### Methods

#### Automated Ligamer Design

Hand design of ligamers can be a time consuming and error prone process. To automate the process, I taught myself how to script in the PERL programming language, and use the robust BioPerl modules to design ligamers. The following script: (1) retrieves sequences flanking each ligation event, (2) adjusts the length of these sequences in order to normalize each set of ligamers in terms of Tm, (3) Incorporate barcodes and priming sequences, (4) determine and report all possible outcomes of each isoform-dependent FLLP, and (5) presents the data in a format facilitating convenient ligamer synthesis.

##### README.md

|  |
| --- |
| This package of scripts will assembler ligamers from input UCSC regions.  Initially uploaded to GitHub on 02/19/13  First written by Christian Roy while learning Perl and BioPerl  LAST UPDATE: 02/19/13  Majors steps  Create an input csv file with required information  Run this information sequentially through the scripts  Use the results to order oligos from IDT  Required  Perl  BioPerl  Ensembl Perl APIs  String::Random Perl Package  Helpful hints on installing BioPerl and Emsembl Perl APIs  Install BioPerl, use git  cpan App::cpanminus *# First prep cpan*  cpanm DBI *## Install necessary DBI perl module*  mkdir ~/src; cd ~/src  git clone git://github.com/bioperl/bioperl-live.git  cp ~/.bash\_profile ~/.bash\_profile.bak  echo -e 'PERL5LIB=$HOME/src/bioperl-live:$PERL5LIB' >> ~/.bash\_profile  source ~/.bash\_profile  Install ensembl perl apis  mkdir ~/src; cd ~/src  wget ftp://ftp.ensembl.org/pub/ensembl-api.tar.gz && tar xvfz ensembl-api.tar.gz  Add locations to perlfile libs  echo -e 'PERL5LIB=${PERL5LIB}:${HOME}/src/ensembl/modules PERL5LIB=${PERL5LIB}:${HOME}/src/ensembl-compara/modules PERL5LIB=${PERL5LIB}:${HOME}/src/ensembl-variation/modules PERL5LIB=${PERL5LIB}:${HOME}/src/ensembl-functgenomics/modules export PERL5LIB' >> ~/.bash\_profile |

##### 2-ligamer.assembler.pl

|  |
| --- |
| *#! /usr/bin/perl*  *#Pre requisites*  *# These are working on 02/19/13*  **use** lib "/home/royc/perl5/lib/perl5/"; *# BioPerl location*  **use** lib *"/home/royc/lib/ensembl.perl.zpi/ensembl/modules"; #ensembl packages*  *=head1 Ligamer Assembler*  *This script will automatically create ligamers.*  *=head2 Contact information*  *Script made by Christian Roy, Umass Medical School*  *christian.roy@umassmed.edu*  *=cut*  **use** strict; *# To help wtih variable control*  **use** warnings; *# To help me catch mistakes*  **use** Bio::EnsEMBL::Registry; *# To load remote EnsEMBL Registry*  **use** Bio::EnsEMBL::Slice; *# To retreave sequences from EnsEMBL registry*  **use** Bio::DB::Fasta; *# BioPerl tool to retreave sequnce from local FastA file*  **use** Bio::SeqFeature::Primer; *# BioPerl Tool for Tm normalization*  **use** Cwd; *# To retreave current working directory information*  **my** $dir **=** getcwd; *# Assign current working directory to scalar*  **my** $timestamp **=** localtime(); *# Grab the time at script start*  *## Variables*  **my** (  $file\_input, *# Name of specified input file*  $output\_file, *# Name of file to print results too*  $species, *# The species to grab from Ensembl*  $strand, *# The strand to grab for ligamer sequences*  $working\_sequence, *# The slice sequence variable*  $line\_counter, *#Keep track of stepping through input file*  @arguments, *# Keep track of input arguments*  $fa\_reference, *# Fill if using a local FASTA Reference file*  $chr, *# Obvious*  $coordinates, *# Interim variable for splitting UCSC*  $start, *# obvious*  $end, *# Obvious*  $gene, *# target gene name*  $lig\_location, *# Ligamer prime variable*  $target\_prime, *# Broad variable to define ligamer type - see man*  $UCSCcoordinates, *# Obvious*  $pcrsequence, *# fill with appropriate PCR sequence for terminal oligos*  $barcode, *# Fill will barcode for sequence between regions of comp.*  $note\_line, *# Fill with notes for a ligamer query*  $three\_prime\_PCR\_sequence, *# Fill with three prime PCR sequence*  $five\_prime\_PCR\_sequence, *# Fill with five prime PCR sequence*  $lig\_joiner\_code, *# Internal varialbe for assembling ligamers see man*  $set, *# Move set assembly information input to output file*  );  *#Variables with Defaults*  **my** $verbose**=**0; *# Verbose loading of ensembl databases*  **my** $db\_version**=**62; *# Default database version for ensembl database loading*  **my** $temp**=**"58"; *# Defalt temp for Tm normalization*  **my** $salt**=**"0.05"; *# Default salt concetration for Tm calculation in M*  **my** $lig\_conc**=**"0.00000025"; *# Defeult ligamer conc for Tm calc in M*  **my** $man\_print**=**0; *# for printing manual information*  **my** $help\_print**=**0; *# For printing help informatio to HTML file*  **my** $ligamer\_name**=**0; *# Internal variable for sequental numbering of ligamers*  **my** $remote**=**0; *# set to 1 for ensembl database loading*  **my** $control\_length**=**20; *# Default length for control variables in nt*  **my** $plname**=**$0; *# assign $plname scalar to script name (for help printing)*  *#Print Usage information if nothing is entered at commandline*  **if** (@ARGV**==**0) {system "pod2text $0 | less"; die}  *=head2 Usage*  *-hp = Print HTML POD data for scriptname*  *-mp = Print and view Manual POD data for scriptname*  *-i [File] = File Input*  *-o [File] = File output*  *-v [#] = Verbose for Ensembl loading*  *-d [#] = data\_base version for Ensembl loading*  *-t [#] = Temp in degrees celcius*  *-salt [#] = Salt concentration for Tm in mM*  *-lig\_conc [#] = Ligamer concentration for Tm in nM*  *-c [#} = Minimum length for Control ligamers (default=20)*  *=cut*  *## Finish message if run with no arguments*  *#Parse the command line*  **while**(@ARGV**>**0)  {    @arguments **=** @ARGV; *#Store the command line for printing later*  **my** $next\_arg**=**shift(@ARGV);  **if** ($next\_arg **eq** "-hp") { *# Do you want to print HTML POD Data?