentially just look for the assemblies with the highest coverage, then pick the ones with a reasonable length in the scaffolds:

```
[jef11@bert spades_scaffolds]$ for i in phage_*_spades_scaffolds.fasta; do echo $i; grep ">" $i | sed 's/_/\t/g' | sort -k6 -Vr |
head -10; done
phage_C_spades_scaffolds.fasta
>NODE   278     length  128     cov     557.000000
>NODE   276     length  165     cov     124.263158
>NODE   185     length  1843    cov     79.016900
>NODE   274     length  186     cov     78.067797
>NODE   148     length  4818    cov     69.342145
>NODE   193     length  1531    cov     62.215100
>NODE   182     length  1935    cov     59.768252
>NODE   49      length  33354   cov     53.276913
>NODE   277     length  132     cov     41.400000
>NODE   162     length  3438    cov     41.165811
phage_D_spades_scaffolds.fasta
>NODE   13      length  31245   cov     592.622244
>NODE   675     length  128     cov     428.000000
>NODE   669     length  170     cov     81.348837
>NODE   306     length  4762    cov     80.925135
>NODE   472     length  1844    cov     76.140361
>NODE   501     length  1531    cov     59.183761
>NODE   658     length  368     cov     53.170124
>NODE   371     length  3438    cov     35.963153
>NODE   616     length  654     cov     33.641366
>NODE   666     length  212     cov     32.541176
phage_J_spades_scaffolds.fasta
>NODE   870     length  128     cov     801.000000
>NODE   6       length  33354   cov     573.119090
>NODE   342     length  4762    cov     115.791154
>NODE   612     length  1843    cov     101.720862
>NODE   9       length  31255   cov     101.420811
>NODE   669     length  1435    cov     72.848624
>NODE   594     length  1935    cov     71.391593
>NODE   434     length  3438    cov     55.538206
>NODE   491     length  2840    cov     45.026539
>NODE   869     length  129     cov     34.000000
phage_M_spades_scaffolds.fasta
>NODE   447     length  128     cov     1120.000000
```

```
>NODE   440     length  181     cov     144.092593
>NODE   20      length  39872   cov     136.182740
>NODE   312     length  1843    cov     89.714452
>NODE   234     length  4762    cov     88.875512
>NODE   329     length  1531    cov     72.407407
>NODE   309     length  1935    cov     62.214602
>NODE   444     length  132     cov     49.400000
>NODE   257     length  3438    cov     46.836303
>NODE   441     length  170     cov     46.418605
phage_P_spades_scaffolds.fasta
>NODE   45      length  33626   cov     295.842503
>NODE   182     length  128     cov     250.000000
>NODE   178     length  229     cov     104.941176
>NODE   127     length  1843    cov     70.824592
>NODE   134     length  1531    cov     61.509259
>NODE   108     length  4733    cov     53.727095
>NODE   125     length  1935    cov     47.194690
>NODE   117     length  2941    cov     38.145345
>NODE   118     length  2840    cov     34.526355
>NODE   113     length  3438    cov     31.592268
```

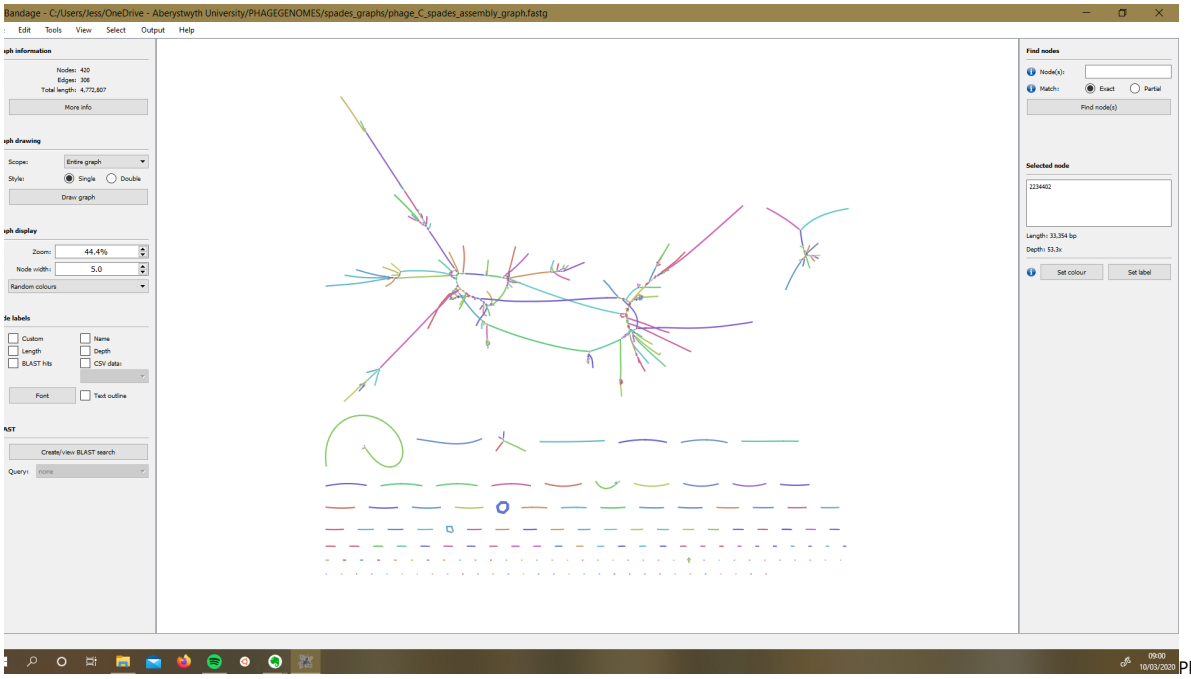
Phage genome - NODES:

Phage_C_Genome_scaffold.fasta	>NODE_49_length_33354_cov_53.276913
Phage_D_Genome_scaffold.fasta	>NODE_13_length_31245_cov_592.622244
Phage_J_Genome_1_scaffold.fasta	>NODE_6_length_33354_cov_573.119090
Phage_J_Genome_2_scaffold.fasta	>NODE_9_length_31255_cov_101.420811
Phage_M_Genome_scaffold.fasta	>NODE_20_length_39872_cov_136.182740
Phage_P_Genome_scaffold.fasta	>NODE_45_length_33626_cov_295.842503

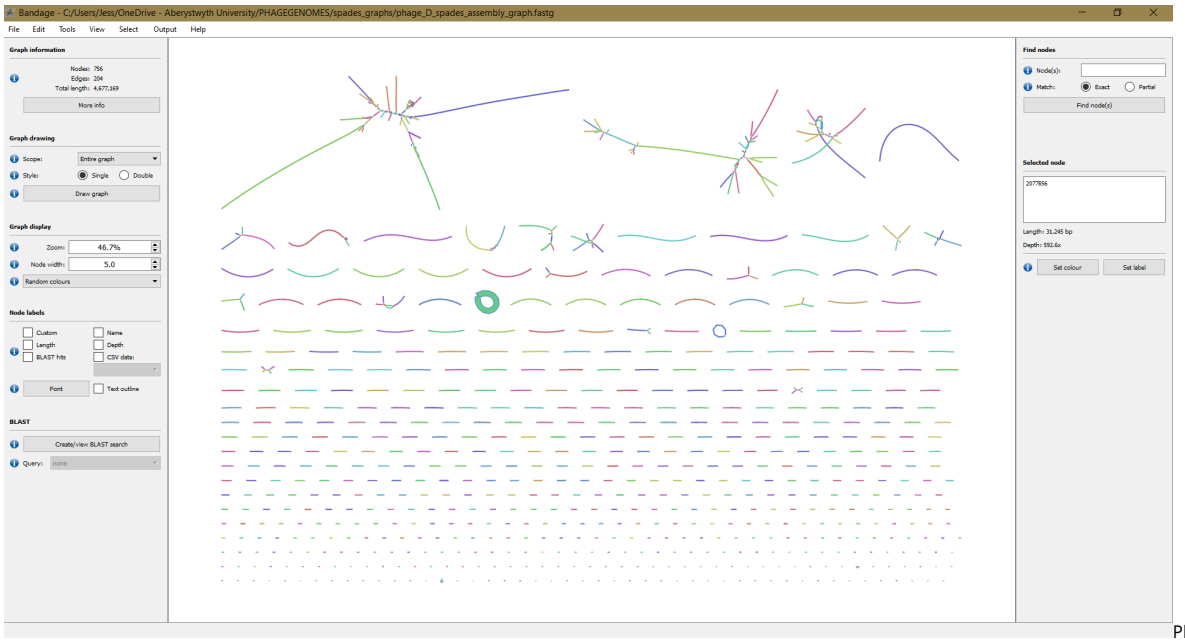
Note that SPAdes does not circularise. In other words, if these genomes can circularise, then there may be overlap in the nucleotides - A -> B -> A. Look at the genomes in Bandage to see whether they circularise.

