Group 7:
Ryno Lawson
Vasileios-Evripidis Kyriakidis

Comparative Genomics
Practical 2 Report : Gene prediction

Summary

The main goal of this exercise is to get us familiar with two important softwares in gene finding, Glimmer and GENSCAN. We used both of them for gene prediction on our query sequences that were described on the previous exercise and attempted to understand how they perform the gene search, what are their advantages and disadvantages and we also tried to explain the results we got by both of the softwares provided.

Glimmer

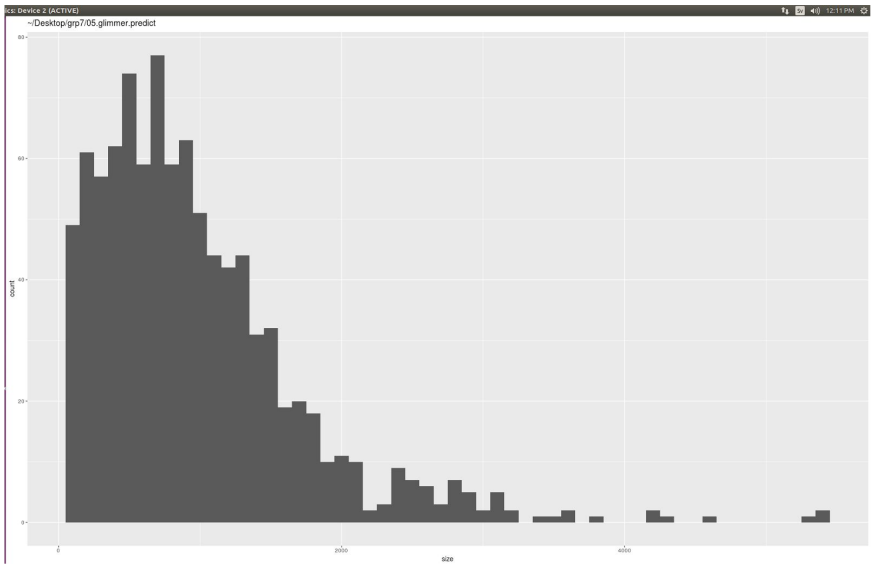
By using Glimmer for this exercise our goal is to create histograms of the gene size distributions for the genomes of the organisms we are assigned with. The first step when running Glimmer is to search for long ORFs in the query genome. The data provided by this search are used for creating training sets and that leads to the last part, where Glimmer predicts existing genes using HMM. For the creation of a training set we extracted the long ORFs from the genomes. In creation of the training set for Glimmer3, it is possible to use any set of genes and/or genes from a organism of close similarity. It is possible to even use low coverage draft genome assembly sequences, which are short sequence fragments.

What became clear after reading tutorials and guides about how GLIMMER works, is that it's not suitable for all genomes. Glimmer is not suitable for eukaryotic genome because it can't identify introns and exons. Glimmer 2.0 though has been proved to be capable of identifying the genes of some parasites with high gene density and few or no introns. In order to find genes in eukaryotes a special version of Glimmer was created under the name GlimmerHMM, which incorporates splice site models adapted from the GeneSplicer program and uses interpolated Markov models for evaluating the coding regions.

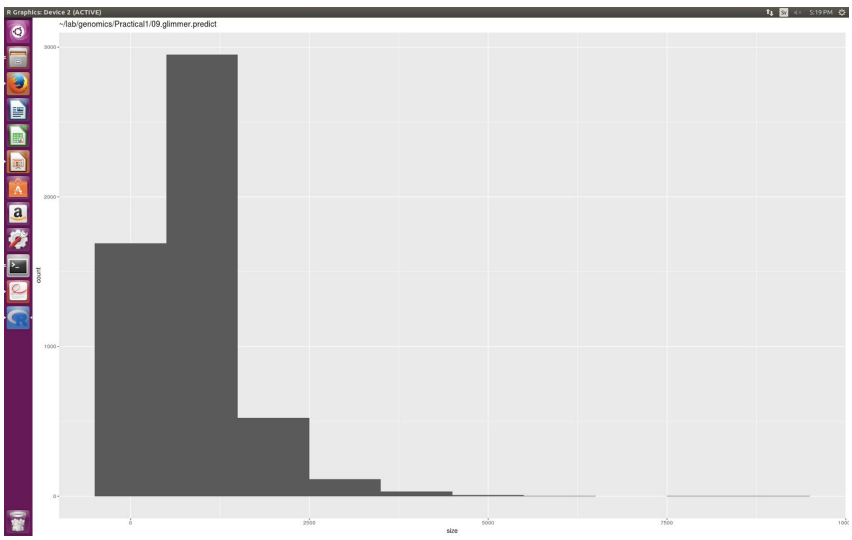
The major advantage of Glimmer3 is that it is self contained. That means that instead of using ORFs as source for the training set, we can also use either another set of genes of our choice or even another closely related organism.

