Appendix 1

Preliminary Investigation: 'prelim_investigation.py'

Contains all functions for the preliminary investigation in addition to a function to make a histogram of information by line in the file, aka the the information in the cDNA sequence 10 nucleotides upstream and downstream of the start codon for each gene. It's important to note that for this investigation and set of files, when gene is referred to this means the cDNA sequences 10 nucleotides downstream and upstream of the start codon for that gene. Pertinent functions include:

> **information_by_line(**file_name**)** which returns a dictionary of the information values of a file for each line (gene) in that file.
>
> > Important note: This function relies on the range in the function **total_weights()**. The user will be prompted to input the length of their aligned sequences.
>
> **information_of_aligned_sequences(**file_name**)** which returns the overall information score for all of the genes in the file.
>
> **make_histogram(**file_name**)** which returns a histogram that shows the amount of genes with one level of information versus the other. This function also allows the user to input a title for the histogram that can correspond to the file title of what they're graphing. Information levels for each gene were rounded to integer values to allow the number of genes with each information level to be more easily quantified.
>
> **information_bar_chart_comparison(**filename1, filename2, filename3**)** takes three arguments. This bar chart compares over all information content of the sets of aligned sequences. These are intended to be three files each containing a set of genes grouped by their level of protein expression. In this project the files used were genes with the top 10% of protein expression in dev_Yeast350, the middle 10% of protein expression, and the bottom 10%
>
> > <u>Files used:</u>

- top10%_nogene.txt
    - cDNA sequences 10 nucleotides upstream and downstream of the start codon of genes with the top 10% of protein expression in devYeast_350. Gene names are excluded as well as the start codon sequence itself..
- middle10%_nogene.txt

    - cDNA sequences 10 nucleotides upstream and downstream of the start codon of genes with the middle 10% of protein expression in devYeast_350. Gene names are excluded as well as the start codon sequence itself..
- bottom10%_nogene.txt
    - cDNA sequences 10 nucleotides upstream and downstream of the start codon of genes with the bottom 10% of protein expression in devYeast_350. Gene names are excluded as well as the start codon sequence itself..

Appendix 2

Investigating G Nucleotide Frequency versus GNN Codon frequency: 'question_e.py'

Contains functions to answer part e in the final project description. It also includes a function to plot a bar chart that compares G and GNN frequencies within a file or "protein expression group" and across files or protein expression groups. It's important to note that for this set of files, 1069 nucleotides downstream of the start codon are included because G and GNN frequencies are being investigated within the yeast ORF. According to sourceX, this is the average ORF length for yeast. Pertinent functions include:

> **gnn_freq(**file_name**)** which returns the percent of overall codons that are GNN codons for a certain file or protein expression group.
>
> **g_freq(**file_name**)** which returns the percentage of nucleotides that are G for a certain file or protein expression group.
>
> **bar_chart_comparison(**filename1, filename2, filename3**)** takes three arguments. It compares G versus GNN frequencies within each file and across files. These are intended to be three files each containing a set of genes grouped by their level of protein expression. In this project the files used were genes with the top 10% of protein expression in dev_Yeast350, the middle 10% of protein expression, and the bottom 10%.
>
> <u>Files used:</u>

- 1069nt_orf_top10.txt
    - cDNA sequences 1069 nucleotides downstream of the start codon of genes with the top 10% of protein expression in devYeast_350. Gene names are excluded as well as the start codon sequence itself.
- 1069nt_orf_mid10.txt

    - cDNA sequences 1069 nucleotides downstream of the start codon of genes with the middle 10% of protein expression in devYeast_350. Gene names are excluded as well as the start codon sequence itself..
- 1069nt_orf_bottom10.txt
    - cDNA sequences 1069 nucleotides downstream of the start codon of genes with the bottom 10% of protein expression in devYeast_350. Gene names are excluded as well as the start codon sequence itself.

Appendix 3

Predicting Consensus Sequences from Weight Information: 'Final Project Code Joyce.py'

Contains collaborative functions from **'prelim_investigation.py'** (Appendix 1) that are called to calculate predicted consensus sequences, corresponding to parts A and C in the final project description. Also contains a formatting function used to format data given from part C of the final project handout. Data called in for this analysis was selected using the formatting.py file (). Pertinent functions include:

predicted_seq(file_name): calls the total_weights(file_name) function from the Preliminary Investigation that returns a nested dictionary of the weights calculated for each position in the selected sequence, from positions -6 to -1 relative to the annotated ATG. This nested dictionary is sorted into a list of tuples in descending weight order, which indexes the zeroth term to determine the consensus sequence, which is printed. This function can also be used for files other than the Cavener file by inputting the length of your aligned sequences.

format_file(file, new_file): formats the 'cavener.txt' file downloaded from the final project description, essentially formats the downloaded file into a new file (newfile), 'readable_cavener.txt' into a format that is readable by the functions necessary to predict the consensus sequence based off of the partial data set.