Bandage - finding the phage genomes

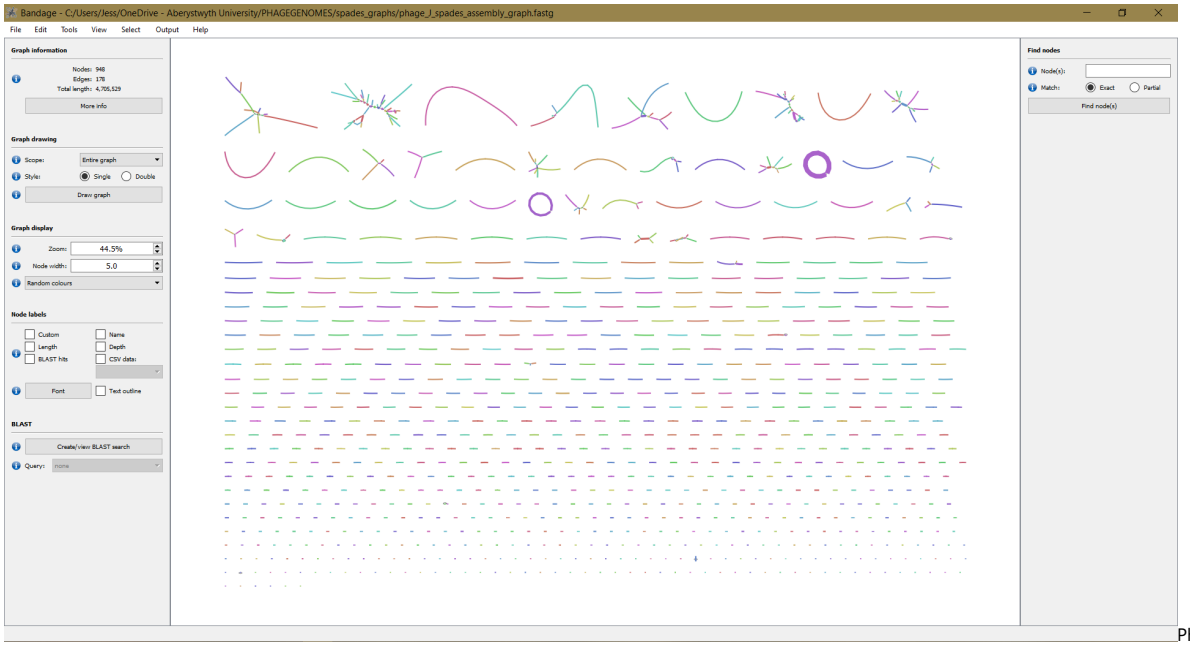
Upload the fastg path files from SPAdes into Bandage, choose the contig that represents the phage genome (the length will be the same as the node length in the spades contig file) and export the path to get just the sequence of the genome.



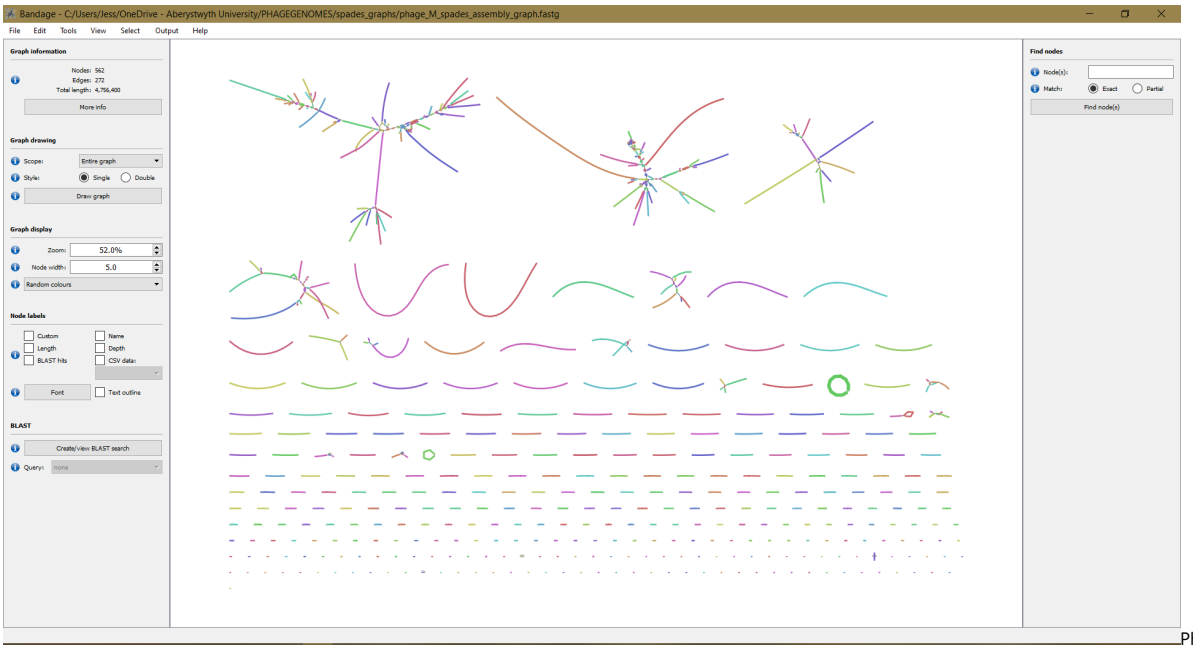
Phage C



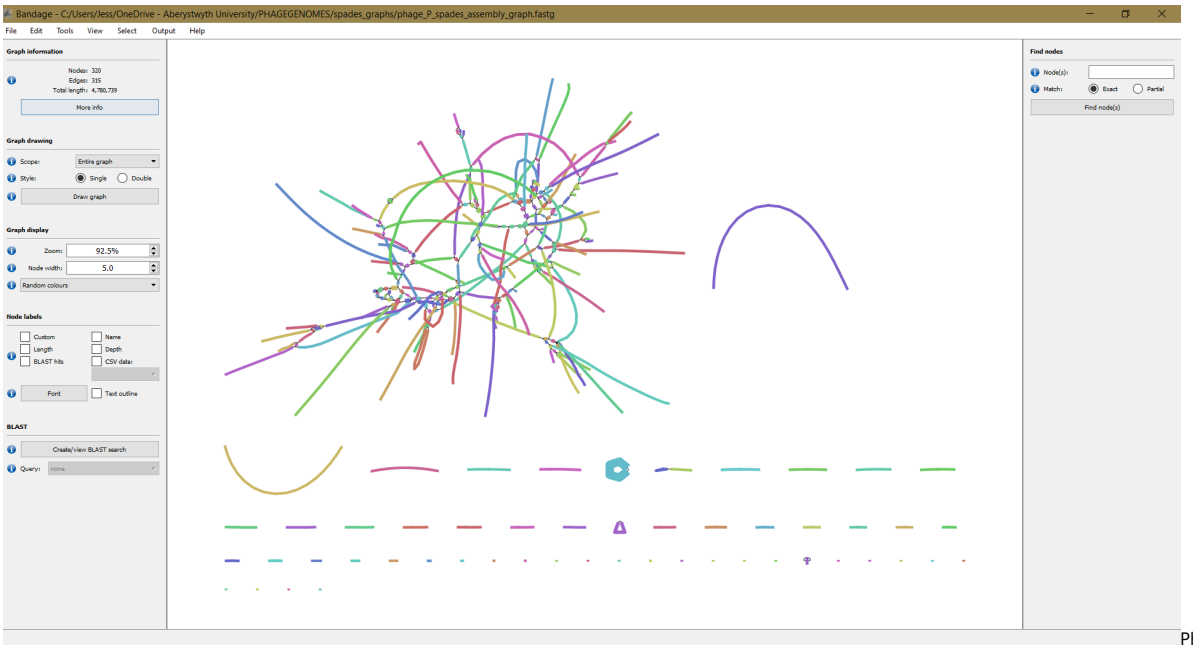
Phage D



Phage J - note the larger contig is J-1 and the smaller is J-2.



