**Comparative genomics final report**

*by Shuhan Xu, Fuqi Xu, Milda Valiukonytė*

In this project, we have performed three studies of our five genomes: analysis of genetic composition (GC, nucleotide, dinucleotide, amino acid and diamino acid frequencies), prediction of open reading frames and calculation of distances between genomes to understand their evolutionary relationships. We wrote three Python scripts for this purpose, named GC\_content.py, ORF\_predictor.py and distance.py (see attachments).

The five genomes included 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).

The genetic composition analysis has shown that there is a great variation among the five genomes regarding their GC content, nucleotide, dinucleotide, and diaminoacid frequencies. The GC content was calculated using the formula:

.

The highest GC content was found in *Streptomyces coelicolor*,while *Saccharomyces cerevisiae* had the lowest GC content. (Fig 1.)

**Fig 1.** GC content among genomes

Dinucleotide frequencies were calculated using formula:

The results have shown that bacterial species are rich in CC, CG, GC, GG dinucleotides, which is in line with the GC content results. Overall, we have observed great differences between dinucleotide frequencies among species which can indicate that dinucleotide frequency pattern is species-specific (Fig 2). Similar patterns have been observed in nucleotide and amino acid frequency distributions (Fig 3 and Fig 4).

**Fig 2.** Nucleotide frequencies among genomes

**Fig 3.** dinucleotide frequencies among genomes

**Fig 4.** Amino acid frequencies among genomes

**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 the start codon (TAC) and ends with stop codons (ATT, ACT, ATC). Second, the minimum length of genes in prokaryotes is 200bp [1], 300bp [2] in eukaryotes (excluding the stop codons). Third, in the same reading frame, if there are overlapping genes sharing the same stop codons. the longest ORF is the one we are looking for. Fourth, in different reading frames, the maximum overlapping length is 60bp [3]. Fifth, in different reading frames, if the maximum overlapping length is greater than 60bp, we keep the longer ORFs. We predicted 5079 ORFs *Escherichia coli***,** 7172 genes in *Streptomyces coelicolor*, 724 genes in the fourth chromosome of *Saccharomyces cerevisiae*, 2778 genes in *Rubrobacter xylanophilus* and 1758 genes in *Spiribacter curvatus*.

**Evaluation**

To evaluate the performance of Fumish predictor, we compared it with the state of art predictor, GLIMMER and assumed the prediction of GLIMMER is the real genes. Moreover, we examined the translated ORFs of Fumish with their proteome in UniProt.

1) Comparison with GLIMMER

We compared the prediction accuracy of GLIMMER and Fumish at the nucleotide level and compared the gene size distribution of them. We defined nucleotides predicted both in GLIMMER and our own predictor as true positive (TP), Nucleotides that appears only in our predictor instead of GLIMMER as false positive (FP), nucleotides that presents in GLIMMER instead of our own predictor as true negative (TN) and nucleotides that don’t show up in both predictions as false negative (FN).

The following formulas were used to compare:

The prediction accuracy of *E.coli* is 0.922, and the lowest prediction accuracy is in yeast with AC = 0.529. The average length Fumish predicted is shorter in all prokaryote genomes. Table 1 summarizes the result.

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

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **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 |

We also plotted the gene size distribution of both GLIMMER and Fumish. Our predictor predicts more short genes (gene size <500bp) than GLIMMER. The prediction agrees with GLIMMER in longer, especially genes that longer than 2000bps. The gene length distribution of *E.coli* and *S.coelicolor* of Fumish are very similar with the predictions of GLIMMER, which agrees with the approximate correlation coefficient value.

A screenshot of a cell phone

Description generated with very high confidenceA screenshot of a cell phone

Description generated with high confidenceA screenshot of a cell phone

Description generated with very high confidenceA screenshot of a cell phone

Description generated with high confidenceA screenshot of a cell phone

Description generated with very high confidence

Figure 1: 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*, 51 is *S.curvatus*

2) Comparison with UniProt proteome

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 2: 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 [1], and the transcription start site is 20 to 40 nucleotides upstream the start codon [2]. 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) [3] as the minimum gene length for prokaryotes and the minimum gene length in yeast (300bp) [4] 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 [5] 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.

