**Comparative genomics final report:ORF predictor FuMiSh**

*by Shuhan Xu, Fuqi Xu, Milda Valiukonytė*

In this project, we performed three types of investigations our five genomes: analysis of genetic composition (GC, nucleotide, dinucleotide, amino acid and diamino acid frequencies), prediction of open reading frames (ORFs) and calculation of distances between genomes to understand their evolutionary relationships. We wrote Python scripts for these investigations. Details about our python scripts can be found at the “Python Scripts” at the end of the report.

The five genomes includes four prokaryotes and one eukaryote: *Escherichia coli* of *Enterobacteriaceae* bacterial family (genome length 5443340 bp), *Streptomyces coelicolor* of *Streptomycetaceae* family (9054847 bp), *Rubrobacter xylanophilus* of *Rubrobacteraceae* family (3225748 bp), *Spiribacter curvatus* of *Ectothiorhodospiraceae* family (1926631 bp), and *Saccharomyces cerevisiae* from *Saccharomycetaceae* yeastfamily (1531933 bp).

1. **Genetic composition analysis**

The genetic composition analysis showed that there is a great variation among the five genomes regarding their GC, nucleotide, dinucleotide, amino acid and diamino acid frequencies. The nucleotide analyses were performed on the genomes given to us while the amino acid analyses were performed on the proteome predicted by our ORF predictor. The GC content was calculated using the formula:

.

Both *S.coelicolor* and *R.xylanophilus* have high GC content (0.72 and 0.70). *S.cerevisiae* had the low GC content (0.38) (Fig. 1)

Figure 1: GC content among genomes

Nucleotide frequencies were calculated using the formula:

The results are shown in Fig. 2 below. *S.coelicolor* and *R.xylanophilus* have very similar nucleotide frequencies profiles. *S.curvatus* has slightly similar profiles to the two.

Figure 2: Nucleotide frequencies among genomes

Dinucleotide frequencies were calculated using the formula:

The results are shown in Fig. 3. *S.coelicolor* and *R.xylanophilus* again have very similar dinucleotide profiles, particularly in CC, CG, GC and GG frequencies. This is not surprising since both organisms have high GC content results. *S.curvatus* have similar but slightly lower CC, CG, GC and GG frequencies as compared to the two organisms above. This is because *S.curvatus* have the third highest GC content after the two.

Figure 3:Dinucleotide frequencies among genomes

The following two equations are used to calculate amino acid and diamino acid frequencies:

Amino acid frequencies are shown in Fig. 4. Diamino acid frequencies are not shown as there are 400 diamino acids. S.coelicolor and R.xylanophilus have very similar amino acid profiles as well, particularly in alanine (A), arginine (R), glycine (G) and proline (P). This is because codons for A, R, G and P start with the dinucleotides GC, CG, GG and CC respectively and both organism have high GC content. S.curvatus have similar but lower frequencies for A, R, G and P since it has lower GC content than the two organisms.

Figure 4:Amino acid frequencies among genomes

From this exercise, we can draw two conclusions. First, we inferred that *S.coelicolor* and *R.xylanophilus* are more closely related to each other than to the rest of the studied organisms. *S.curvatus* is next closest to the two. Second, we can see that GC content influences nucleotide frequencies which influences dinucleotide frequencies which subsequently influences amino acid frequencies.

1. **ORF prediction**

Next, we used our own written script to predict open reading frames (ORFs) in our genomes. For the prediction, we made the following basic assumptions. First, an ORF starts with a start codon (ATG) and ends with a stop codon (TAA, TAG, TGA). Second, if more than one ORF share the same stop codon, the longest ORF is kept. This is because in a random sequence, a stop codon will appear approximately every 20 codons (3 stop codons in 64 possible codons). Hence, the longer a sequence without a stop codon, the more likely that it is not random but an ORF. Third, the minimum length of genes is 200bp [1] in prokaryote and 300bp [2] in eukaryotes (excluding the stop codons). Fourth, the maximum overlap of ORF is 60bp [3]. Fifth, if two ORFs overlap by more than 60bp, we keep the longer ORF. We predicted 5079 ORFs in *Escherichia coli*, 7172 ORFs in *Streptomyces coelicolor*, 724 ORFs in the fourth chromosome of *Saccharomyces cerevisiae*, 2778 ORFs in *Rubrobacter xylanophilus* and 1758 ORFs in *Spiribacter curvatus*. We named our predictor ‘FuMiSh’ based on the first two characters of our first names.