Phage M



Phage P

Export each of these using output -> save selected path sequence to FASTA.
Then I can rename these, and check the lengths:

Phage scaffold name	length (bp)
>Phage_C_Bandage_seq	34177
>Phage_D_Bandage_seq	32008
>Phage_J-1_Bandage_seq	34177
>Phage_J-2_Bandage_seq	32018
>Phage_M_Bandage_seq	40881
>Phage_P_Bandage_seq	34457

Finding and Removing Repeats:

I used Repeat Finder in Geneious with default settings:

phage genome	repeat length (bp)	
C	127	NODE_49_length_33354_cov_53.276913

D	127	NODE_13_length_31245_cov_592.622244
J-1	127	NODE_6_length_33354_cov_573.119090
J-2	127	NODE_9_length_31255_cov_101.420811
P	127	NODE_46_length_33626_cov_295.842503
M	127	NODE_20_length_39872_cov_136.182740Then I

I think 127 is the kmer length used which is why they are all the same. Then manually removed this 127 section from the relevant NODES in the contig files from spades output within Geneious. I then realigned the reads to the contigs using bwa-mem and extracted the node corresponding to the phage genomes using samtools again:

```
bwa index phage_C_spades_contigs.fasta.edited
bwa mem phage_C_spades_contigs.fasta.edited -t 7 sickle_trimmed_C_S40_L001_R1.fastq sickle_trimmed_C_S40_L001_R2.fastq | samtools
view -bS -F260 | samtools sort -o phage_C_bwa.bam.edited
samtools index phage_C_bwa.bam.edited
samtools view -b phage_C_bwa.bam.edited NODE_49_length_33354_cov_53.276913 | samtools sort -o phage_C_bwa_phagereads.bam.edited
samtools index phage_C_bwa_phagereads.bam.edited
samtools depth phage_C_bwa_phagereads.bam.edited > phage_C_bwa_phagereads.depth.edited
samtools stats phage_C_bwa_phagereads.bam.edited > phage_C_bwa_phagereads.stats.edited
samtools flagstats phage_C_bwa_phagereads.bam.edited > phage_C_bwa_phagereads.flagstats
```

```
samtools faidx phage_C_spades_contigs.fasta.edited NODE_49_length_33354_cov_53.276913 > phage_C_genomeseq_edited.fasta
samtools faidx phage_D_spades_contigs.fasta.edited NODE_13_length_31245_cov_592.622244 > phage_D_genomeseq_edited.fasta
samtools faidx phage_J_spades_contigs.fasta.edited NODE_6_length_33354_cov_573.119090 > phage_J-1_genomeseq_edited.fasta
samtools faidx phage_J_spades_contigs.fasta.edited NODE_9_length_31255_cov_101.420811 > phage_J-2_genomeseq_edited.fasta
samtools faidx phage_P_spades_contigs.fasta.edited NODE_46_length_33626_cov_295.842503 > phage_P_genomeseq_edited.fasta
samtools faidx phage_M_spades_contigs.fasta.edited NODE_20_length_39872_cov_136.182740 > phage_M_genomeseq_edited.fasta
```

then these were just copied locally (the bam and the .fasta I made using samtools faidx) and put them into geneious.

Assessing Genome Assembly Quality using Pilon

Used Pilon version 1.23 and running it using this:

```
java -jar pilon-1.23.jar --genome phage_J-1_genomeseq_edited.fasta --bam phage_J-1_bwa_phagereads.bam.edited --output phage_J-1
>phage_J-1_pilon.output
java -jar pilon-1.23.jar --genome phage_J-2_genomeseq_edited.fasta --bam phage_J-2_bwa_phagereads.bam.edited --output phage_J-2
>phage_J-2_pilon.output
java -jar pilon-1.23.jar --genome phage_C_genomeseq_edited.fasta --bam phage_C_bwa_phagereads.bam.edited --output phage_C
>phage_C_pilon.output
java -jar pilon-1.23.jar --genome phage_D_genomeseq_edited.fasta --bam phage_D_bwa_phagereads.bam.edited --output phage_D
>phage_D_pilon.output
java -jar pilon-1.23.jar --genome phage_M_genomeseq_edited.fasta --bam phage_M_bwa_phagereads.bam.edited --output phage_M
>phage_M_pilon.output
java -jar pilon-1.23.jar --genome phage_P_genomeseq_edited.fasta --bam phage_P_bwa_phagereads.bam.edited --output phage_P
>phage_P_pilon.output
```

Outputs

```
for i in *.output; do cat ${i}; echo -e "\n----------\n";  done
Pilon version 1.23 Mon Nov 26 16:04:05 2018 -0500
Genome: phage_C_genomeseq_edited.fasta
Fixing snps, indels, gaps, local
Input genome size: 33227
Scanning BAMs
phage_C_bwa_phagereads.bam.edited: 17626 reads, 0 filtered, 17626 mapped, 17559 proper, 36 stray, FR 99% 443+/-1659, max 5420 frags
Processing NODE_49_length_33354_cov_53.276913:1-33227
frags phage_C_bwa_phagereads.bam.edited: coverage 115
Total Reads: 17626, Coverage: 115, minDepth: 12
Confirmed 33216 of 33227 bases (99.97%)
Corrected 0 snps; 0 ambiguous bases; corrected 0 small insertions totaling 0 bases, 0 small deletions totaling 0 bases
NODE_49_length_33354_cov_53.276913:1-33227 log:
Finished processing NODE_49_length_33354_cov_53.276913:1-33227
Writing updated NODE_49_length_33354_cov_53.276913_pilon to phage_C.fasta
Mean frags coverage: 115
Mean total coverage: 115


----------


Pilon version 1.23 Mon Nov 26 16:04:05 2018 -0500
Genome: phage_D_genomeseq_edited.fasta
Fixing snps, indels, gaps, local
Input genome size: 31118
Scanning BAMs
phage_D_bwa_phagereads.bam.edited: 179683 reads, 0 filtered, 179683 mapped, 177810 proper, 1352 stray, FR 100% 429+/-686, max 2487
frags
Processing NODE_13_length_31245_cov_592.622244:1-31118
frags phage_D_bwa_phagereads.bam.edited: coverage 1265
Total Reads: 179683, Coverage: 1265, minDepth: 127
Confirmed 30946 of 31118 bases (99.45%)
Corrected 0 snps; 0 ambiguous bases; corrected 0 small insertions totaling 0 bases, 0 small deletions totaling 0 bases
NODE_13_length_31245_cov_592.622244:1-31118 log:
Finished processing NODE_13_length_31245_cov_592.622244:1-31118
Writing updated NODE_13_length_31245_cov_592.622244_pilon to phage_D.fasta
Mean frags coverage: 1265
Mean total coverage: 1265


----------


Pilon version 1.23 Mon Nov 26 16:04:05 2018 -0500
Genome: phage_J-1_genomeseq_edited.fasta
```

```
Fixing snps, indels, gaps, local
Input genome size: 33227
Scanning BAMs
phage_J-1_bwa_phagereads.bam.edited: 208496 reads, 0 filtered, 208496 mapped, 206514 proper, 1520 stray, FR 100% 324+/-1188, max 3888
frags
Processing NODE_6_length_33354_cov_573.119090:1-33227
frags phage_J-1_bwa_phagereads.bam.edited: coverage 1264
Total Reads: 208496, Coverage: 1264, minDepth: 126
Confirmed 33084 of 33227 bases (99.57%)
Corrected 0 snps; 0 ambiguous bases; corrected 0 small insertions totaling 0 bases, 0 small deletions totaling 0 bases
# Attempting to fix local continuity breaks
NODE_6_length_33354_cov_573.119090:1-33227 log:
Finished processing NODE_6_length_33354_cov_573.119090:1-33227
Writing updated NODE_6_length_33354_cov_573.119090_pilon to phage_J-1.fasta
Mean frags coverage: 1264
Mean total coverage: 1264


-----------


Pilon version 1.23 Mon Nov 26 16:04:05 2018 -0500
Genome: phage_J-2_genomeseq_edited.fasta
Fixing snps, indels, gaps, local
Input genome size: 31128
Scanning BAMs
phage_J-2_bwa_phagereads.bam.edited: 33068 reads, 0 filtered, 33068 mapped, 32806 proper, 224 stray, FR 99% 354+/-1261, max 4138
frags
Processing NODE_9_length_31255_cov_101.420811:1-31128
frags phage_J-2_bwa_phagereads.bam.edited: coverage 220
Total Reads: 33068, Coverage: 220, minDepth: 22
Confirmed 30956 of 31128 bases (99.45%)
Corrected 0 snps; 0 ambiguous bases; corrected 0 small insertions totaling 0 bases, 0 small deletions totaling 0 bases
# Attempting to fix local continuity breaks
# fix break: NODE_9_length_31255_cov_101.420811:15079-15083 0 -0 +0 NoSolution
NODE_9_length_31255_cov_101.420811:1-31128 log:
Finished processing NODE_9_length_31255_cov_101.420811:1-31128
Writing updated NODE_9_length_31255_cov_101.420811_pilon to phage_J-2.fasta
Mean frags coverage: 220
Mean total coverage: 220


-----------


Pilon version 1.23 Mon Nov 26 16:04:05 2018 -0500
Genome: phage_M_genomeseq_edited.fasta
Fixing snps, indels, gaps, local
Input genome size: 39745
Scanning BAMs
phage_M_bwa_phagereads.bam.edited: 65133 reads, 0 filtered, 65133 mapped, 64920 proper, 136 stray, FR 100% 327+/-1043, max 3457 frags
Processing NODE_20_length_39872_cov_136.182740:1-39745
frags phage_M_bwa_phagereads.bam.edited: coverage 320
Total Reads: 65133, Coverage: 320, minDepth: 32
Confirmed 39745 of 39745 bases (100.00%)
Corrected 0 snps; 0 ambiguous bases; corrected 0 small insertions totaling 0 bases, 0 small deletions totaling 0 bases
NODE_20_length_39872_cov_136.182740:1-39745 log:
Finished processing NODE_20_length_39872_cov_136.182740:1-39745
Writing updated NODE_20_length_39872_cov_136.182740_pilon to phage_M.fasta
Mean frags coverage: 320
Mean total coverage: 320


-----------


Pilon version 1.23 Mon Nov 26 16:04:05 2018 -0500
Genome: phage_P_genomeseq_edited.fasta
Fixing snps, indels, gaps, local
Input genome size: 33499
Scanning BAMs
phage_P_bwa_phagereads.bam.edited: 120220 reads, 0 filtered, 120220 mapped, 119611 proper, 396 stray, FR 100% 331+/-1026, max 3409
frags
Processing NODE_46_length_33626_cov_295.842503:1-33499
frags phage_P_bwa_phagereads.bam.edited: coverage 681
Total Reads: 120220, Coverage: 681, minDepth: 68
Confirmed 33499 of 33499 bases (100.00%)
Corrected 0 snps; 0 ambiguous bases; corrected 0 small insertions totaling 0 bases, 0 small deletions totaling 0 bases
NODE_46_length_33626_cov_295.842503:1-33499 log:
Finished processing NODE_46_length_33626_cov_295.842503:1-33499
Writing updated NODE_46_length_33626_cov_295.842503_pilon to phage_P.fasta
Mean frags coverage: 681
Mean total coverage: 681


-----------
```