After running Glimmer and getting the files with the predicted genes, we used R to create histograms that display the distribution of the gene sizes of our query genomes.

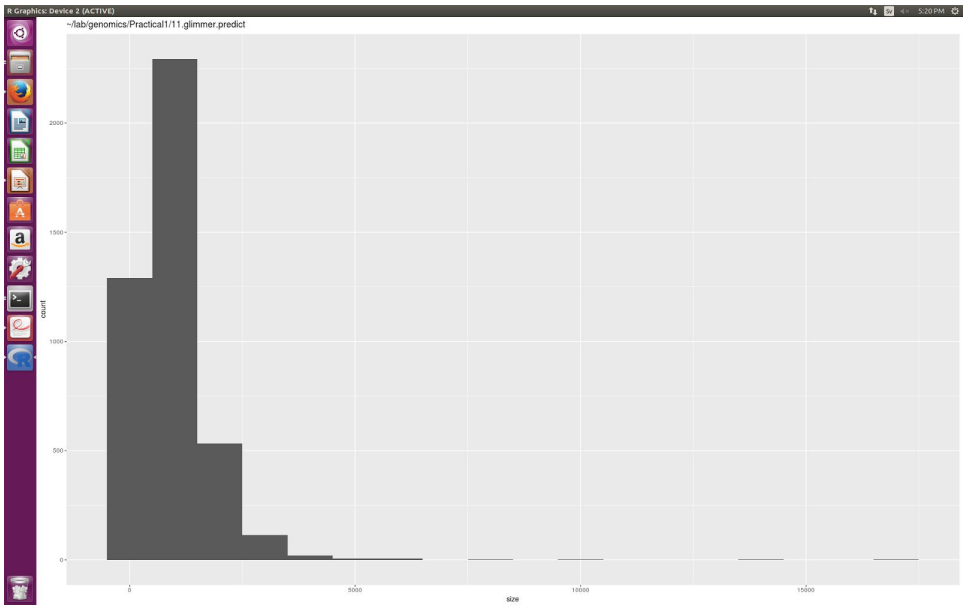
These are our results:



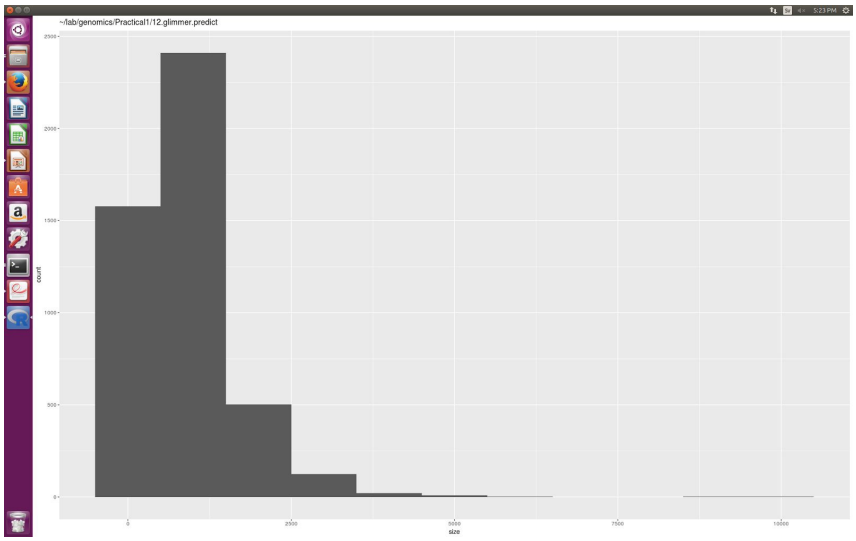
Histogram genome 05. Note: For 05.fa.txt we changed the binwidth from 1000 to 100 and we removed the outlier from the predicted file as it was the size of the whole genome. There is an additional outlier which is questionable but it proved challenging to find the entry.