**Evaluation of predictor**

To evaluate the performance of FuMiSh, we first compared it with the state of art predictor, GLIMMER and assumed that the predictions of GLIMMER are real ORFs. We then compared the predicted ORFs of FuMiSh with the proteomes from UniProt.

1) Comparison with GLIMMER

For this comparison, we ran FuMiSh using minimum gene size of 110bp and maximum overlap of 50bp because these were the settings we used for GLIMMER in practical 2. We compared the predictions of FuMiSh against the predictions of GLIMMER at the nucleotide level. For each reading frame, each nucleotide was classified as True Positive (TP), True Negative (TN), False Positive (FP) or False Negative (FN). Nucleotide predicted as ORF nucleotide by both GLIMMER and FuMiSh is a TP; nucleotide predicted as ORF nucleotide by GLIMMER but not FuMiSh is a FN; nucleotide predicted as ORF nucleotide by FuMiSH but not GLIMMER is a FP; and nucleotide not predicted as ORF nucleotide by both GLIMMER and FuMiSh is a FN. The following accuracy metrics were calculated:

The results are summarized in Table 1.

Table 1: The comparison of ORF predictions between GLIMMER and FuMiSh

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Species** | **No. of GLIMMER ORF** | **No. of FuMiSh ORF** | **Ave. GLIMMER length(bp)** | **Ave. FuMiSh length(bp)** | **sensitivity** | **specificity** | **AC** |
| *E.coli* | 5323 | 6329 | 888 | 773 | 0.95 | 0.917 | 0.922 |
| *S.coelicolor* | 8548 | 8175 | 939 | 954 | 0.59 | 0.61 | 0.529 |
| *S.cerevisiae ch4* | 921 | 1604 | 1235 | 782 | 0.97 | 0.88 | 0.91 |
| *R.xylanophilus* | 3375 | 3167 | 879 | 869 | 0.623 | 0.671 | 0.586 |
| *S.curvatus* | 1862 | 1955 | 973 | 897 | 0.841 | 0.869 | 0.829 |

For *S.coelicolor*, *R.xylanophilus* and *S.curvatus*, the number of ORFs predicted by FuMiSh is close to the number of ORFs predicted by GLIMMER. For *E.coli* and *S.cerevisiae*, FuMiSh greatly overpredicted the number of ORFs. The average length of ORFs predicted by FuMiSh is shorter than the average length of ORFs predicted by GLIMMER. To better understand these results, we plotted the size distributions of the ORFs predicted by GLIMMER and FuMiSh for each organism (Fig. 5). From Fig. 5, we saw that for ORFs longer than 500bp, GLIMMER and FuMiSh have similar distribution. However, for ORFs shorter than 500bp, FuMiSh tends to overpredict the number of ORFs, especially for *E.coli* and *S.cerevisiae*. This also explains why FuMiSh has shorter average ORFs length than GLIMMER. The reason why GLIMMER did not overpredict the number of short ORFs even when we applied the same minimum ORF length for both predictors is that GLIMMER employed interpolated Markov Model. It used long ORFs which tend to be correct to train the model. From the long ORFs, the model learnt the promoter and ribosomal binding site signals and used them to filter out the incorrect short ORFs.