Genome Similarity

Installed FastANI v1.3 using https://github.com/ParBLiSS/FastANI and boost/1.65.0 https://doi.org/10.1038/s41467-018-07641-9

example use:

```
fastANI -q genome1.fa -r genome2.fa -o output.txt
```

```
for i in *.fasta; do for j in *.fasta; do fastANI -q ${j} -r ${i} -o ${j}_${i}.out; done; done
cat *.out > ANI.results
rm -rf *.out
```

Note that FastANI only gives outputs if similarity is >80%. lower than that requires amino acid identity.

Phage Genome query	Phage genome subject	ANI estimate	Sequence Fragments aligned with orthologous matches	Total Sequence Fragments
Phage_P_genome.fasta	phage_J-1_genomeseq_edited.fasta	98.5221	10	11
phage_J-1_genomeseq_edited.fasta	Phage_P_genome.fasta	98.5422	10	11
Phage_C_genome.fasta	Phage_P_genome.fasta	98.6118	10	11
Phage_P_genome.fasta	Phage_C_genome.fasta	98.6335	8	11
Phage_D_genome.fasta	Phage_J-2_genome.fasta	98.9617	9	10
Phage_J-2_genome.fasta	Phage_D_genome.fasta	98.9629	9	10
Phage_C_genome.fasta	Phage_C_genome.fasta	100	11	11
Phage_C_genome.fasta	**phage_J-1_genomeseq_edited.fasta**	**100**	**10**	**11**
Phage_D_genome.fasta	Phage_D_genome.fasta	100	9	10
phage_J-1_genomeseq_edited.fasta	**Phage_C_genome.fasta**	**100**	**10**	**11**
phage_J-1_genomeseq_edited.fasta	phage_J-1_genomeseq_edited.fasta	100	11	11
Phage_J-2_genome.fasta	Phage_J-2_genome.fasta	100	9	10
Phage_M_genome.fasta	Phage_M_genome.fasta	100	13	13
Phage_P_genome.fasta	Phage_P_genome.fasta	100	11	11

Identifying the Terminase for Rearranging the Genome

I used prokka and prodigal to predict orfs and annotate the genes against the caudovirales ncbi database.

Prokka version 1.12.

Used the Caudovirales specific databases, downloaded 6th September 2019; http://s3.climb.ac.uk/ADM_share/Caudovirales.tar.gz

```
[jef11@bert prokka]$ wget http://s3.climb.ac.uk/ADM_share/Caudovirales.tar.gz
[jef11@bert prokka]$ tar -xzf Caudovirales.tar.gz
[jef11@bert prokka]$ prokka --setupdb
```

Use the pilon-checked fasta files and run this:

```
for i in *.fasta; do name=$(echo ${i} | cut -d "." -f1); prokka ${i} -outdir "prokka_"${name} --prefix ${name} --usegenus --genus
Caudovirales --centre X --compliant; done
```

Then put the gffs into Geneious (Geneious Prime® 2020.0.3; Build 2019-11-07 12:34; Java Version 11.0.4+11 (64 bit))

Then looked for the predicted terminase gene, found a logical breakpoint and set this as the new origin. Also reversed the sequence by selecting the reverse complement if the terminase was not forward. Almost all of the genomes had a gene overlapping the terminase gene (sometimes by ~4bp) or sometimes with a small gap between that and the terminase, then a larger gap before that one, so this was chose as the logical gap. Then the "first base of the start codon of the first gene downstream of the break point" was used - as per Russell's chapter (Russell D.A. (2018) Sequencing, Assembling, and Finishing Complete Bacteriophage Genomes. In: Clokie M., Kropinski A., Lavigne R. (eds) Bacteriophages. Methods in Molecular Biology, vol 1681. Humana Press, New York, NY, URL:https://link.springer.com/protocol/10.1007/978-1-4939-7343-9_9).

Note that M phage genome has two genes before it. This is because there was no GAGG (Shine-Delgarno sequence) before the gene immediately before the terminase, but there was before the gene in before that.

Now all the genomes are named Phage_X_reordered.

ORF Prediction

Glimmer version 3.02
Glimmer3 -

```
long-orfs -n -t 1.15 genom.seq run1.longorfs
extract -t genom.seq run1.longorfs > run1.train
build-icm -r run1.icm < run1.train
glimmer3 -o50 -g110 -t30 genom.seq run1.icm run1

for i in *.fasta; do long-orfs -n -t 1.15 ${i} ${i}.longorfs; done
for i in *.fasta; do extract -t ${i} ${i}.longorfs > ${i}.train; done
for i in *.fasta; do build-icm -r ${i}.icm < ${i}.train; done
for i in *.fasta; do glimmer3 -o50 -g110 -t30 ${i} ${i}.icm ${i}_glimmer; done
```

Use this to make the ORF predictions from glimmer:

Minimum gene length = 110 bp
Maximum overlap bases = 50

Threshold score = 30
Use first start codon = false
Start codons = atg,gtg,ttg

I am using these default settings because a previous paper didn't specify the settings, but they did use Glimmer.(Gilbert RA, Kelly WJ, Altermann E, et al. Toward Understanding Phage:Host Interactions in the Rumen; Complete Genome Sequences of Lytic Phages Infecting Rumen Bacteria. *Front Microbiol*. 2017;8:2340. Published 2017 Dec 5. doi:10.3389/fmicb.2017.02340 - https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5723332/)

Phage	number of CDS
C	57
D	52
J-1	57
J-2	50
M	77
P	57

Then made a gff for the glimmer output:

```
for i in *.predict; do awk 'BEGIN{OFS="\t"; i=1;}{split($4, a, ""); if ($1 ~ /^>/) {node=$1; gsub(">","",node); print "##gff-version 3"} else {print node, "glimmer3", "CDS", $2, $3, $5, a[1], a[2], "ID=gene00"i; i++}}' ${i} > ${i}.gff; done
```

GeneMarkS-2

Ran with prokaryotic sequence type, GFF3 output, genetic code 11. http://exon.gatech.edu/GeneMark/genemarks2.cgi, accessed 10/03/2020
Lomsadze A, Gemayel K, Tang S, Borodovsky M Modeling leaderless transcription and atypical genes results in more accurate gene prediction in prokaryotes. *Genome Res*, 2018, 29(7), pp 1079-1089

Phage	number of CDS	Info
C	49	job ID = genemarks2.20200311.115800.15536 ; Estimated run time: 1 second(s)
D	50	job ID = genemarks2.20200311.115821.15943 ; Estimated run time: 1 second(s)
J-1	49	job ID = genemarks2.20200311.115845.18732 ; Estimated run time: 1 second(s)
J-2	49	job ID = genemarks2.20200311.115904.2359 ; Estimated run time: 1 second(s)
M	71	job ID = genemarks2.20200311.115922.18738 ; Estimated run time: 1 second(s)
P	50	job ID = genemarks2.20200311.115940.15618 ; Estimated run time: 1 second(s)

Prodigal

Prodigal V2.6.3: February, 2016

Ran with default settings apart from output format as gff:

```
for i in *.fasta; do name=$(echo ${i} | cut -d "." -f1); prodigal -f gff -i ${i} -o ${name}_prodigal.gff; done
```

