Code Documentation
Chromosome 8

Jennifer J. Stiens
Bioinformatics, Birkbeck College
28 April, 2018

Overall Schema:

1) Access database for list of genes and gene identifiers.
   -get genes (pymysql data access script)

2) Create gene objects and compile list of genes and identifiers (xml file) for front-end.
   — gene_module
   — getGeneList.py

3) Front-end uses accession number to call functions/scripts for details page.
   — seq_module
   — codon_usage

4) Scripts access database using accession number to:
   -get sequence
   -get exons

5) Use sequence and exon list to calculate annotated sequence, coding sequence, translation, enzyme sites, and codon usage. (Codon usage for entire chromosome is pre-calculated and stored in flat file).
   — whole_genome_freq

Instructions for front-end:

Front end should import three modules: gene_module, seq_module and codon_usage.

In order to populate list of all genes from Chromosome 8, they will need to run:
   getGeneList

For the gene detail page, they will need to use these functions with the accession number:

annotateSeq	to return sequence with exon boundaries marked
codingSeq	to return coding sequence
translate	to return peptide translation
getEnzymes	to return restriction enzyme information
getCodonusage	to return codon usage statistics

Additionally, they will need file of codon usage for entire chromosome using:
   whole_genome_freq

Module Descriptions:

I. gene_module

Class	Method	
Gene	__init__	Create gene object
	Gene.total	Static method for total number of gene objects created
	Gene.number	Assigns id for each gene object
	Gene.acc	Accession number (primary id)
	Gene.genid	Gene name/identifier
	Gene.product	Product name
	Gene.location	Chromosomal location
	Gene._registry.append(self)	Creates registry of gene objects
	__str__	string method for printing attributes of gene object

II. seq_module

This module contains the functions for returning various sequences and annotations.

Function	
getSequence (acc)	Returns genomic DNA sequence dictionary using accession number (key=bp number, value=base)
annotateSeq(acc)	Returns genomic sequence with exon boundaries indicated with '*exon/exon*' annotations (string)
codingSeq(acc)	Returns coding sequence for particular gene (string)
translate(acc)	Uses coding sequence to translate into peptide sequence (string)
enz_cut(acc, seq, enzyme)	Returns dictionary of enzyme cleavage sites for any of 5 common enzymes that may cut the genomic sequence, as well as optional custom cleavage sequence (optional enzyme parameter). Sequence is optional and used for testing coding sequence.
getEnzyme(acc, enzyme)	Returns list of enzymes which cut the sequence and includes 'good' or 'bad' indication to show whether it is useful ('good' does not cut in coding region, 'bad' enzymes do), number of cleavage sites and position of cleavage sites (enzyme parameter is optional).

III. **codon_usage**

This module calculates codon usage information for indicated gene.

Function	
codonFreq(acc, dna)	Analyses the frequency of every possible codon in coding sequence of specified gene
codonPercent(acc, freq_table)	Uses frequency information to calculate the codon percentage (number of times codon used per 100bp)
codonRatio(acc, freq_table)	Uses frequency information to calculate the relative ratio of codon usage for each amino acid in the peptide product

getCodonusage(acc)	Returns two dictionaries: percentage {codon: freq/100bp} and usage ratio {amino acid: {codon:ratio}} for indicated gene.

Other Programs

getGenelist.py	Uses gene_module to produce a list of all genes and their identifiers (accession number, gene name, protein product, chromosomal location). Stores list as .xml file.
whole_genome_freq.py	Uses gene_module to get registry of gene objectes, seq_module to obtain coding sequence for each gene and codon_usage module to aggregate frequency/percent codon usage. Calculates relative codon usage ratios across the chromosome. Returns 3 dictionaries. Stores this information in a .txt file.