<u>Files used:</u>
- -6…-1_ top10.txt
    - cDNA sequences from the -6 to -1 position, nucleotides upstream and of the start codon of genes of the bottom 10% of protein expression in devYeast_350. Gene names are excluded as well as the start codon sequence itself..
- -6…-1_mid10.txt

        - cDNA sequences from the -6 to -1 position, nucleotides upstream and of the start codon of genes of the middle 10% of protein expression in devYeast_350. Gene names are excluded as well as the start codon sequence itself..
    - -6…-1_bottom.txt
        - cDNA sequences from the -6 to -1 position, nucleotides upstream and of the start codon of genes of the bottom 10% of protein expression in devYeast_350. Gene names are excluded as well as the start codon sequence itself..
    - readable_cavener.txt
        - cDNA sequence given from final project description, formatted through format_file(file_name) function by indexing positions -6 to -1 relative to the start codon. Contains 50 sequences 10 nucleotides upstream of the ATG start codon, indexed as aforementioned.

Appendix 4

SQL Code: 'Project Data.sql'

To get data for our studies, I used SQL to create 6 tables from devYeast_350. We wanted to study the consensus sequences of yeast genes in the top 10% of protein expression, the middle 10% of protein expression (45%-55%), and the bottom 10% of protein expression (excluding 0%, noted in code since we want to exclude genes that aren't expressed). 3 of them were in depth tables, which were joined from 3 separate data tables(dbo.tableTranscriptsWithOrfs, dbo.tblProteinExpressions, and dbo.tableTranscripts) and consisted of gene names, the length of the ORF, the ORF sequence, the cDNA sequence, and the protein expression. I used these tables to determine the exact index of the ORF sequence and to ensure protein expression was filtered correctly. The other 3 tables only consisted of gene names and cDNA sequence (from dbo.tblProteinExpressions and dbo.tableTranscripts), and were sorted on protein expression percentage of the desired data. This ended up being the data I downloaded and used in our analyses. I could not figure out how to download as a .txt, so downloaded data directly as an Excel spreadsheet, then exported as a .txt, which then had to be formatted – see Formatting SQL Data.

Appendix 5

Formatting SQL Data: 'formatting.py'

Contains a singular function that formats downloaded and exported SQL data. True function was manually changed multiple times to fit specific parameters for each specific study to alter original downloaded data files. Pertinent functions and files:

<u>Functions</u>:

- **info_frame(file, newfile)** takes 2 parameters, file, the original data file (either 'top.txt', 'mid.txt', or 'bottom.txt') and newfile, the name of the .txt file the altered data will be stored in. Creates a new file in the current working directory with specified file name (newfile). This function can be altered (all previous versions have been commented out and annotated with how they filtered data) to sequence data specific to our studies. The second parameter newfile can be changed to a descriptive name of the specific data used for each study. All .txt data used in these studies were created by using and altering this function

<u>Files used</u>:

- top.txt
    - Consists of cDNA sequences in the top 10% of protein expression from devYeast_350, with gene names one line above its corresponding cDNA sequence
- mid.txt
    - Consists of cDNA sequences in the middle 10% of protein expression from devYeast_350, from 45% to 55%, with gene names one line above its corresponding cDNA sequence
- bottom.txt
    - Consists of cDNA sequences in the bottom 10% of protein expression from devYeast_350, with gene names one line above its corresponding cDNA sequence

Appendix 6

Investigating differences in information content of annAUGs with and without upstream ATGs: 'question_d.py'; 'UpstreamATGCount.ipynb'

This notebook consists of several functions that altogether output separate text files of the sequences that do and do not contain ATG occurrences in the 5' UTR. The output of the last function, **info_sequence_prep(source)**, generates files that are usable for functions created in prelim_investigation.py (Appendix 1). The **information_of_aligned_sequences(**file_name**)** function from Appendix 1 was used to calculate the infoscore for each set of aligned sequences. A function was also created to calculate the percent of genes with upstream ATGs in each protein expression set. A new bar chart function was created to visualize the differences in information content of annAUGs with and without upstream ATGs for the top, middle, and bottom 10% of protein expressing genes.

<u>Pertinent Functions to Analysis</u>:

ATG_hit_dict(source): works in concert with the returned dictionary from **ATG_make_dict** and uses ‘135upstream_*10_out.txt’ as source files, iterating through and adding one to the value of the dictionary keys of gene names as hits are recorded for upstream ATGs.

upstream_ATG_percent(sourcename): using the returned dictionary from **ATG_hit_dict(source)** determines the percentage of genes that contain upstream ATG sequences from the original file of all 5’ UTRs and returns the percentage. Takes inputs of (‘135upstream_top10_out.txt’, ‘135upstream_mid10_out.txt’, ‘135upstream_bottom10_out.txt’)

information_of_aligned_sequences(file_name**):** using functions from the prelim_investigation.py file, this function was ultimately used to input the outputted ATG and non_ATG files and return the information content of their aligned sequences

information_bar_chart_comparison2(filename1, filename2, filename3, filename4, filename5, filename6): creates a bar chart using calculated information content of ATG and non ATG sequences of sequences of genes in the top 10%, middle 10% (45%-55%), and bottom 10% of yeast protein expression.