Phage	number of CDS
C	51
D	51
J-1	51
J-2	49
M	73
P	53

PHANOTATE

version - 1.1.2 using python 3.5.4

```
for i in *.fasta; do python3 ~/bin/phanotate.py -o ${i}_phanotate.tab ${i}; done
```

ls

Phage	number of ORFS
C	65
D	57
J-1	65
J-2	55

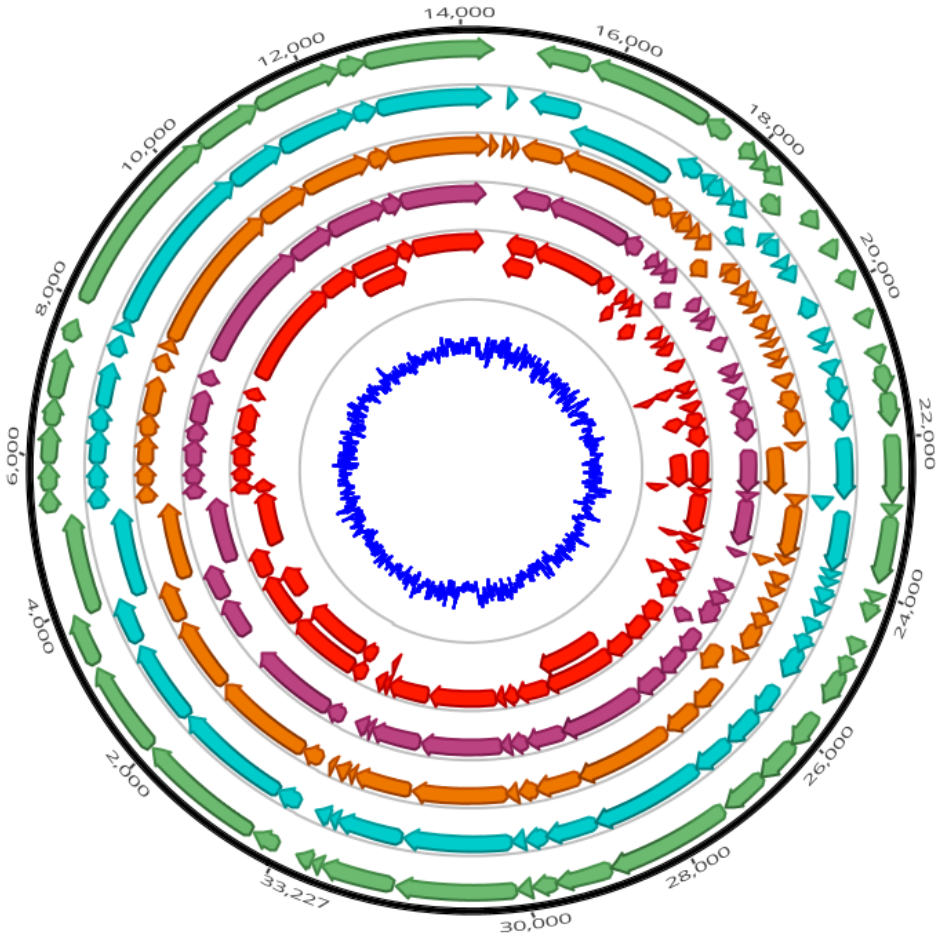
M	85
P	65

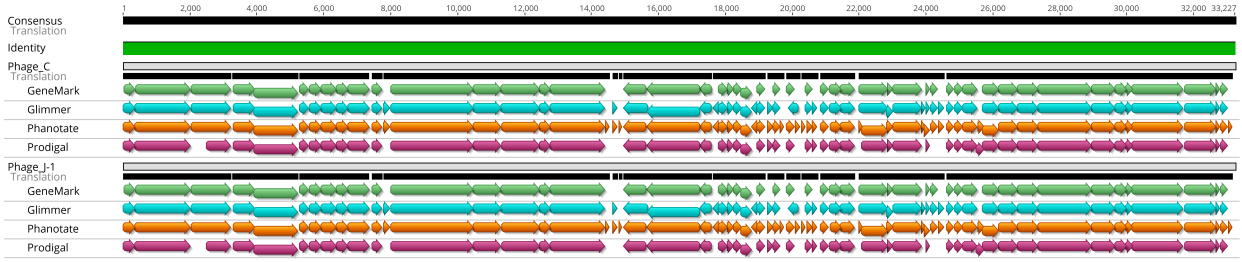
Make the gffs from these output files too:

```
for i in *.tab; do name=$(echo ${i} | cut -d "_" -f1,2); echo "##gff-version 3" > ${name}_phanotate.gff; awk 'BEGIN{FS=OFS="\t";
i=1;}{if ($1 ~ /^#/) {print $0} else {print $4, "Phanotate_v1.2.2", "CDS", $1, $2, $5, $3, ".", "ID=gene_00"i; i++}}' ${i} >>
${name}_phanotate.gff; done
```

Phage	Phanotate	Prodigal	GeneMarkS-2	Glimmer
C	65	52	49	57
D	57	50	50	52
J-1	65	53	49	57
J-2	55	50	49	50
M	85	73	70	77
P	65	55	50	57

I copied these gff files into Geneious, added all orfs to a track - one track per gene caller, and made a consensus track. This had all orfs that fit into the categoires below.
Note that it was at this point that phage C and J-1 were noticed to be completely syntenous, and a pairwise alignment using MUSCLE in Geneious with default settings showed 100% identity.





Opened them into Geneious, and counted how many genes fell into these categories:
A) all gene callers agreed with ORF presence, start and end.
B) all gene callers agreed ORF presence, but start and end varied for some gene callers.
C) 3/4 gene callers agreed ORF presence, start and end.
D) 3/4 gene callers agreed ORF presence, different start and end.

	A	B	C	D
C	33	27	5	0
D	41	17	1	0
J-1	35	25	4	0
J-2	41	15	0	0
M	52	34	4	2
P	27	40	5	2

Now I can export these annotations as a gff and edit them in excel- remove all orfs not from the consensus track, and add in a name - orf_1 and if two orfs are from category B or D then they have the same orf_ID number.

Make the relevant peptide and gene files
First copy all the "*categories.gff" files onto HPC- excluding J-1.

I can check for the "correct ORF by blasting these genes against a database and find the "best" - ie this one with the higher bitscore and percentage identity.

```
/ibers/ernie/home/jef11/mar2020_phage_genomes/7_orf_prediction/make_faa_files
for i in *.gff; do name=$(echo ${i} | cut -d "." -f1); gff2bed < ${i} > ${name}.bed; done
for i in *.fasta; do name=$(echo ${i} | cut -d "_" -f1,2); bedtools getfasta -fi ${i} -bed ${name}*bed -name >
${name}_categories.ffn; done
for i in *.ffn; do name=$(echo ${i} | cut -d "." -f1); transeq ${i} ${name}.faa; done
```

gff2bed, part of bedops -version: 2.4.37 (typical)
(http://bioinformatics.oxfordjournals.org/content/28/14/1919.abstract; https://doi.org/10.1093/bioinformatics/bts277)
bedtools Version: v2.27.1
emboss v6.6.0.0

Databases for Annotating:

Name	Size	Last Modified	link
ref_viruses_rep_genomes.tar.gz	101857 KB	23/08/2019 20:05:00	ftp://ftp.ncbi.nlm.nih.gov/blast/db/ref_viruses_rep_genomes.tar.gz
swissprot		08-2019 release.	(ftp://ftp.uniprot.org/pub/databases/uniprot/current_release/knowledgebase/complete/uniprot_sprot.fasta.gz)
viral refseq - with protein, genomic and non-redundant proteins		11/07/2019 22:05:00	ftp://ftp.ncbi.nlm.nih.gov/refseq/release/viral/
taxdb	8855 KB	05/09/2019 04:00:00	ftp://ftp.ncbi.nlm.nih.gov/blast/db/taxdb.tar.gz
caudovirales		July 20th 2017	http://s3.climb.ac.uk/ADM_share/Caudovirales.tar.gz
Eggnog 5 viruses		11th July 2018	(http://eggnog5.embl.de/download/eggnog_5.0/e5.viruses.faa)

As well as the pVOGs, which are from the multiphate v.1.0 that Carol Zhou made from the pVOGs database.

I did:

```
cat viral.1.protein.faa viral.2.protein.faa >viral.protein.faa
makeblastdb -in viral.protein.faa -out viral.protein -dbtype prot

cat viral.1.1.genomic.fna  viral.2.1.genomic.fna > viral.genomic.fna
makeblastdb -in viral.genomic.fn -out viral.genomic -dbtype nuc
```

The path to the databases are

~/bin/databases/
then either NCBI_virus_databases or pVOGS:

viral.protein
viral.genomic
ref_viruses_rep_genomes
Caudovirales
viral.nonredundant_protein.1.protein.faa
pVOGs.faa