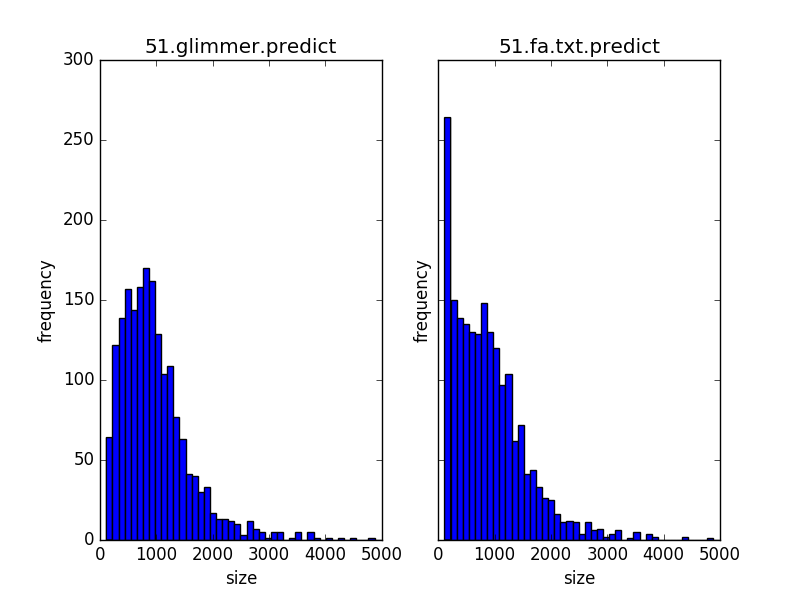
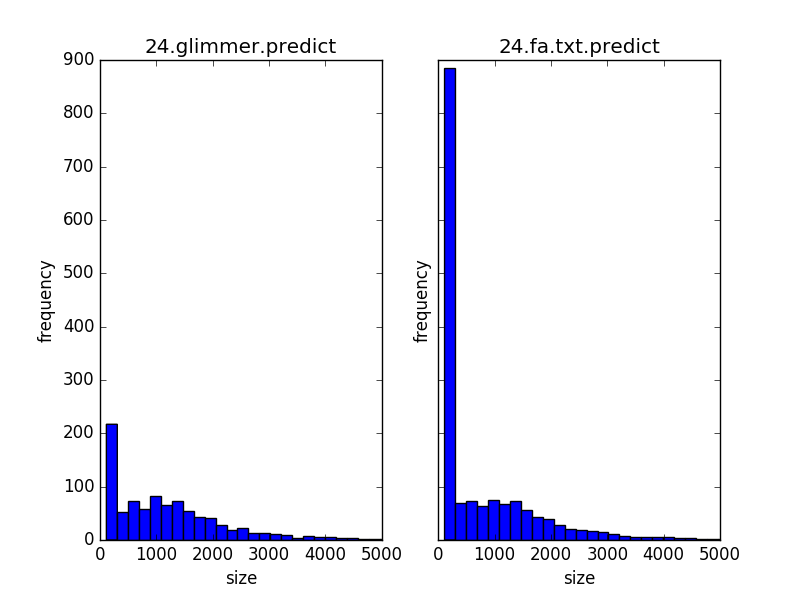
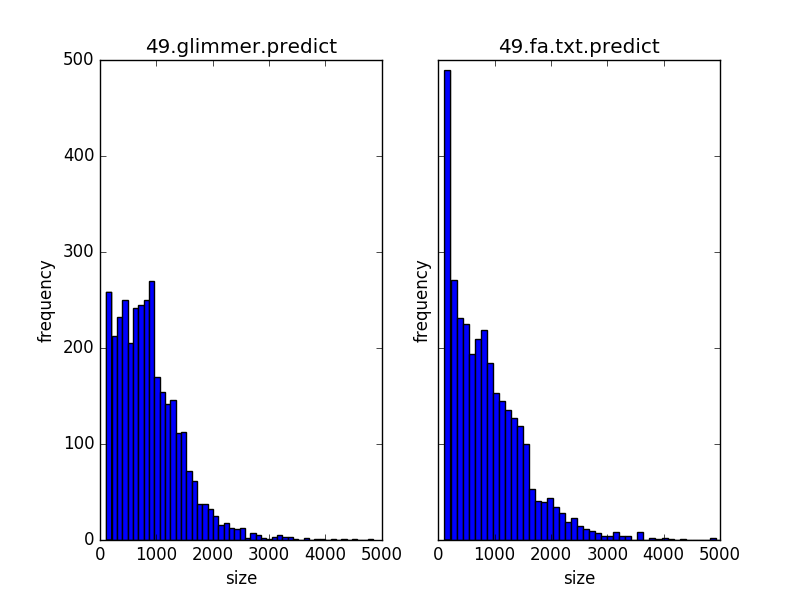
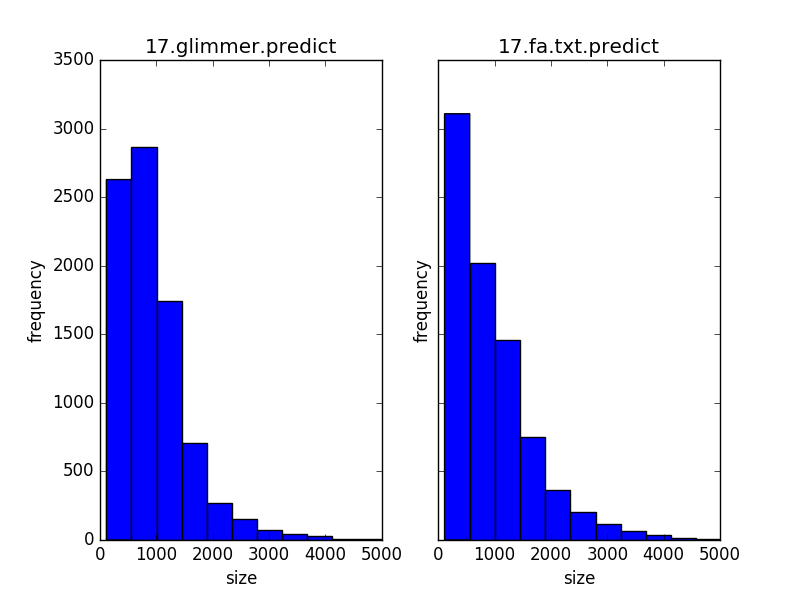
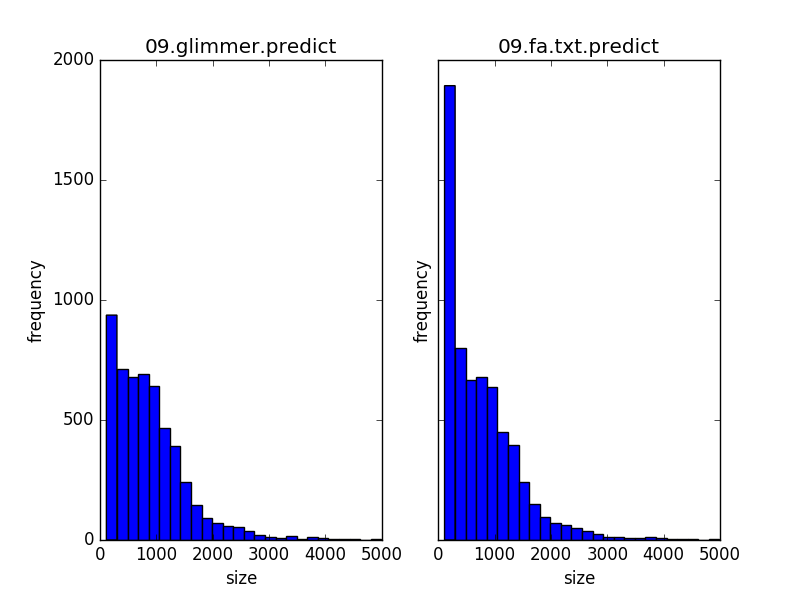