<u>Other Functions:</u>

- **ATG_scan(sourcefile, newfile)**: the sourcefile inputs consist of the gene names and respective sequences of 135 nucleotides before the annAUGs generated by SQL. The three different sourcefiles used were ‘135upstream_*10.txt' (where * = top, mid, or bottom protein expressing genes). The newfile outputs a log of the gene names that contain upstream ATGs and the number of occurrences and positions of noted ATGs.
- **ATG_make_dict(source)**: this function uses the output of **ATG_scan** as its source and initializes a dictionary that contains the names of genes with upstream ATGs as the keys with the value zero.
- **atg_hit_list(sourcefile,newfile)**: this function also uses ‘135upstream_*10_out.txt’ as input files, but this time outputs a new file that uniquely lists the names of the genes with upstream ATGs (meaning that gene names wouldn’t be repeated even if they had more than one ATG). These outputted as '135upstream_*10_hit_genename.txt'.

- **make_dict_all(sourcefile):** this used the original SQL-generated text files, ‘135upstream_*10.txt', as inputs to return a dictionary with all the gene names as keys with their 135 nucleotide sequences as values.
- **make_atg_hit_list(source):** this uses the previously generated '135upstream_*10_hit_genename.txt' as inputs, and is intended to reformat the information not as a string object, but as a list, so that it may be used for the last function.
- **info_sequence_prep(source)**: this is the last function that generates the properly formatted files for information content calculation. It is the most involved function and involves editing the names of the output files embedded in the definition of the function as it needs to output two separate logs. Additionally, a third file name is edited within the function for creating the comprehensive dictionary for each of the top, mid, and bottom groups of protein expressors. The actual input for the function is the '135upstream_*10_hit_genename.txt', and the function determines whether or not the dictionary key is in the input list. It writes desired nucleotide sequences to two different output files, 'atg_info_*10_1.txt' and 'no_atg_info_*10.txt', depending on whether or not the key name is present in the hit_genename.txt file.

<u>Files used</u>:

- ‘135upstream_*10.txt’
  - Consists of 135 nucleotide-long cDNA sequences starting from 135 bases before the annotated start AUG, extracted from the SQL databases of the top, middle, and bottom 10% of protein expressing genes.
- ‘135upstream_*10_out.txt’
  - A log of all the gene names that have occurrences of ATGs located in their 5’ UTRs and the number of occurrences of upstream ATGs per gene. Used as input for **upstream_ATG_percent(sourcename)**.
- ‘135upstream_*10_hit_genename.txt’
  - Uniquely lists (as a string object) the names of all the genes with upstream ATGs.
- 'atg_info_*10.txt'
  - Outputs the cDNA sequences of genes that contain upstream ATGs as individual lines of text. This is used as input for calculating information content.
- 'no_atg_info_*10.txt'
  - Outputs the cDNA sequences of genes that do not contain upstream ATGs as individual lines of text. This is used as input for calculating information content.

References/Sources

(1) Average 5’ UTR length in yeast is 134 nt long
Accessed at: Folding Free Energies of 5′-UTRs Impact Post-Transcriptional Regulation on a Genomic Scale in Yeast | PLOS Computational Biology.

(2) average ORF length is 1069 nt.
Accessed at: Genome-wide analysis of mRNA lengths in Saccharomyces cerevisiae - PMC

(3) Accessed at: TRII: A Probabilistic Scoring of Drosophila melanogaster Translation Initiation Sites - PMC

(4) Information Theory Explanation
Accessed at: 17.1: Motif Representation and Information Content - Biology LibreTexts

(5) Information Theory Class Handout
Accessed at:
https://wesmoodle.wesleyan.edu/pluginfile.php/412351/mod_resource/content/1/6_InformationTheory.pdf

(6) Final Project Description
Accessed at:
https://wesmoodle.wesleyan.edu/pluginfile.php/412563/mod_resource/content/7/projects_2022S.pdf

(7) Nucleotide sequences surrounding start codons
Accessed at: From RNA to Protein - Molecular Biology of the Cell - NCBI Bookshelf

(8) Applications of Studies Using Information Theory
Accessed at: 'Information theory' recruited to help scientists find cancer genes -- ScienceDaily

(9) Gabriel Gutiérrez, Lorenzo Márquez, Antonio Marín, Preference for Guanosine at First Codon Position in Highly Expressed Escherichia Coli Genes. A Relationship with Translational Efficiency , Nucleic Acids Research, Volume 24, Issue 13, 1 July 1996, Pages 2525–2527, https://doi.org/10.1093/nar/24.13.2525

Accessed at:
[Preference for Guanosine at First Codon Position in Highly Expressed Escherichia Coli Genes. A Relationship with Translational Efficiency | Nucleic Acids Research | Oxford Academic]

(10) Susana Vinga. 2013. Information theory applications for biological sequence analysis
Accessed at:
https://academic.oup.com/bib/article/15/3/376/183705

(11) Cavener 1987. COmparison of the consensus sequence flanking translational start sites in Drosophila and vertebrates.
Accessed at:
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC340553/?page=2