Searching databases:
using BLAST to search databases - mostly blastp with no restrictions but then filtering afterwards.
BLAST 2.8.1+

```
for i in *annotations; do cd ${i}; subheader.sh ${i} 10G > ${i}.sub; cd ..; done

for i in *annotations;  do name=$(echo ${i} | cut -d "_" -f1,2); echo blastn -db viral.genomic -query ${name}_categories.ffn -out
${name}.ffn.viral.genomic.out -outfmt \"6 std qlen slen qcovs qcovhsp stitle\" >> ${i}/${i}.sub;  done

for i in *annotations;  do name=$(echo ${i} | cut -d "_" -f1,2); echo blastn -db viral.genomic -query ${name}_reordered.fasta -out
${name}.fasta.viral.genomic.out -outfmt \"6 std qlen slen qcovs qcovhsp stitle\" >> ${i}/${i}.sub;  done

for i in *annotations; do name=$(echo ${i} | cut -d "_" -f1,2); echo blastn -db ref_seq_rep_genomes -query ${name}_categories.ffn -
out ${name}.ffn.refseq_viralgenomes.out -outfmt \"6 std qlen slen qcovs qcovhsp stitle\" >> ${i}/${i}.sub;  done

for i in *annotations; do name=$(echo ${i} | cut -d "_" -f1,2); echo blastn -db ref_seq_rep_genomes -query ${name}_reordered.fasta -
out ${name}.fasta.refseq_viralgenomes.out -outfmt \"6 std qlen slen qcovs qcovhsp stitle\" >> ${i}/${i}.sub;  done

for i in *annotations; do name=$(echo ${i} | cut -d "_" -f1,2); echo blastp -db viral.protein -query ${name}_categories.faa -out
${i}.viral.protein.out -outfmt \"6 std qlen slen qcovs qcovhsp stitle\" >> ${i}/${i}.sub;  done

for i in *annotations;  do name=$(echo ${i} | cut -d "_" -f1,2); echo blastp -db Caudovirales -query ${name}_categories.faa -out
${i}.Caudovirales.out -outfmt \"6 std qlen slen qcovs qcovhsp stitle\" >> ${i}/${i}.sub;  done

for i in *annotations;  do name=$(echo ${i} | cut -d "_" -f1,2); echo blastp -db viral.nonredundant_protein.1.protein.faa -
query ${name}_categories.faa -out ${i}.viralnrprot.out -outfmt \"6 std qlen slen qcovs qcovhsp stitle\" >> ${i}/${i}.sub;  done

for i in *annotations; do name=$(echo ${i} | cut -d "_" -f1,2); echo blastp -db pVOGs.faa -query ${name}_categories.faa -out
${i}.pvogs.out -outfmt \"6 std qlen slen qcovs qcovhsp stitle\" >> ${i}/${i}.sub;  done

for i in *annotations; do name=$(echo ${i} | cut -d "_" -f1,2); echo blastp -db eggnog5_viruses -query ${name}_categories.faa -out
${i}.eggnog.out -outfmt \"6 std qlen slen qcovs qcovhsp stitle\" >> ${i}/${i}.sub;  done

for i in *annotations; do name=$(echo ${i} | cut -d "_" -f1,2); echo blastp -db swissprot -query ${name}_categories.faa -out
${i}.swissprot.out -outfmt \"6 std qlen slen qcovs qcovhsp stitle\" >> ${i}/${i}.sub;  done
```

This took less than 0.5G and ~4-6 mins each.

Filtering
I'm going to filter using an evalue of <10^-5 (0.00001) (as per Aziz chapter - Methods Mol Biol. 2018;1681:197-215. doi: 10.1007/978-1-4939-7343-9_15), query coverage of 80% (as per prokka), check that the lengths of the proteins are similar (80%) and then if sequence similarity is >30% assign homology, if less than 30 say putative, something done previously (see quote below)- which usually what is used as a cut off for homology according to Pearson (2014, doi: 10.1002/0471250953.bi0301s42)

Take all out files from all of the databases and sort by the query.
Filter using evalue <0.0001 and qcovhsp >80 (as per prokka).
Get just the top hit for each orf irrelevant of database.
If equally good bitscore - report all:

```
cat *.out | sort -Vk1,12 | awk '{if ($11 <= 0.00001 && $16 >=80 ) print $0}' | awk 'BEGIN {prev=""; score=""} {if ($1 != prev) {print
$0; prev=$1; score=$12} else if ($1 == prev  && $12 >= score) {print $0; score=$12}}' > Phage_C_tophits_alldb.tab
```

" Peptide sequences for each predicted ORF also underwent homology searches using BLASTp against nr, PhAnTOME [41], pVOGs [42] and the PHASTER Prophage/Virus databases [27]. The following threshold values were applied in general. Putative ORFs with 50–70% sequence identity [43] to a given gene were assigned "putative." When peptide sequences exhibited low identity (less than 50% [44]), protein sequences were also submitted for the analysis of hidden Markov models by hmmscan [45] against the Pfam database [46] and NCBI's Conserved Domain Database. Consensus gene functions were assigned to ORFs manually" Philipson et al, *Viruses* **2018**, *10*(4), 188; https://doi.org/10.3390/v10040188

Protein Motif Searching using HMMER:
hmmer version HMMER 3.1b2 (February 2015); http://hmmer.org/

databases:
pfam, tigrfam, hamap

pfams - wget ftp://ftp.ebi.ac.uk/pub/databases/Pfam/current_release/Pfam-A.hmm.gz - last modified 30/08/2018
TIGRfams - https://ftp.ncbi.nlm.nih.gov/hmm/TIGRFAMs/release_15.0/TIGRFAMs_15.0_HMM.tar.gz Current Release: 15.0, last modified: 2018-06-19
HAMAP (already in prokka)
eggnog v5 caudovirales hmms - http://eggnog5.embl.de/download/eggnog_5.0/per_tax_level/28883/28883_hmms.tar 02-Mar-2019

I then used hmmpress to make the relevant files from the hmms (tigr fam ones were cat together into one).

```
for i in *annotations; do cd ${i}; subheader.sh ${i}_hmms 10G > ${i}_hmms_longformat.sub; cd ..; done
for i in $(ls ~/bin/prokka/db/hmm/*.hmm); do hmm=$(echo ${i} | cut -d "/" -f10); for j in *annotations; do name=$(echo ${j} | cut -d
"_" -f1,2); echo hmmscan --cpu 1 -o ${j}_${hmm}_longformat.out ${i} ${name}_categories.faa >> ${j}/${j}_hmms_longformat.sub; done;
done
```

This took <5Gb and around 1-2 minutes each.

Ignore anything above 10^-4 (Aziz chapter - Methods Mol Biol. 2018;1681:197-215. doi: 10.1007/978-1-4939-7343-9_15) in the HMMS (<0.0001)

Go through this by hand for each genome, and choose the suitable ORFs that match these criteria:

- Majority of the ORF finders agree
- ORF shares homology with protein in database above the thresholds
- overlap is 4bp with previous ORF
- ORF is in frame with previous ORF
- If unclear, state in notes of GFF that there is an alternative start site predicted by which gene callers.

Now that this is done the gffs can be finalised and used to make the new faa files - these can be searched for transmembrane regions and signal peptides.

Finding tRNA
I ran tRNAscan using the online tool:
tRNAscan-SE v2.0: http://lowelab.ucsc.edu/cgi-bin/tRNAscan-SE2.cgi (24/03/2020)

- Lowe, T.M. and Chan, P.P. (2016) tRNAscan-SE On-line: Search and Contextual Analysis of Transfer RNA Genes. Nucl. Acids Res. 44: W54-57.
- Chan, P.P., Lin, B., and Lowe, T.M. tRNAscan-SE 2.0. (In Preparation)

I used default settings and bacterial sequence source.
The results were copied here and added to the gff:

```
Sequence               tRNA    Bounds  tRNA    Anti    Intron Bounds   Inf     Isotype Isotype
Name            tRNA #  Begin   End     Type    Codon   Begin   End     Score   CM      Score   Note
--------        ------  -----   ------  ----    -----   -----   ----    ------  ------- ------- ------
Phage_C         1       20253   20325   Gln     TTG     0       0       68.7    Gln     73.0
>Phage_C.trna1-GlnTTG (20253-20325)  Gln (TTG) 73 bp  Sc: 68.7
AACGGTGTAGTGAAGTGGTTAACACATCAGATTTTGACTCTGAAATACGCGGGTTCAAATCCCGCCGCCGTTG
```

0 tRNA for phage_D
0 tRNA for phage J-2
0 tRNA for phage M

```
Sequence               tRNA    Bounds  tRNA    Anti    Intron Bounds   Inf     Isotype Isotype
Name            tRNA #  Begin   End     Type    Codon   Begin   End     Score   CM      Score   Note
--------        ------  -----   ------  ----    -----   -----   ----    ------  ------- ------- ------
Phage_P         1       20517   20588   Gln     TTG     0       0       64.5    Gln     71.3
>Phage_P.trna1-GlnTTG (20517-20588)  Gln (TTG) 72 bp  Sc: 64.5
GCGGCGTTGTGAAGTGGTTAACACATCAGATTTTGATTCTGAAATACGCGGGTTCAAATCCCGCCGCCGCCG
```

These results were added into the gff files.