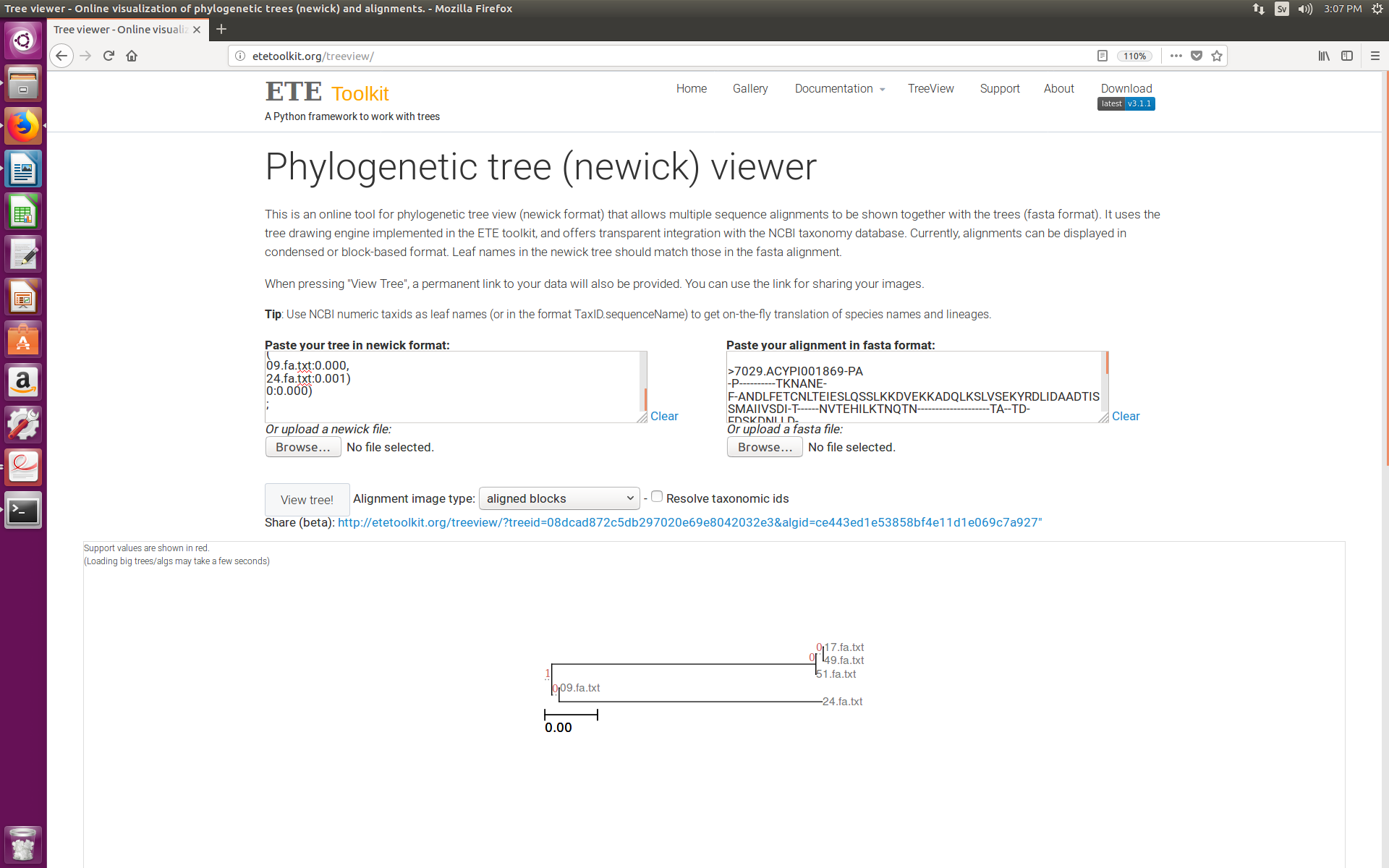
4) Alternative start codons

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 5). 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.

**Fig 5.** GC content and prediction specificity

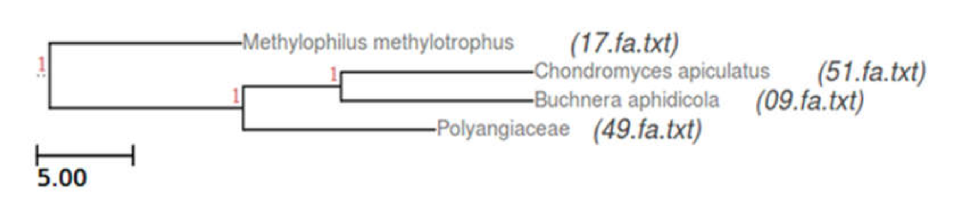
The higher the GC content is, the more likely the ORFs starts with GTG instead of ATG. We should also include other possible start codons in the predictor.

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 6). 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.



**Fig 6.** 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 7.)



**Fig 7.** Evolutionary relationship between species based on gene order

**Reference:**

1. Mendoza-Vargas, Alfredo, et al. "Genome-wide identification of transcription start sites, promoters and transcription factor binding sites in E. coli

2.

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

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

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

5. 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.

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.