*      $help\_print**=**1;      }  **if** ($next\_arg **eq** "-mp") { *# Do you want to print a manual?*      $man\_print**=**1;      }  **if** ($next\_arg **eq** "-i") { *# What is the name of the input file?*      $file\_input **=** shift @ARGV;     }  **if** ($next\_arg **eq** "-f") { *#n Name of the fasta file your sequences are in?*      $fa\_reference **=** shift @ARGV ;      }  **if** ($next\_arg **eq** "-r") { *# Do you want to fetch sequences from ensembl?*      $remote **=** 1      }  **if** ($next\_arg **eq** "-o") { *# Name of output file*      $output\_file **=** shift(@ARGV);      }  **if** ($next\_arg **eq** "-v") { *# Do you want to see the ensembl load data?*      $verbose **=** shift @ARGV;      }  **if** ($next\_arg **eq** "-d") { *# What version of ensembl do you want to use?#*      $db\_version **=** shift @ARGV;      }  **if** ($next\_arg **eq** "-t") { *# What temperature in degrees C do you want to norm ?*      $temp **=** shift @ARGV;      }  **if** ($next\_arg **eq** "-salt") { *# Salt concentration for Tm calculations?*      $salt **=** shift @ARGV ;      $salt **=** $salt **/** 1000; *# from micro Molar to Molar*      }  **if** ($next\_arg **eq** "-lig\_conc") { *# Concentration for Tm calculations?*      $lig\_conc**=**shift@ARGV;      $lig\_conc **=** $lig\_conc **/** 1000000000; *# nM to M*      }  **if** ($next\_arg **eq** "-c") { *# What length would you like (min) for control ligs?*      $control\_length **=** shift @ARGV ;      $control\_length **=** $control\_length**-**1      }  } *## Finish Parsing the command line*  *###################### POD HELP SUBROUTINE CALLS################################*  **my** $scriptname**=**$0;  podhelp( $scriptname, $help\_print, $man\_print, $dir);  *##################### POD HTML Subroutine CALLS#################################*  *#################### open the ensembl registry#################################*  **my** $db;  **if** ($fa\_reference) {    $db **=** Bio::DB::Fasta**->new**($fa\_reference);    }  **if** ($remote**==**1) {    $db **=** ensembl\_database($verbose, $db\_version)    }  *################################################################################*  *#open the output file*  open (OUT, '>'**.**$output\_file) **||** die "The output file could not be created.\n";  *## Print the headers*  **print** OUT  *# Start with general assembler information*  ">Source Program\t",$dir,$0,"\n"**.**  ">Date Run \t$timestamp \n"**.**  ">Arguments entered \t", "@arguments"," \n"**.**  ">Input filename \t$file\_input\n"**.**  ">Output filename \t$output\_file\n"**.**  ">Control Seq Length \t$control\_length plus 1\n"**.**  ">Normalization temperature \t$temp\n"**.**  *##### now all on 1 line print the ligmaer-specific information*  ">Gene\t"**.** *#1*  "Ligamer\_Number\t"**.** *#2*  "Species\t"**.** *#3*  "Strand\t"**.** *#4*  "Ligamer Joiner Code\t"**.** *#5*  "Target Prime\t"**.** *#6*  "UCSC coordinates\t"**.** *#7*  "PCR Used\t"**.** *#8*  "Barcode Used\t"**.** *#9*  "Total Query span\t"**.** *#10*  "Five Prime Sequence\t"**.** *#11*  "5 Prime Length\t"**.** *#12*  "Five Prime Tm\t"**.**  "3 Prime Sequence\t"**.**  "3 Prime Length\t"**.**  "3 Prime Tm\t"**.**  *#"Ligamer Identifier\t".*  "Ligamer Sequence\t"**.**  "Ligamer Length\t"**.**  "Notes\t"**.**  "Set\t"**.**  "\n";  *################################################################################*  *## open the input file*  open (INPUT, $file\_input) **||** die "The file $file\_input couldn't be opened.\n";  *###############################################################################*  *#################read and analyze each line of the input file #################*  **while** (**my** $line**=**<INPUT>) { *## starting brace to read through csv*  **if** ($line**=~**/^#/){**next**} *#skips comments*  **if** ($line**=~**/^>/){**print** OUT $line; **next**} *#skips and trans.these lines*  **if** ($line**=~**/^~/){$line**=~**s/~//;chomp $line; $gene**=**$line;} *# find gene identifier*    $gene**=~**s/[\s]+//g;  **if** ($line**=~**/^\@/){chomp $line;$note\_line**=**$line;**next**} *#store notes*    chomp $line;  **if** ($line**=~**/^PCR-Primer-5'-/g) { *#Find the 5' adaptor*      $five\_prime\_PCR\_sequence**=**$line;      $five\_prime\_PCR\_sequence**=~**s/PCR-Primer-5'-//;      $five\_prime\_PCR\_sequence**=~**s/[\s]+//g;  **print** OUT ">5\_pcr\t"**.**$five\_prime\_PCR\_sequence**.