Use these commands again to make the .faa files:

```
for i in *.gff; do name=$(echo ${i} | cut -d "." -f1); gff2bed < ${i} > ${name}.bed; done
for i in *.fasta; do name=$(echo ${i} | cut -d "_" -f1,2); bedtools getfasta -s -fi ${i} -bed ${name}*bed -name | tr " " "_" | sed
"s/^>/>${name}_/"  > ${name}.ffn; done
for i in *.ffn; do name=$(echo ${i} | cut -d "." -f1); transeq ${i} ${name}.faa -table 11; done
```

Report the current EMBOSS version number v6.6.0.0
bedtools v2.27.1
gff2bed version: 2.4.36

Transmembrane regions
Used tmhmm - (TMHMM2.0)

- Anders Krogh and Bjorn Larsson, Gunnar von Heijne, and Erik L.L. Sonnhammer: Predicting Transmembrane Protein Topology with a Hidden Markov Model: Application to Complete Genomes. J. Mol. Biol. 305:567-580, 2001.)

```
for i in *.faa; do name=$(echo ${i} | cut -d "." -f1); tmhmm ${i} > ${name}_tmhmm.out; done
```

Identifying Phage family - ClassiPhage

- Chibani CM, Farr A, Klama S, Dietrich S, Liesegang H. Classifying the Unclassified: A Phage Classification Method. *Viruses*. 2019;11(2):195. Published 2019 Feb 24. doi:10.3390/v11020195

I downloaded the hmms for the families from here (http://appmibio.uni-goettingen.de/index.php?sec=sw, accessed 24/03/2020)
I then cat all the hmms together and made a database to search for each of the families and each of the phage genomes (the proteins):

```
cat *.hmm > *.hmm.db.hmm
hmmpress  *.hmm.db.hmm

for i in *.faa; do name=$(echo ${i} | cut -d "." -f1); for j in *db.hmm; do lib=$(echo ${j} | cut -d "_" -f1); hmmscan --tblout
${name}_{lib}.out ${j} ${i} ; done; done
```

Results:
Below is a table of the number of genes that matched the hmm profile for that phage family, where the this were <0.0001 (10^-4)

Phage	Myo	Podo	Sipho	Ino
C	1	1	0	0

D	0	0	0	0
J-2	0	0	0	0
M	6	2	0	0
P	1	1	0	0

PHACTS - to find out the lifestyle of the phage.
I used the online tool (http://edwards.sdsu.edu/PHACTS/upload.php on 24/03/2020)

- Katelyn McNair, Barbara A. Bailey, Robert A. Edwards, PHACTS, a computational approach to classifying the lifestyle of phages, *Bioinformatics*, Volume 28, Issue 5, 1 March 2012, Pages 614–618, https://doi.org/10.1093/bioinformatics/bts014

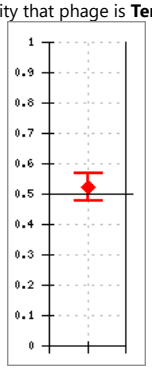
Phage C_Log for job RID: 15850666091; non-confidently Temperate, non-confidently Gram Positive host
Phage D Log for job RID: 15850675991; confidently temperate, confidently Gram positive host
Phage J-2 Log for job RID: 15850675894; confidently temperate, confidently Gram positive host
Phage M Log for job RID: 15850676433; non-confidently lytic; confidently Gram positive
Phage P Log for job RID: 15850676433; non-confidently lytic; confidently Gram positive.

PHACTS output:

Phage C
Lifestyle:

Analysis Statistics

Probability that phage is **Temperate**



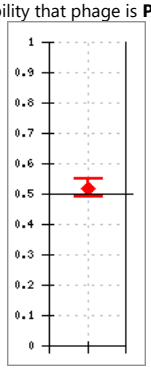
Ten iterations of PHACTS were performed using the default settings. The phage was **non-confidently** predicted as having a **Temperate lifestyle**.

Predicted Class	Averaged Probability	Standard Deviation
Temperate	0.522	0.046

Gram-stain of host:

Analysis Statistics

Probability that phage is **Positive**



Ten iterations of PHACTS were performed using the default settings. The phage was **non-confidently** predicted as infecting a **Gram Positive** host.

Predicted Class	Averaged Probability	Standard Deviation
Positive	0.52	0.03

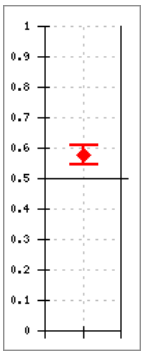
Phage D
Log for job RID: 15850675991
Lifestyle:

Analysis Statistics

Probability that phage is **Temperate**

Ten iterations of PHACTS were performed using the default settings. The phage was **confidently** predicted as having a **Temperate lifestyle**.

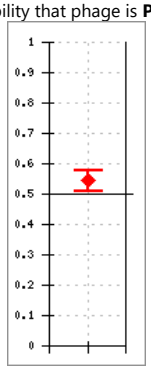
Predicted Class	Averaged Probability	Standard Deviation
Temperate	0.576	0.031



Gram-stain of host:

Analysis Statistics

Probability that phage is **Positive**



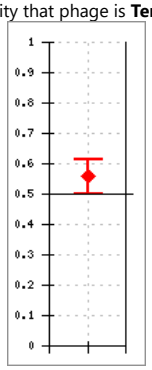
Ten iterations of PHACTS were performed using the default settings. The phage was **confidently** predicted as infecting a **Gram Positive** host.

Predicted Class	Averaged Probability	Standard Deviation
Positive	0.544	0.035

Phage J-2
Log for job RID: 15850675894
Lifestyle:

Analysis Statistics

Probability that phage is **Temperate**



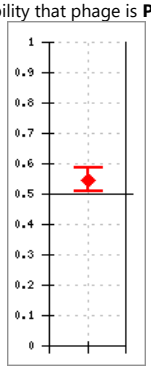
Ten iterations of PHACTS were performed using the default settings. The phage was **confidently** predicted as having a **Temperate lifestyle**.

Predicted Class	Averaged Probability	Standard Deviation
Temperate	0.558	0.056

Gram-stain of host:

Analysis Statistics

Probability that phage is **Positive**



Ten iterations of PHACTS were performed using the default settings. The phage was **confidently** predicted as infecting a **Gram Positive** host.

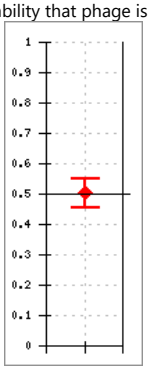
Predicted Class	Averaged Probability	Standard Deviation
Positive	0.547	0.038

Phage M

Log for job RID: 15850676433
Lifestyle:

Analysis Statistics

Probability that phage is **Lytic**



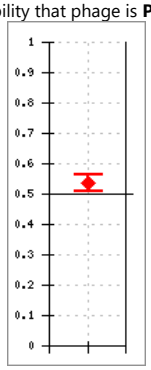
Ten iterations of PHACTS were performed using the default settings. The phage was **non-confidently** predicted as having a **Lytic lifestyle**.

Predicted Class	Averaged Probability	Standard Deviation
Lytic	0.503	0.046

Gram-stain of host:

Analysis Statistics

Probability that phage is **Positive**



Ten iterations of PHACTS were performed using the default settings. The phage was **confidently** predicted as infecting a **Gram Positive** host.

Predicted Class	Averaged Probability	Standard Deviation
Positive	0.536	0.027

Phage P

Log for job RID: 15850676433

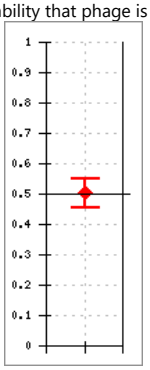
Results Page

Log for job RID: 15850676433

Lifestyle:

Analysis Statistics

Probability that phage is **Lytic**



Ten iterations of PHACTS were performed using the default settings. The phage was **non-confidently** predicted as having a **Lytic lifestyle**.

Predicted Class	Averaged Probability	Standard Deviation
Lytic	0.503	0.046

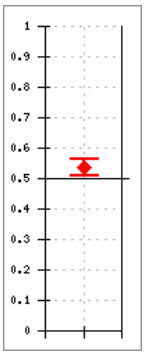
Gram-stain of host:

Analysis Statistics

Probability that phage is **Positive**

Ten iterations of PHACTS were performed using the default settings. The phage was **confidently** predicted as infecting a **Gram Positive** host.

Predicted Class	Averaged Probability	Standard Deviation
Positive	0.536	0.027



PhagePromoter
Using Galaxy tool (https://galaxy.bio.di.uminho.pt/)

- Predicting promoters in phage genomes using PhagePromoter; Marta Sampaio, Miguel Rocha, Hugo Oliveira, Oscar Dias; Bioinformatics, Volume 35, Issue 24, 15 December 2019, Pages 5301–5302, https://doi.org/10.1093/bioinformatics/btz580

I used the following settings:

phage	file format	phage family	Host bacteria genus	phage type	threshold	Strandedness
C	fasta	siphoviridae	other	temperate	0.5	both
D	fasta	siphoviridae	other	temperate	0.5	both
J-1	fasta	siphoviridae	other	temperate	0.5	both
M	fasta	siphoviridae	other	lytic	0.5	both
P	fasta	siphoviridae	other	lytic	0.5	both

Then I downloaded the fasta files, made gff files from them and added them to Geneious. Then by hand, for each promoter in each genome, kept only those promoters ithat were either n intergenic regions in roughly the 3' ends of previous genes roughly a couple of hundred bp away form the next gene and in the corresponding direction. (Aziz chapter)
Used this to make the gffs from the fasta files.

```
for i in phage_*_phagepromoter_output.fasta; do name=$(echo ${i} | cut -d "_" -f1,2); grep ">" ${i} | sed 's/host
complement(/hostcomplement (/g' | sed 's/phage complement(/phagecomplement (/g' | awk 'BEGIN{FS=" "; OFS="\t"}{split($1, chr, ":");
print chr[1], "PhagePromoter", "promoter", $3,$4, "+", ".", "Note=promoter predicted by PhagePromoter;"$2}' | tr -d "(" | tr -d ")" |
sed 's/\.\./\t/g' | sed 's/score=//g' | awk 'BEGIN{FS=OFS="\t"}{if ($9 ~ /complement/) $7="-";print}' | tr -d ">" >
${name}_promoter.gff; done
```