Figure 5: The gene length distribution of GLIMMER(left) and Fumish(right). 09 is *E.coli*, 17 is *S.coelicolor*, 24 is *S.cerevisiae*, 49 is *R.xylanophilus* and 51 is *S.curvatus*.

Next, we examined the prediction accuracy of FuMiSh. From Table 1, FuMiSh performed well for *E.coli*, *S.cerevisiae* and *S.curvatus*, with approximate correlation coefficient (AC) of 0.922, 0.91 and 0.829 respectively. However, FuMiSh performed poorly for *S.coelicolor* (0.529) and *R.xylanophilus* (0.586). This is because *S.coelicolor* and *R.xylanophilus* used alternate start codons, GTG and TTG, which code for valine and Leucine, in addition to ATG, which codes for Methionine (see Table 2). In fact, more than 40 percent of start codons in *S.coelicolor* and *R.xylanophilus* are GTG. This could be due to the fact that the two organisms are GC rich, and thus have a higher tendency to use GTG as a start codon. Because FuMiSh is unable to predict the actual start codons of *S.coelicolor* and *R.xylanophilus*, the AC values are much lower for these two organisms.

Table 2: Alternate start codons usage

|  |  |  |
| --- | --- | --- |
| **Species** | **Fraction of GTG and TTG start codons** | **Fraction of GTG start codons** |
| *E.coli* | 0.199 | 0.140 |
| *S.coelicolor* | 0.454 | 0.408 |
| *S.cerevisiae ch4* | 0.194 | 0.106 |
| *R.xylanophilus* | 0.562 | 0.438 |
| *S.curvatus* | 0.249 | 0.186 |

2) Comparison with UniProt proteomes

To better evaluate the performance of Fumish, we translated our predicted genes and compared them with the proteome in UniProt. To identify the proteins that were correctly predicted, we performed two BLASTp tests. Firstly, we used the UniProt proteome as database and the predicted proteins as queries to blast. Hits with e-value less than 0.001 were selected as the predicted proteins that appear in the UniProt proteome. Then, we used the UniProt proteins as queries and the predicted proteome as the dataset and ran blast again. Proteins that show up in both blast tests are considered as true positive predictions.