**"\n";  **next**      }  **if** ($line**=~**/^PCR-Primer-3'-/) {*## find the 3' adaptor*      $three\_prime\_PCR\_sequence**=**$line;      $three\_prime\_PCR\_sequence**=~**s/PCR-Primer-3'-//;      $three\_prime\_PCR\_sequence**=~**s/[\s]+//g;  **print** OUT ">3\_pcr\t"**.**$three\_prime\_PCR\_sequence**.**"\n";  **next**      }  **if** ($line**=~**/^</) {*#ligamer query lines start with a '<'*  **unless** ($note\_line) {$note\_line**=**" ";}  **my** $lig\_joiner\_code;  **my** $slice\_sequence;      $line\_counter**++**;       (       $gene,       $species,       $strand,       $lig\_location,       $target\_prime,       $UCSCcoordinates,       $barcode,       $set       ) **=** parse\_the\_line($line);      $lig\_location **=** uc $lig\_location;  *# Parse the ligamer query line*      ($chr, $start, $end)**=** parse\_coordinates($UCSCcoordinates);  **my** $target\_seq\_length **=** ($end**-**$start);  *############ Get genomic slice from ensembl registry ###############*  **if** ($remote**==**1) {        $slice\_sequence **=**          get\_genomic\_sequence          (          $chr,          $start,          $end,          $species,          $db,          )        }  *#####################################################################*  *############ Get the genomic slice from local Fasta #################*  **if** ($fa\_reference) {  *#$chr="chr".$chr;*  **my** $obj **=** $db **->** get\_Seq\_by\_id($chr);        $slice\_sequence **=** $obj **->** subseq ($start **=>** $end);        }  *####################################################################*  *#### get the correct orientation*  **my** ($working\_sequence) **=**        revcom\_slice\_based\_on\_strand          (           $strand,           $slice\_sequence           );  *# Get the T5 end*  **my** ($T5\_seq, $T5\_tm, $T5\_seq\_length)**=**        obtain\_T5\_tm\_sequence  (  $working\_sequence,  $temp,  $lig\_location,  $control\_length,  $salt,$lig\_conc  );  *# Get the T3 end*  **my** ($T3\_seq, $T3\_tm, $T3\_seq\_length) **=**        obtain\_T3\_tm\_sequence  (  $working\_sequence,  $temp,  $lig\_location,  $control\_length,  $salt,  $lig\_conc  );  *#start to build your working HASH*  **my** %common **=**        (        working\_sequence **=>** $working\_sequence,        temp **=>** $temp,        UCSCcoordinates **=>** $UCSCcoordinates,        UCSC\_chr **=>** $chr,        UCSC\_start **=>** $start,        UCSC\_end **=>** $end,        gene **=>** $gene,        ligamer\_name **=>** $ligamer\_name,        species **=>** $species,        strand **=>** $strand,        target\_prime **=>** $target\_prime,        five\_prime\_PCR\_sequence **=>** $five\_prime\_PCR\_sequence,        three\_prime\_PCR\_sequence **=>** $three\_prime\_PCR\_sequence,        barcode **=>** $barcode,        target\_seq\_length **=>** $target\_seq\_length,        seed **=>** $control\_length,        T3\_seq **=>** $T3\_seq,        T3\_tm **=>** $T3\_tm,        T3\_seq\_length **=>** $T3\_seq\_length,        T5\_seq **=>** $T5\_seq,        T5\_tm **=>** $T5\_tm,        T5\_seq\_length **=>** $T5\_seq\_length,        notes **=>** $note\_line,        set **=>**$set,        );  **if** ($lig\_location **eq** "T" **&&** $target\_prime **eq** "5") { *#Terminal 5' targeted*  *#Advance the ligamer number*        $ligamer\_name**++**;        $common{ligamer\_name} **=** $ligamer\_name;  *# Add to the hash table*        $lig\_joiner\_code **=** "T-5";        $common {lig\_joiner\_code} **=** $lig\_joiner\_code;  **my** %lig\_results **=** Terminal\_5(%common);  **my** %final **=** ligamer\_piece\_joiner(%lig\_results);  **my** %bed\_output **=** %final;        output (%final);        };  **if** ($lig\_location **eq** "TC" **&&** $target\_prime **eq** "5") {*# Grab the internal*        $ligamer\_name**++**;        $common{ligamer\_name} **=** $ligamer\_name;        $lig\_joiner\_code **=** "T-C-5-I";        $common {lig\_joiner\_code} **=** $lig\_joiner\_code;  **my** %lig\_results **=** Terminal\_5(%common);  **my** %final\_internal **=**           ligamer\_piece\_joiner               (%lig\_results);  **my** $working\_sequence **=** $lig\_results{working\_sequence};  **my** $T5\_seq\_length **=** $lig\_results{T5\_seq\_length};  *# Now grab the sequence inside of the control*        $working\_sequence **=** $common{working\_sequence};  **my** $T5\_ctrl\_length **=** $common{T5\_seq\_length};        $common{T5\_ctrl\_length} **=** $T5\_ctrl\_length;        $working\_sequence **=** substr ($working\_sequence,$T5\_ctrl\_length);        $lig\_location **=** "IC";        ($T5\_seq, $T5\_tm, $T5\_seq\_length) **=**        obtain\_T5\_tm\_sequence  (  $working\_sequence,  $temp,  $lig\_location,  $salt,  $lig\_conc  );        $common{working\_sequence} **=** $working\_sequence;        $common{T5\_seq} **=** $T5\_seq;        $common{T5\_tm} **=** $T5\_tm;        $common{T5\_seq\_length} **=** $T5\_seq\_length;        $ligamer\_name**++**;        $common{ligamer\_name}**=** $ligamer\_name;        $lig\_joiner\_code**=**"T-C-5-T";        $common {lig\_joiner\_code}**=** $lig\_joiner\_code;        %lig\_results **=** Terminal\_5 (%common);  **my** %final **=** ligamer\_piece\_joiner(%lig\_results);        output (%final\_internal);        output (%final);        }  **if** ($lig\_location **eq** "T" **&&** $target\_prime **eq** "3") {        $ligamer\_name**++**;        $common{ligamer\_name} **=** $ligamer\_name;        $lig\_joiner\_code **=** "T-3";        $common {lig\_joiner\_code} **=** $lig\_joiner\_code;  **my** %lig\_results **=** Terminal\_3 (%common);  **my** %final **=** ligamer\_piece\_joiner (%lig\_results);  **my** %bed\_output **=** %final;        output (%final);        };  **if** ($lig\_location **eq** "TC" **&&** $target\_prime **eq** "3") {  *#Grab the control*        $ligamer\_name**++**;        $common{ligamer\_name} **=** $ligamer\_name;        $lig\_joiner\_code **=** "T-C-3-I";        $common {lig\_joiner\_code} **=** $lig\_joiner\_code;  **my** %lig\_results **=** Terminal\_3 (%common);  **my** %final\_internal **=** ligamer\_piece\_joiner (%lig\_results);  *# Grab the sequence internal of the control*        $working\_sequence **=** $common{working\_sequence};  **my** $T3\_ctrl\_length **=** $common{T3\_seq\_length};        $common{T3\_ctrl\_length} **=** $T3\_ctrl\_length;        $T3\_seq\_length **=** $common{T3\_seq\_length};        $working\_sequence **=** substr ($working\_sequence,0, $T3\_ctrl\_length);        $lig\_location **=** "IC";        ($T3\_seq, $T3\_tm, $T3\_seq\_length) **=**        obtain\_T3\_tm\_sequence          (          $working\_sequence,          $temp,          $lig\_location           );        $common{working\_sequence} **=** $working\_sequence;        $common{T3\_seq} **=** $T3\_seq;        $common{T3\_tm} **=** $T3\_tm;        $common{T3\_seq\_length} **=** $T3\_seq\_length;        $common{bed\_start} **=** $start;        $common{bed\_end} **=** $end;        $ligamer\_name**++**;        $common{ligamer\_name} **=** $ligamer\_name;        $lig\_joiner\_code **=** "T-C-3-T";        $common {lig\_joiner\_code} **=** $lig\_joiner\_code;        %lig\_results **=** Terminal\_3 (%common);  **my** %final **=** ligamer\_piece\_joiner (%lig\_results);        output (%final);        output (%final\_internal);        };  **if** ($lig\_location **eq** "I" **&&** $target\_seq\_length**>**60) {        $ligamer\_name**++**;        $common{ligamer\_name} **=** $ligamer\_name;  **if** ($lig\_location **eq** "I" **&&** $target\_prime **eq** "C") {          $lig\_joiner\_code **=** "I-L-C";          $common {lig\_joiner\_code} **=** $lig\_joiner\_code;  **my** %lig\_results **=** (%common);  **my** %final **=** ligamer\_piece\_joiner(%lig\_results);          $final{pcrsequence} **=** "";  **my** %bed\_output **=** %final;          output (%final);          }  **if** ($lig\_location **eq** "I" **&&** $target\_prime **eq** "N") {          $lig\_joiner\_code **=** "I-L";          $common {lig\_joiner\_code} **=** $lig\_joiner\_code;  **my** %lig\_results **=** (%common);  **my** %final **=** ligamer\_piece\_joiner(%lig\_results);          $final{pcrsequence} **=** "";  **my** %bed\_output **=** %final;          output (%final);  *#my %bed\_final = prep\_bed (%bed\_output);*          }        }  **if** ($lig\_location **eq** "I" **&&** $target\_seq\_length**<=**60) {        $ligamer\_name**++**;        $common{ligamer\_name} **=** $ligamer\_name;        $lig\_joiner\_code **=** "I-S";        $common {lig\_joiner\_code} **=** $lig\_joiner\_code;  **my** %lig\_results **=** obtain\_short\_interal\_tm (%common);  **my** %final **=** ligamer\_piece\_joiner (%lig\_results);        $final{pcrsequence} **=** "";  **my** %bed\_output **=** %final;        output (%final);  *#my %bed\_final = prep\_bed (%bed\_output);*        };      }*## matching brace for ligamer data lines*  **else** {**next**};  }*## Matching brace for csv file input test*  *####### END LIGAMERS ASSEMBLY PORTION*  close OUT;  close INPUT;  **print** "Program Finished.\n";  exit;  *####### END MAJOR WORK OF PROGRAM !!*  *###############################################################################*  *#### Begin Subroutine section of program.*  **sub Terminal\_5** {  **my** %results **=** (@\_);  **my** $T3\_seq **=** "";  **my** $T3\_tm **=** "";  **my** $T3\_seq\_length **=** "";  $results {T3\_seq} **=** $T3\_seq;  $results {T3\_tm} **=** $T3\_tm;  $results {T3\_seq\_length} **=** $T3\_seq\_length;  **return** %results;  }  *################################################################################*  *##*  *################################################################################*  **sub Terminal\_3** {  **my** %results **=** (@\_);  **my** $T5\_seq**=**"";  **my** $T5\_tm**=**"";  **my** $T5\_seq\_length**=**"";  $results {T5\_seq} **=** $T5\_seq;  $results {T5\_tm} **=** $T5\_tm;  $results {T5\_seq\_length} **=** $T5\_seq\_length;  **return** %results;  }  *################################################################################*  *################ BEGIN Subroutine for short slices #############################*  **sub obtain\_short\_interal\_tm** {  **my** %results **=** (@\_);  **my** $working\_sequence **=** $results{working\_sequence};  **my** $working\_sequence\_tm\_obj**=**  Bio::SeqFeature::Primer **->** **new**(**-**seq**=>**$working\_sequence);  **my** $T5\_tm **=** $working\_sequence\_tm\_obj**->**  Tm  (  **-**salt **=>** $salt,  **-**oligo **=>** $lig\_conc  );  $T5\_tm**=**substr($T5\_tm,0,5);  **my** $T5\_seq **=** $working\_sequence;  $results{working\_sequence} **=** $working\_sequence;  $results{T5\_seq} **=** $T5\_seq;  $results{T5\_tm} **=** $T5\_tm;  **return** (%results);  }  *################################################################################*  *################################################################################*  **sub output** { **my** %results**=**(@\_);  **print** OUT $results{gene },"\t"; *# 0*  **print** OUT $results{ligamer\_name },"\t"; *# 1*  **print** OUT $results{species },"\t"; *# 2*  **print** OUT $results{strand },"\t"; *# 3*  **print** OUT $results{lig\_joiner\_code },"\t"; *# 4*  **print** OUT $results{target\_prime },"\t"; *# 5*  **print** OUT $results{UCSCcoordinates },"\t"; *# 6*  **print** OUT $results{pcrsequence },"\t"; *# 7*  **print** OUT $results{barcode },"\t"; *# 8*  **print** OUT $results{target\_seq\_length },"\t"; *# 9*  **print** OUT $results{T5\_seq },"\t"; *# 10*  **print** OUT $results{T5\_seq\_length },"\t"; *# 11*  **print** OUT $results{T5\_tm },"\t"; *# 12*  **print** OUT $results{T3\_seq },"\t"; *# 13*  **print** OUT $results{T3\_seq\_length },"\t"; *# 14*  **print** OUT $results{T3\_tm },"\t"; *# 15*  **print** OUT $results{ligamer },"\t"; *# 16*  **print** OUT $results{warning }; *#*  **print** OUT $results{ligamer\_length },"\t"; *# 17*  **print** OUT $results{notes },"\t"; *# 18*  **print** OUT $results{set },"\t"; *# 19*  *#Commented on 022013*  *#if ( defined $results{T5\_ctrl\_length} ) {*  *# print OUT $results{ T5\_ctrl\_length },"\t";*  *# }*  *#if ( defined $results{T3\_ctrl\_length} ) {*  *# print OUT $results{ T3\_ctrl\_length },"\t";*  *# }*  **print** OUT "\n";  }  *################################################################################*  *######BEGIN Subroutine to parse csv file into variables ########################*  **sub parse\_the\_line** {  **my** $line **=** shift(@\_);  **my** (  $gene,  $ligamer\_name,  $species,  $strand,  $lig\_location,  $target\_prime,  $UCSCcoordinates,  $barcode,  $set  )  **=** split /\t/ , $line ;  $gene**=~**s/^<//;  **print** "Gene - $gene\n";  **print** "Ligmamer name - $ligamer\_name\n";  **print** "species - $species\n";  **print** "strand - $strand\n";  **print** "lig\_location - $lig\_location\n";  **print** "target\_prime - $target\_prime\n";  **print** "UCSC - $UCSCcoordinates\n";  **print** "barcode - [$barcode]\n";  **print** "Set - [$set]\n";  **if** ($barcode**=~**/ /){$barcode**=~**s/ //} *## GO HERE!*  $gene**=~**s/<//;  **return**    (    $gene,    $species,    $strand,    $lig\_location,    $target\_prime,    $UCSCcoordinates,    $barcode,    $set    );  }  *######END Subroutine to parse csv file into variables ############*  *######BEGIN Subroutine to parse genomic coordnates into variables ############*  **sub parse\_coordinates** {  **my** $input**=**shift(@\_);  **my** ($chr,$coordinates) **=**split /\:/,$input;  **my** ($start,$end) **=**split /\-/,$coordinates;  *#$chr=~s/chr//; # I have comment out this to behave with local fasta files!*  $start**=~**s/\,//g;  $end**=~**s/\,//g;  **return** ($chr, $start, $end);  }  *######END Subroutine to parse genomic coordnates into variables ################*  *######Subroutine to make revcom depending on strand annoation###################*  **sub revcom\_slice\_based\_on\_strand** {  **my** ($strand, $slice\_sequence) **=** @\_;  *#if the strand is positive - make the reverse compliment*  $strand**=**lc($strand);  **if** ($strand **eq** 'plus') {    $working\_sequence **=** reverse($slice\_sequence);    $working\_sequence **=~** tr/ACGTacgt/TGCAtgca**/**;    }  *# if the strand is minus - do nothing*  **if** ($strand **eq** 'minus') {    $working\_sequence **=** $slice\_sequence;    }  **return** $working\_sequence  }  *###### END SUBROUTINE revcom\_slice\_based\_on\_strand ####################*  *################ BEGIN Subroutine to obtained only 5' end of working sequence##*  **sub obtain\_T5\_tm\_sequence** {  **my** $working\_sequence **=** shift (@\_);  **my** $temp **=** shift (@\_);  **my** $lig\_location **=** shift (@\_);  **my** $control\_length**=**shift (@\_);  **my** $salt**=**shift (@\_);  **my** $lig\_conc**=**shift (@\_);  **my** $working\_sequence\_length **=** length ($working\_sequence);  **my** $T5\_seq\_length;  **if** ($lig\_location **eq** "TC") {$T5\_seq\_length**=**$control\_length};  **if** ($lig\_location **eq** "T") {$T5\_seq\_length**=**19};  **if** ($lig\_location **eq** "I") {$T5\_seq\_length**=**19};  **my** $T5\_tm**=**0;  **my** $T5\_seq;  **my** $T5\_seq\_out;  **while**($T5\_tm **<** $temp)    {    $T5\_seq\_length**++**;  **print** ".";    $T5\_seq**=**substr $working\_sequence,0, $T5\_seq\_length;  **my** $T5\_seq\_primer**=**    Bio::SeqFeature::Primer **->**  **new**      (  **-**seq**=>**$T5\_seq      );  $T5\_tm **=** $T5\_seq\_primer **->**      Tm      (  **-**salt**=>**$salt,  **-**oligo**=>**$lig\_conc      );    $T5\_tm**=**substr($T5\_tm,0,5);  **if** ($T5\_seq\_length **eq** $working\_sequence\_length) {**last**;}  **if** ($T5\_tm**>=**$temp)      {      $T5\_seq\_out **=** $T5\_seq;  **print** "\n";  **last**      }  **if** ($T5\_seq\_length **eq** 33)      {      $T5\_seq\_out**=**$T5\_seq;  **print** "\n";  **print** STDERR "Warning: "**.