Inverted repeats
searched for using einverted within emboss version 6.6.0.0:
I used default settings for all.

```
for i in *.fasta; do name=$(echo ${i} | cut -d "_" -f1,2); einverted ${i} -outseq ${name}_inv.fa; done
```

```
Phage_C: Score 52: 36/50 ( 72%) matches, 0 gaps
   18846 tgtaacattgtgcaacttttaataaaataaatgttgcaatatgcaacaat 18895
         |||  |||  |||||| |||| |  |  ||||||||||  |||  |||||
   19764 acaacgtataacgttgtaaataaaatattttacaacgtgttacaatgtta 19715


Phage_C: Score 59: 29/36 ( 80%) matches, 0 gaps
   27713 tttcgagattgcacagcgttaacagatattgatttg 27748
         ||||| ||||   |||| |||||| || ||||||||
   29306 aaagcactaaaaagtcgtaattgtttacaactaaac 29271
>Phage_C_18846_18895
tgtaacattgtgcaacttttaataaaataaatgttgcaatatgcaacaat
>Phage_C_19715_19764
attgtaacattgtgcaacattttataaaataaatgttgcaatatgcaaca
>Phage_C_27713_27748
tttcgagattgcacagcgttaacagatattgatttg
>Phage_C_29271_29306
caaatcaacatttgttaatgctgaaaaatcacgaaa


Phage_M: Score 52: 20/22 ( 90%) matches, 0 gaps
   16617 atttcacaatattttcggttac 16638
         |||| |||||||||||  |||||
   16839 taaactgttataaaagacaatg 16818
>Phage_M_16617_16638
atttcacaatattttcggttac
>Phage_M_16818_16839
gtaacagaaaatattgtcaaat


Phage_P: Score 53: 35/45 ( 77%) matches, 1 gaps
   19104 acattgtgcaacattttataaaat-aaatgttgcaatatgcaacaa 19148
         || ||  |||||||| |||| | | ||||||||||  |||  ||||
   20466 tgcaaagcgttgtaatatatgtaattttacaacgtgttacaatgtt 20421
```

```
Phage_P: Score 50: 18/19 ( 94%) matches, 0 gaps
   27293 tgtcaagttttgatacatc 27311
         |||||| ||||||||||||
   28163 acagtttaaaactatgtag 28145
>Phage_P_19104_19148
acattgtgcaacattttataaaataaatgttgcaatatgcaacaa
>Phage_P_20421_20466
ttgtaacattgtgcaacattttaatgtatataatgttgcgaaacgt
>Phage_P_27293_27311
tgtcaagttttgatacatc
>Phage_P_28145_28163
gatgtatcaaaatttgaca
```

Interesting papers for this:
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5941160/
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC92705/

Terminators
To find Terminators, I used FindTerm online (v2.8.1, http://www.softberry.com/berry.phtml?topic=findterm&group=programs&subgroup=gfindb)

- ***Reference:*** V. Solovyev, A Salamov (2011) Automatic Annotation of Microbial Genomes and Metagenomic Sequences. In Metagenomics and its Applications in Agriculture, Biomedicine and Environmental Studies (Ed. R.W. Li), Nova Science Publishers, p. 61-78

I used default settings apart from showing results <-10 and showing all putative terminators as suggested in the Aziz chapter. Then those that were in intergenic regions were annotated, excluding those that occur in the middle of genes. The score was annotated onto each terminator.

Phylogenenomics and Phylogenetics:
Most common phage name with the number of best hit homologous proteins:

Phage	
C	
D	Paenibacillus phage PG1
J-2	Paenibacillus phage PG1
M	Clostridium phage phiCP13O, Clostridium phage phiCP26F
P	

Generated a proteomic tree using ViPTree, with the settings dsDNA, Prokaryote host categories, and prokaryote (table 11) for genetic code.
From the resulting tree from the ViPTree analysis, the closest related taxa were identified as those in the same clade as the Butyrivibrio phage genomes, of which there were 13. downloaded these sequences from NCBI using the ID supplied by ViPTree. The nucleotide sequences of these 13 genomes, along with the five phage genomes from this study, were then analyzed using the Genome-BLAST Distance Phylogeny (GBDP) method (Meier-Kolthoff et al., 2013) with default settings recommended for prokaryotic viruses in VICTOR and use the D0 formula. (D0 is the optimum formula for nucleotides, D6 for amino acids and D4 for incomplete genomes - according to the Victor paper - https://academic.oup.com/bioinformatics/article/33/21/3396/3933260).

NCBI 3071:
I downloaded the 3071 genome from the NCBI (https://www.ncbi.nlm.nih.gov/nuccore/1120532029?report=fasta)
Butyrivibrio fibrisolvens DSM 3071, whole genome shotgun sequence; NCBI Reference Sequence: NZ_FQXK01000003.1
GC Content / Codon Usage
Using Geneious to have a look at codon usage and GC content:

		Bacterial host			Arian			Bo-Finn			Arawn			Idris			Ceridwen		
Amino Acid	AA	AA	Freq	%	AA	Freq	%	AA	Freq	%	AA	Freq	%	AA	Freq	%	AA	Freq	%
Alanine	A	W:	1,182	1.0%	C	123	1.2%	W	143	1.3%	M	116	1.1%	C	109	1.1%	W	226	1.7%
Cysteine	C	C:	1,721	1.4%	W	119	1.2%	M	200	1.8%	W	130	1.3%	W	107	1.1%	C	318	2.4%
Aspartic Acid	D	H:	2,003	1.7%	H	142	1.4%	C	241	2.2%	C	203	2.0%	H	137	1.4%	H	333	2.5%
Glutamic Acid	E	Q:	3,246	2.7%	P	259	2.5%	P	285	2.6%	F	220	2.1%	M	222	2.3%	P	332	2.5%
Phenylalanine	F	M:	3,607	3.0%	M	267	2.6%	H	297	2.7%	P	234	2.3%	P	253	2.6%	M	357	2.7%
Glycine	G	P:	3,643	3.0%	Q	372	3.6%	D	348	3.1%	D	289	2.8%	F	283	2.9%	F	416	3.1%

Histidine	**H**	R:	4,700	3.9%	F	398	3.9%	E	375	3.4%	H	289	2.8%	R	413	4.3%	D	426	3.2%
Isoleucine	**I**	F:	5,227	4.3%	Y	421	4.1%	Y	387	3.5%	N	360	3.5%	Q	433	4.5%	Y	539	4.1%
Lysine	**K**	Y:	5,491	4.5%	R	428	4.2%	Q	489	4.4%	I	374	3.6%	Y	446	4.6%	N	581	4.4%
Leucine	**L**	N:	5,799	4.8%	S	550	5.4%	S	522	4.7%	V	370	3.6%	S	464	4.8%	V	622	4.7%
Methionine	**M**	T:	6,578	5.4%	V	590	5.8%	V	581	5.2%	E	392	3.8%	N	476	4.9%	Q	645	4.9%
Asparagine	**N**	S:	7,729	6.4%	E	620	6.1%	F	586	5.3%	Y	478	4.6%	D	574	6.0%	G	665	5.0%
Proline	**P**	D:	7,941	6.6%	N	676	6.6%	G	598	5.4%	Q	557	5.4%	G	611	6.3%	A	672	5.1%
Glutamine	**Q**	V:	7,990	6.6%	D	683	6.7%	L	621	5.6%	G	582	5.6%	I	615	6.4%	E	679	5.1%
Arginine	**R**	G:	8,082	6.7%	G	683	6.7%	N	681	6.1%	K	587	5.7%	T	642	6.7%	T	709	5.4%
Serine	**S**	A:	8,588	7.1%	L	698	6.8%	T	725	6.5%	S	709	6.8%	V	640	6.7%	I	818	6.2%
Threonine	**T**	E:	8,760	7.2%	K	712	7.0%	A	741	6.7%	T	825	8.0%	K	696	7.2%	S	900	6.8%
Valine	**V**	K:	8,777	7.3%	I	752	7.4%	K	859	7.8%	L	912	8.8%	E	718	7.5%	K	998	7.5%
Tryptophan	**W**	I:	9,504	7.9%	T	752	7.4%	I	875	7.9%	A	933	9.0%	L	788	8.2%	L	1,145	8.6%
Tyrosine	**Y**	L:	9,986	8.3%	A	909	8.9%	R	925	8.4%	R	1,271	12.3%	A	949	9.9%	R	1,183	8.9%

Predicting Prophages in the Host Using Phaster

I ran the genome through PHASTER with the contigs option.
Two prophages found: 6kb region incomplete prophage with 7 proteins and 7.3kb region incomplete prophage with 10 proteins all complement.