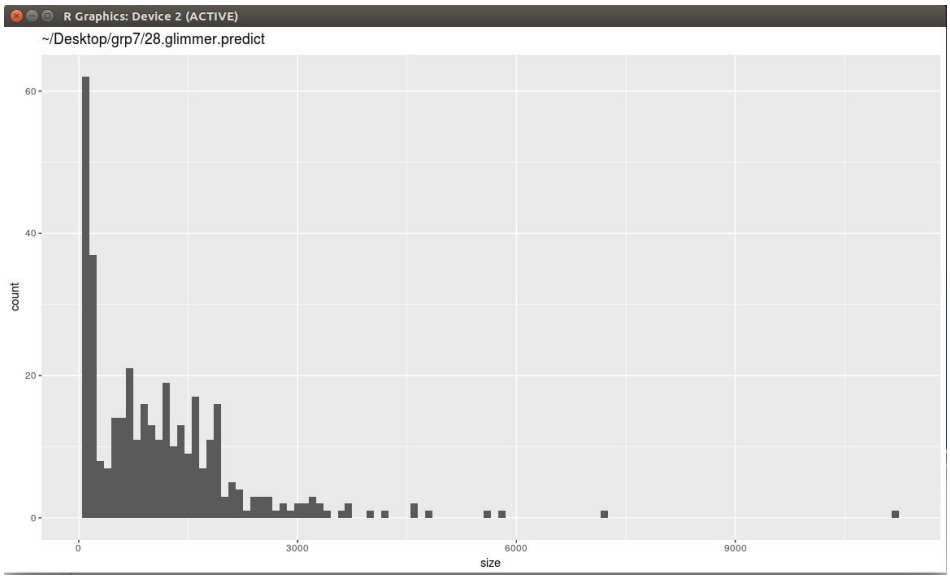
Histogram genome 09. Note: There are outliers which skew the distribution and are not probable genes as they are very large. These should be checked and possibily be excluded to get addition resolution of the distribution



Histogram genome 11. Note: There are outliers which skew the distribution and are not probable genes as they are very large. These should be checked and possibily be excluded to get addition resolution of the distribution



Histogram genome 12. Note: There are outliers which skew the distribution and are not probable genes as they are very large. These should be checked and possibily be excluded to get addition resolution of the distribution



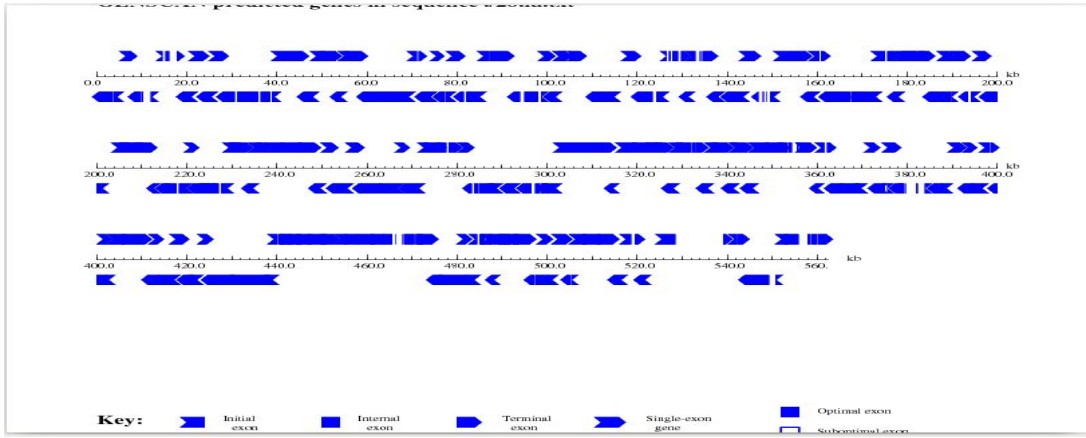
Histogram genome 28. Notes: For 28.fa.txt we changed the binwidth from 1000 to 100 and we removed the outlier from the predicted file as it was the size of the whole genome. There is an additional outlier which is questionable but it proved challenging to find the entry.

Discussion

After a thorough observation on the histograms it seems that not all kinds of genomes have the same gene size distribution. That might be affected by the specificity or the complexity of the organism of interest. While *genome 28* is a single chromosome of a Eukaryotic genome (whereas the others are Prokaryotes who have circular DNA fragment opposed to chromosomes) this is likely to have influence over the distribution of its genes across the chromosome and the genome.

GENSCAN

From GENSCAN we received 2 outputs- 1. the image below and 2. The a fasta file the predicted genes/exons: nucleotide seq and protein seq.



This image shows a visualization of the features of the genes/exons. Sizes in kb, exons positions of start/end, directions, etc.