*True positive: Proteins that appears in both the real proteome and the predicted proteome.*

*False positive: Proteins that appears only in the predicted proteome instead of the UniProt proteome.*

*False negative: Proteins that appears only in the UniProt proteome instead of the predicted proteome.*

The sensitivity and specificity are calculated using formulas above. The F1 score measures the overall prediction accuracy.

Our predictor achieves 95.2% prediction accuracy in yeast, and the average prediction accuracy is 74.9%. Table 2 is the summary of all results.

Table 3: The comparison of proteins predicted by Fumish and UniProt proteome

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Species** | **UniProt protein number** | **Predicted gene number** | **Reciprocal best hit number** | **Specificity** | **Sensitivity** | **F1 score** |
| Escherichia coli | 4313 | 5079 | 3691 | 0.727 | 0.856 | 0.786 |
| Streptomyces coelicolor | 7731 | 7172 | 4009 | 0.559 | 0.519 | 0.538 |
| Saccharomyces cerevisiae | 769 | 724 | 711 | 0.982 | 0.925 | 0.952 |
| Rubrobacter xylanophilus | 3127 | 2778 | 1945 | 0.700 | 0.622 | 0.659 |
| Spiribacter curvatus | 1861 | 1752 | 1460 | 0.833 | 0.785 | 0.808 |

**Improvements**

1) Adding promoter information

Adding promoter information filters small ORFs and selects the overlapping ORFs more precisely. Pribnow boxes are used as the promoter in prokaryotes. The Pribnow box is located at 10bp upstream the transcriptional start site [4], and the transcription start site is 20 to 40 nucleotides upstream the start codon [5]. We can search for the Pribnow box and select the start codons that are 30bp-50bp downstream the Pribnow box. In eukaryotes, TATA boxes information can be used in the same way as the Pribnow box.

In our *E.coli* genome, there are only around 700 Pribnow boxes(TATAAT) for 5000 genes. The major explanation is that in prokaryotes, one operon is shared by several genes. In such case, the Pribnow box can’t be used directly as a selection criterion in ORF finding. Instead, it can be used to validate prediction and selecting overlapping genes, or this information can be applied to the model training in machine learning based predictions.

2) The minimum gene length

We selected the minimum gene length in *E.coli*(200bp) [1] as the minimum gene length for prokaryotes and the minimum gene length in yeast (300bp) [2] to filter small genes in eukaryotes. The minimum gene length varies in different species and the threshold should be set according to different species.

3) Overlapping genes

Overlaps in different reading frames that are longer than 60bp [3] are forbidden in our predictor. In this case, we only select the longest ORF in the overlapping genes. To improve the prediction accuracy, we can apply different maximum overlapping length in different species. Also, overlapping genes have preference in orientation, convergent overlapping genes are three times more likely than divergent overlapping genes. [6] We should also consider the orientation when selecting overlaps.

4) Alternative start codons

Besides ATG, GTG and TTG are also used as start codons. In the prediction result of Fumish, the prediction accuracy decreases with the increase of GC content, which indicates that alternative start codons are more frequently used in high GC content organism (fig 6). Also, in the prediction result of GLIMMER some genes encode Val as the first amino acid, and Fumish mispredicted the first few amino acids until it meets ATG (Met). Adding alternative start codon information can improve the prediction accuracy.

Figure 6: GC content and prediction accuracy

The higher the GC content is, the more likely the ORFs starts with GTG instead of ATG. We should also include other possible start codons in the predictor.

**Evolutionary relationship**

For the final part, evolutionary relationship analysis, we chose dinucleotide frequencies to calculate the distance between genomes and visualized the evolutionary relationship between species in a phylogenetic tree using neighbor-joining method (Fig 7). The motivation for choosing dinucleotide frequencies for distance calculation is that it has been shown that dinucleotide frequency distributions are quite species specific and are even considered a genomic signature [7]. Secondly, we have chosen this distance calculation method over methods using amino acid/diamino acid frequencies because it is not based on the prediction.