**      "Assembly at line $. T5 side cut"**.**      " off due to low Tm \n";  **last**      }  **elsif** ($T5\_tm**<=**$temp){**next**}    }  **return** ($T5\_seq\_out, $T5\_tm, $T5\_seq\_length);  }  *################ BEGIN Subroutine to obtained only 3' end of working sequence##*  **sub obtain\_T3\_tm\_sequence** {  **my** $working\_sequence **=** shift (@\_);  **my** $temp **=** shift (@\_);  **my** $lig\_location **=** shift (@\_);  **my** $control\_length**=**shift (@\_);  **my** $salt**=**shift (@\_);  **my** $lig\_conc**=**shift (@\_);  **my** $working\_sequence\_length **=** length ($working\_sequence);  **my** $T3\_seq\_length;  **if** ($lig\_location **eq** "TC") {$T3\_seq\_length**=**(**-**$control\_length)};  **if** ($lig\_location **eq** "T") {$T3\_seq\_length**=**(**-**19)};  **if** ($lig\_location **eq** "I") {$T3\_seq\_length**=**(**-**19)};  **my** $T3\_tm**=**0;  **my** $T3\_seq;  **my** $T3\_seq\_out;  **while** ($T3\_tm **<** $temp )    {  **print** ".";    $T3\_seq\_length**--**;    $T3\_seq**=**    substr $working\_sequence, $T3\_seq\_length;  **my** $T3\_seq\_primer**=**    Bio::SeqFeature::Primer **->**  **new**      (  **-**seq**=>**$T3\_seq      );    $T3\_tm **=** $T3\_seq\_primer **->**      Tm      (  **-**salt**=>**$salt,  **-**oligo**=>**$lig\_conc       );    $T3\_tm**=**substr($T3\_tm,0,5);  **if** ($T3\_seq\_length **eq** (**-**$working\_sequence\_length)) {**last**;}  **if** ($T3\_seq\_length**<**(**-**80)){die}  **if** ($T3\_tm**>=**$temp)      {      $T3\_seq\_out**=**$T3\_seq;  **print** "\n";  **last**      }  **if** ($T3\_seq\_length **eq** (**-**33))      {      $T3\_seq\_out**=**$T3\_seq;  **print** "\n";  **print** STDERR "Warning: "**.**      "Assembly at line $. T3 side cut"**.**      " off due to low Tm \n";  **last**      }  **if** ($T3\_tm**<**$temp){**next**}    }  **return** ($T3\_seq\_out, $T3\_tm, $T3\_seq\_length);  }  *################ END Subroutine to obtained only 3' end of working sequence####*  *################ BEGIN Subroutine to joined pieces of ligamer #########*  **sub ligamer\_piece\_joiner**{  **my** %results **=** @\_;  **my** $lig\_joiner\_code **=** $results{lig\_joiner\_code};  **my** $T5\_seq **=** $results{T5\_seq};  **my** $barcode **=** $results{barcode};  **my** $T3\_seq **=** $results{T3\_seq};  **my** $pcrsequence;  **my** $short\_sequence**=**$T5\_seq;  **my** $ligamer;  **my** $ligamer\_length;  **my** $warning**=**" ";  **my** $Phos\_mod\_code**=**"\/5Phos\/";  **if** ($lig\_joiner\_code **eq** "T-5")    {    $pcrsequence**=**$results{three\_prime\_PCR\_sequence};    $ligamer **=** join ("",$Phos\_mod\_code, $T5\_seq, $barcode,$pcrsequence);    $ligamer\_length **=** length $ligamer;    $ligamer\_length **=** $ligamer\_length**-**7;    }  **if** ($lig\_joiner\_code **eq** "T-C-5-I")    {    $pcrsequence**=**$results{three\_prime\_PCR\_sequence};    $ligamer **=** join ("",$Phos\_mod\_code,$short\_sequence);    $ligamer\_length **=** length $ligamer;    $ligamer\_length **=** $ligamer\_length**-**7;    }  **if** ($lig\_joiner\_code **eq** "T-C-5-T")    {    $pcrsequence**=**$results{three\_prime\_PCR\_sequence};    $ligamer **=** join ("",$Phos\_mod\_code, $T5\_seq, $barcode, $pcrsequence);    $ligamer\_length **=** length $ligamer;    $ligamer\_length **=** $ligamer\_length**-**7;    }  **if** ($lig\_joiner\_code **eq** "T-3")    {    $pcrsequence**=**$results{five\_prime\_PCR\_sequence};    $ligamer **=** join ("",$pcrsequence,$barcode,$T3\_seq);    $ligamer\_length **=** length $ligamer;    }  **if** ($lig\_joiner\_code **eq** "T-C-3-I")    {    $pcrsequence**=**$results{five\_prime\_PCR\_sequence};    $ligamer **=** join ("",$Phos\_mod\_code,$T3\_seq);    $ligamer\_length **=** length $ligamer;    $ligamer\_length **=** $ligamer\_length**-**7;    }  **if** ($lig\_joiner\_code **eq** "T-C-3-T")    {    $pcrsequence**=**$results{five\_prime\_PCR\_sequence};    $ligamer **=** join ("",$pcrsequence,$barcode,$T3\_seq);    $ligamer\_length **=** length $ligamer;    }  **if** ($lig\_joiner\_code **eq** "I-S")    {    $ligamer **=** join ("",$Phos\_mod\_code,$short\_sequence);    $ligamer\_length **=** length $ligamer;    $ligamer\_length **=** $ligamer\_length**-**7;    }  **if** ($lig\_joiner\_code **eq** "I-L")    {    $ligamer **=** join ("",$Phos\_mod\_code,$T5\_seq,$barcode,$T3\_seq);    $ligamer\_length **=** length $ligamer;    $ligamer\_length **=** $ligamer\_length**-**7;    }  **if** ($lig\_joiner\_code **eq** "I-L-C")    {    $ligamer **=** join ("",$Phos\_mod\_code,$T5\_seq,$barcode,$T3\_seq);    $ligamer\_length **=** length $ligamer;    $ligamer\_length **=** $ligamer\_length**-**7;    }  **if** ($ligamer\_length **>** 60)    {  **print** STDERR    "Warning! The ligamer from input file data line $."**.**    " has a length greater than 60!\n";    };  $results{pcrsequence} **=** $pcrsequence;  $results{ligamer} **=** $ligamer;  $results{ligamer\_length} **=** $ligamer\_length;  $results{warning} **=** $warning;  **return** %results;  }  *################################################################################*  *################################################################################*  *## Load the latest Ensembl Registry*  **sub ensembl\_database**{  **my** $verbose**=**shift@\_;  **my** $db\_version**=**shift@\_;  **my** $registry **=** 'Bio::EnsEMBL::Registry';  **print** "Beginning to login to Ensembl database version $db\_version.