The other output text file writes out the start, end, promoter, polyA and single-exon gene positions of the exons and extracts the predicted coding nucleotide sequence(s) of the exon and it translate the exon region giving the Predicted peptide sequence(s). It shows 205 predicted gene sequences for the chromosome 8 of *Saccharomyces Cerevisiae.*

GENSCAN is a software used for complete gene structure identification in genomic DNA. It uses the general probabilistic Hidden Markov Model library to predict the location of genes and their exon-intron boundaries in genomic sequences from a variety of organisms.

One of the main disadvantages of GENSCAN is that requires a hefty amount of memory from computational systems. The reason behind this is that GENSCAN makes use of the Viterbi Algorithm which is mentioned in the previous practical. It applies the forward-backward dynamic programming algorithm which requires large memory and computational power. The runtime shows to be linearly proportional to the length of the sequence. For that reason we always have to make sure that the system we are working with has enough memory to run the types of sequences we want to run efficiently. That means that if our system is slow that will also increase the computational speed of the program. One suggested solution for that problem is to break the sequences into smaller pieces, but by doing that we may lose important information from our main sequence leading to lower performance and accuracy.

Task: Extract protein sequences (proteomes) and gene sequences (genomes) from glimmer and genescan for relevant genomes. You can validate your dna->protein translation 2 script by using blast on resulting sequences and inspecting fasta file. Save fasta files with protein and nucleotide sequences.

Validation of the fasta file through the input of the fasta file into the BLASTx web server portal
>>>python2.7 parseGlimmer.py 28.fasta 28.glimmer.predict

parseGlimmer.py code - See comments in blue

```
#!/usr/bin/env python
from Bio.SeqRecord import SeqRecord  #Importing of Biopython libraries regarding
from Bio import SeqIO                                   sequences and seq input/outputs

import argparse
parser = argparse.ArgumentParser()
parser.add_argument('fasta', help='Fasta file with your genome')        #assign a variable to the fasta input file
parser.add_argument('glimmer', help='Glimmer prediction file')          #assign a variable to the glimmar input file
parser.add_argument('--translate', help='Translate to protein',                 #the scripts options
            action='store_true')
args = parser.parse_args()
```

```
# Read the sequence file
seq_record = SeqIO.parse(open(args.fasta), "fasta").next()

outseqrecords = []
# Read the glimmer file, record by record
for inline in file(args.glimmer): #reading the line of the glimmer prediction file
        if '>' in inline:                       #checking the line and reading the label with a condition
                seqname = inline.split()[0][1:]
                outfilename = "%s.nfa" % (seqname)      #creating an output file for nucleotide fasta file
                if args.translate:                      #if the translate option is requested
                outfilename = "%s.pfa" % (seqname)      #creating an output file for protein fasta file
                continue
        if "orf" not in inline:
        continue
        orfname, sbegin, send, rf, score = inline.strip().split()       #parsing of the different column in the glimmer file
        sbegin = int(sbegin)                            #position of begin of gene seq
        send = int(send)                                #position of end of gene seq
        rf = int(rf)                                    #Number/direction of reading frame
        # reverse complement
        if rf < 0:
        sbegin, send = send, sbegin                     #negative reading frame and swops begin and end position
        sbegin -= 1             # Python indexes start a 0
        score = float(score)
        # split the sequence record
        newseq = seq_record.seq[sbegin:send]  #assigning the sequence from fasta with the positions
        if rf < 0:                                      from glimmer file
        newseq = newseq.reverse_complement()            #use of the reverse complement function on newseq
        # translate if requested
        if args.translate:                              #if translate option is selected by the user
        newseq = newseq.translate(stop_symbol='')       #use of the translate function on newseq
        # add a sequence record to the output
        name = seqname+"_"+orfname
        if rf < 0:                                      #labelling as reverse complement
        name = seqname+"_"+orfname+'_rev'
        outseqrecords.append(SeqRecord(newseq,                  #printing to output file
                id=name,
                description=''))

SeqIO.write(outseqrecords, open(outfilename, "w"), "fasta")             #writing to output file
```

References

https://en.wikipedia.org/wiki/GENSCAN

Burge C1, Karlin S. Prediction of complete gene structures in human genomic DNA. J Mol Biol. 1997 Apr 25;268(1):78-94.

A.L. Delcher, K.A. Bratke, E.C. Powers, and S.L. Salzberg. Identifying bacterial genes and endosymbiont DNA with Glimmer, Bioinformatics 23:6 (2007), 673-679.