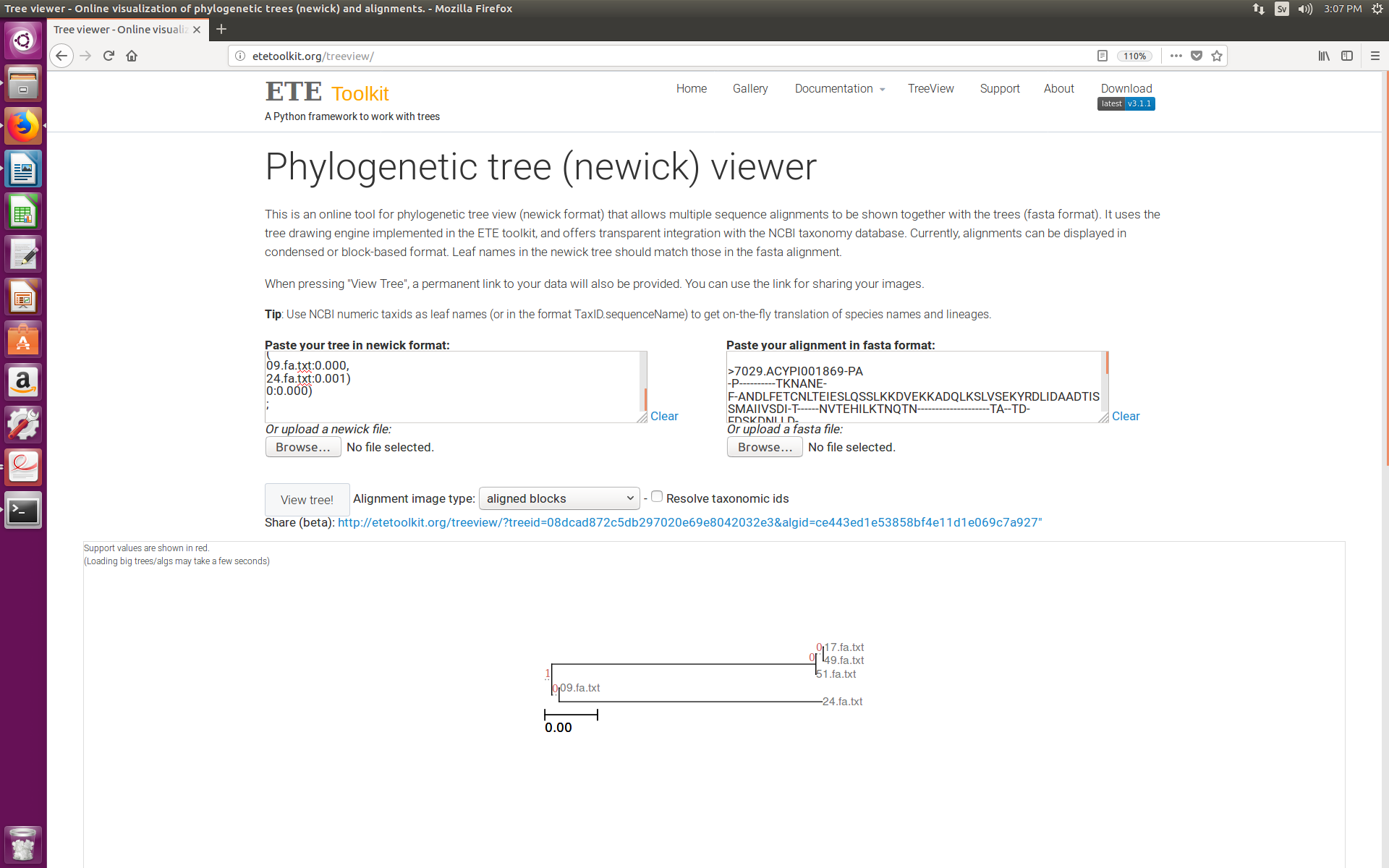


Figure 7 : Evolutionary relationship between species based on dinucleotide frequencies

The tree agrees with the consensus tree built using gene order to calculate distances between species (Fig 8)

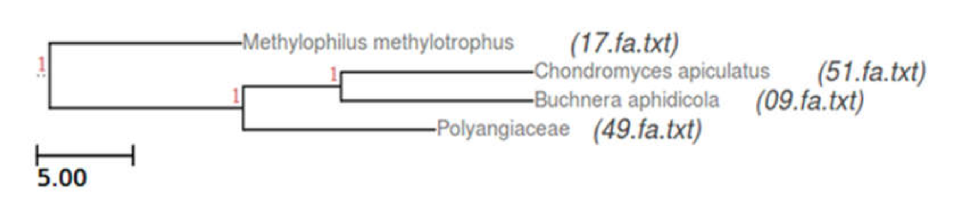


Figure 8: Evolutionary relationship between species based on gene order

**Reference:**

1. Blattner, Frederick R., et al. "The complete genome sequence of Escherichia coli K-12." Science 277.5331 (1997): 1453-1462.