\n";  $registry**->**load\_registry\_from\_db    (  **-**host **=>** 'ensembldb.ensembl.org',  **-**user **=>** 'anonymous',  **-**db\_version **=>** $db\_version,  **-**verbose **=>** $verbose,    );  **print** "Done loading ensembl database.\n";  **return** $registry;  }  *################################################################################*  *################ BEGIN Subroutine to obtained get genomic sequence slice #######*  **sub get\_genomic\_sequence** {  **my** ($chr, $start, $end, $species, $db ) **=** @\_;  **my** $slice\_adaptor **=** $db**->**get\_adaptor( $species, 'Core', 'Slice');  $chr**=~**s/^chr//;  **my** $slice **=** $slice\_adaptor**->**    fetch\_by\_region      (      'chromosome',      $chr,      $start,      $end,      );  **my** $slice\_sequence **=** ($slice**->**seq);  **return** $slice\_sequence;  }  *################ END Subroutine to obtained get genomic sequence slice ######*  *################################################################################*  *################ BEGIN Subroutine to PROVIDE POD HELP DATA ######*  **sub podhelp** {  **my** $scriptname**=** shift@\_;  **my** $help\_print**=** shift@\_;  **my** $man\_print**=** shift@\_;  **my** $perlname**=**$scriptname;  **my** $htmlname**=**$scriptname;  **my** $manname**=**$scriptname;  **if** ($help\_print **eq** 1)    {    $htmlname **=~** s/\.pl/\.html/;    system "pod2html $perlname --title=$perlname --outfile=$htmlname";  **print** "\n\t$htmlname printed in cwd.\n\n";    exit    }  **if** ($man\_print **eq** 1)    {    $manname **=~** s/\.pl/\.man/;    system "pod2man $perlname $manname";  **print** "\n\t$manname printed in $dir.\n\n";    system "man -l $manname|less";    exit    }  }  *################ END Subroutine to PROVIDE POD HELP DATA ######* |

##### 5-ftl\_assembler.pl

This script takes a specific code and sequence of the ligamers, and determins all the permutations of those sequences, creating a FTL sequence fasta file that can be used to make an alignment index.

|  |
| --- |
| *#!/usr/bin/perl*  *# Written by: Alper Kucukural*  *# 06/14/10*  *#Modified by Christian Roy - 01.14.2011*  *=head1 FTL Assembler*  *This script will assemble Ligamers from a specificed input format into FTLs*  *Script orginally writen by Alper Kucukural. Modified by Christian Roy to*  *provide more information on assembler ligatio products*  *=head2 Contact information*  *Christian.Roy@umassmed.edu*  *=cut*  **use** strict;  **use** Data::Dumper;  **use** Cwd;  **my** $dir **=** getcwd;  **my** $timestamp **=** localtime();  $Data::Dumper::Indent **=** 1;  **my** $inputfile**=**"";  **my** $outputfile**=**"";  **my** $plname**=**$0;  **my** $man\_print**=**0;  **my** $help**=**0;  **my** $help\_print**=**0;  **my** %warnings;  **my** $cmd**=**$0**.**" "**.**join(" ",@ARGV); *####command line copy*  *# Parse the command line*  **if** (scalar @ARGV**==**0)  {  **print** "\n\tUsage:\n";  **print** "\t./$0.pl -i inputfile -o outfile\n";  **print** "\tFasta format writen to STDOUT\n";  **print** "\n\tHelp Information\n";  **print** "\t-h = HELP! - displays POD documentation for this script.\n"**.**  "\t-hp = Print HTML POD data for $plname\n"**.**  "\t-mp = Print and view Manual POD data for $plname\n\n";  **print** "\tWant sample input file?\n";  **print** "\tenter -s as a command line option\n\n";    exit;  }  **while**(scalar @ARGV **>** 0){  **my** $next\_arg **=** shift(@ARGV);  **if** ($next\_arg **eq** "-h") { $help**=**1;}  **if** ($next\_arg **eq** "-hp") { $help\_print**=**1;}  **if** ($next\_arg **eq** "-mp") { $man\_print**=**1;}  **if**($next\_arg **eq** "-i") { $inputfile **=** shift(@ARGV); }  **if**($next\_arg **eq** "-s") { sample\_input() }  **elsif**($next\_arg **eq** "-o") { $outputfile**=** shift(@ARGV); }  }  *## POD HELP SUBROUTINE CALLS*  **my** $scriptname**=**$0;  podhelp( $scriptname, $help, $help\_print, $man\_print, $dir);  *## POD HTML Subroutine CALLS*  *#Open the input file or die*  open IN, $inputfile **or** die "Input file could not be not openned";  *#Open the output file or fail*  open OUT, ">$outputfile" **or** die "Output file could not be not created";  *#Print header information to the output file*  **print** OUT "#The command line used was:\n";  **print** OUT "#$cmd\n"; *#Print command line*  **print** OUT "#Run on $timestamp\n"; *#print the time and date script run*  **print** OUT "#FTL-Isoform-Name\tOrder\tCode\t"**.