2. Clément-Ziza, Mathieu, et al. "Natural genetic variation impacts expression levels of coding, non-coding, and antisense transcripts in fission yeast." Molecular systems biology 10.11 (2014): 764.

3. Basrai, Munira A., Philip Hieter, and Jef D. Boeke. "Small open reading frames: beautiful needles in the haystack." Genome research 7.8 (1997)

4. Zvelebil, Marketa J., and Jeremy O. Baum. Understanding bioinformatics. Garland Science, 2007.

5. Mendoza-Vargas, Alfredo, et al. "Genome-wide identification of transcription start sites, promoters and transcription factor binding sites in E. coli." *PLoS One* 4.10 (2009): e7526.

6. Hyatt, Doug, et al. "Prodigal: prokaryotic gene recognition and translation initiation site identification." BMC bioinformatics 11.1 (2010): 119.

7. Kariin, Samuel, and Chris Burge. "Dinucleotide relative abundance extremes: a genomic signature." *Trends in genetics* 11.7 (1995): 283-290.

**Python Scripts**

1. **Statistics tool**

Script for calculating GC content, nucleotide frequencies and dinucleotide frequencies: *nucl\_statistics.py*

Usage: python3 nucl\_statistics.py <genome1.fa.txt> <genome2.fa.txt> <genome3.fa.txt> …

Description: The script takes any number of genome fasta files and calculates the GC content, nucleotide frequencies, and dinucleotide frequencies. The results are output to three .csv files ‘gc\_freq.csv’, ‘nucl\_freq.csv’ and ‘dinucl\_freq.csv’ in the working directory.

Script for calculating amino acids frequencies and diamino acids frequencies: *prot\_statistics.py*

Usage: python3 prot\_statistics.py <proteome1.fa.txt.pfa> <proteome2.fa.txt.pfa> <proteome3.fa.txt> …

Description: The script takes any number of proteome fasta files and calculates the amino acid frequencies and diamino acid frequencies. The results are output to three .csv files ‘amino\_freq.csv’ and ‘diamino\_freq.csv’ in the working directory.

1. **ORF finder**

Script for finding open reading frame: *predict\_orf.py*

Usage: python3 predict\_orf.py <genome.fa.txt>

Description: The script takes one genome fasta file and find ORFs that satisfy the conditions stated in the report above. The coordinates of the predicted ORFs are then saved to ‘genome.fa.txt.predict’ in the working directory.

The ‘genome.fa.txt.predict’ file has the same format as GLMMER’s .predict file. To retrieve the nucleotide or protein sequences, the script ‘parseGlimmer.py.2’ can be used.

python2 parseGlimmer.py.2 <genome.fa.txt> <genome.fa.txt.predict>

or

python2 parseGlimmer.py.2 <genome.fa.txt> <genome.fa.txt.predict> --translate

Script for evaluating the nucleotide level accuracy of our predicted ORFs against GLIMMER ORFs: *evaluate.py*

Usage: python3 evaluate.py <genome.fa.txt> <genome.glimmer.predict> <genome.fa.txt.predict>

Description: The script takes the genome fasta file, GLIMMER prediction file and the prediction file generated above and calculates sensitivity, specificity and approximate correlation coefficient based on nucleotide level accuracy in all six reading frames. In addition, it also gives a summary of the number of ORFs and average ORF length of both prediction files. The results are output to the standard output.

3. **Gene distribution Plotter**

Script for plotting the gene length distribution of two predictors: *plotGeneLength.py*

Usage: python3 plotGeneLength.py <predictionResult1> < predictionResult1>

Description: This script takes two gene prediction results and plots the gene distribution of the two predictions.

The input prediction result file should be written in GLIMMER output format.

<column0> <column1> <column2> <others>

Gene name Start codon position Stop codon position …

4. **Cross BLAST hits counter**

Script for count the proteins that appears in both BLAST results: *blast\_count.py*

Usage: python3 blast\_count.py <blastresult1> <blastresult2>

Description: This script takes two blast result files and finds the number of overlapping hits in both files.