** *#Print column header*  "Content\tSize\tSequence\tInput Ligamer Name\n";  *#Print header informatio nto the screen*  **print** "#The command line used was:\n";  **print** "#$cmd\n";  **print** "#Run on $timestamp\n";  **print** "#FASTA Format\n";  **my** @name\_array**=**"";  **my** %sequences**=**();  **my** %orders**=**();  **my** %codes**=**();  **my** %sizes**=**();  **my** %merged\_seqs**=**();  **my** %names**=**();  *## This part collects the data*  **while** (**my** $line**=**<IN>)  {    chomp($line);    $line**=~**s/\"//g; *#Corrects for Excel or open office inputs*    $line**=~**s/\,/\t/g; *#Corrects for Excel or open office inputs*    **if** ($line**=~**m/\#/) {**next**}; *#Skip header lines*    **my** @vals**=**split(/\t/,$line); *#Split input line based on tabs*  **my** $set**=**$vals[0]; *#collect the set name*  **my** $code**=**$vals[1]; *#collect the assembly code*  **my** $order**=**$vals[2]; *#collect the order code*  **my** $size**=**$vals[3];  **my** $seq**=**$vals[4];     $seq**=~**s/^\/5Phos\///g; *#substitute the modifcation code*  **my** $name**=**$vals[5];    **if** ($set**!~**/^$/)    {     push( @{ $sequences{ $set } }, $seq);     push( @{ $orders{ $set } }, $order);     push( @{ $codes{ $set } }, $code);     push( @{ $sizes{ $set } }, $size);     push( @{ $names{ $set } }, $name);    }  }  *#print Dumper %codes;*  **my** %merged\_seqs**=**();  **my** %merged\_orders**=**();  **my** %merged\_codes**=**();  **my** %merged\_sizes**=**();  **my** %merged\_names**=**();  **for** **my** $key ( keys %codes )      {  **my** $count**=**1;  **my** @my\_codes **=** @{ $codes{$key} };  **my** @my\_orders **=** @{ $orders{$key} };  **my** @my\_sizes **=** @{ $sizes{$key} };  **for** (**my** $j**=**0; $j**<**@my\_codes; $j**++**)        {  **my** $code **=** $my\_codes[$j];  **my** $order **=** $my\_orders[$j];  **my** $seq **=** ${ $sequences{ $key } }[$j];  **my** $size **=** ${ $sizes{ $key } }[$j];  **my** $name **=** ${ $names{ $key } }[$j];  *#print ">".$key.".$count.$code\n";*    **if** ($code**==**1)          {    **if** (defined(@{ $merged\_seqs{$key} }) **==** 0)             {                push(@{ $merged\_orders{$key} } , $order**.**"|");                push(@{ $merged\_orders{$key} }, " |");                push(@{ $merged\_codes{$key} } , $code**.**"|");                push(@{ $merged\_codes{$key} }, " |");                push(@{ $merged\_sizes{$key} } , $size**.**"|");                push(@{ $merged\_sizes{$key} }, " |");                push(@{ $merged\_seqs{$key} }, $seq**.**"");                push(@{ $merged\_seqs{$key} }, "");                push(@{ $merged\_names{$key} }, $name**.**"|");                push(@{ $merged\_names{$key} }, "|");             }  **else**             {  **my** $sized**=**@{ $merged\_seqs{ $key } };  **for**(**my** $i**=**0; $i**<**$sized; $i**++**)                {                   push(@{ $merged\_orders{$key} }, ${ $merged\_orders{$key} }[$i]**.**"\_|");                   ${$merged\_orders{$key}}[$i]**.=**$order**.**"|";                   push(@{ $merged\_codes{$key} }, ${ $merged\_codes{$key} }[$i]**.**"\_|");                   ${$merged\_codes{$key}}[$i]**.=**$code**.**"|";                     push(@{ $merged\_sizes{$key} }, ${ $merged\_sizes{$key} }[$i]**.**"\_|");                   ${$merged\_sizes{$key}}[$i]**.=**$size**.**"|";                   push(@{ $merged\_seqs{$key} }, ${ $merged\_seqs{$key} }[$i]**.**"");                   ${$merged\_seqs{$key}}[$i]**.=**$seq**.**"";                   push(@{ $merged\_names{$key} }, ${ $merged\_names{$key} }[$i]**.**"\_|");                   ${$merged\_names{$key}}[$i]**.=**$name**.**" | ";                }             }          }  **elsif** ($code**==**2)          {  *#print "$key.$count.[".keys( %merged\_seqs )."]\n";*  **if** (defined(@{ $merged\_seqs{$key} }) **==** 0)             {                push(@{ $merged\_orders{$key} }, $order**.**"|");                push(@{ $merged\_codes{$key} }, $code**.**"|");                push(@{ $merged\_sizes{$key} }, $size**.**"|");                push(@{ $merged\_seqs{$key} }, $seq**.**"");                push(@{ $merged\_names{$key} }, $name**.**"|");             }  **else**             {  **my** $sized**=**@{ $merged\_seqs{ $key } };  **for**(**my** $i**=**0; $i**<**$sized; $i**++**)                {                   ${$merged\_orders{$key}}[$i]**.=**$order**.**"|";                   ${$merged\_codes{$key}}[$i]**.=**$code**.**"|";                   ${$merged\_sizes{$key}}[$i]**.=**$size**.**"|";                   ${$merged\_seqs{$key}}[$i]**.=**$seq**.**"";                   ${$merged\_names{$key}}[$i]**.=**$name**.**"|";                }             }  *#print Dumper %merged\_seqs;*          }        $count**++**;        }      }  **for** **my** $key ( keys %merged\_seqs )      {  **my** $sized**=**@{ $merged\_seqs{ $key } };  **for**(**my** $i**=**0; $i**<**$sized; $i**++**)         {  *#FASTA out to Screen*  **print** ">"**.**$key**.**"."**.**($i**+**1)**.**"\n";  **print** ${ $merged\_seqs{$key} }[$i]**.**"\n";    *#Formatted to OUTPUT file*  **print** OUT ">"**.**$key**.**"."**.**($i**+**1)**.**",";  **print** OUT ${ $merged\_orders{$key} }[$i]**.**",";  **print** OUT ${ $merged\_codes{$key} }[$i]**.**",";  **print** OUT ${ $merged\_sizes{$key} }[$i]**.**",";  **my** $seqsize**=**length ${ $merged\_seqs{$key} }[$i];  **print** OUT $seqsize**.**",";  **print** OUT ${ $merged\_seqs{$key} }[$i]**.**"\n";         }      }  *## Begin Subroutine Section*  *## Being subroutine to print a sample input file*  **sub sample\_input** {  **print** "\n\n"; *#break from rest of terminal information*  **print** "#Header\n";  **print** "#Set Name\tCode\tOrder\tSize\tSequence\tLig-Name\n";  **print**  "L-I-19/20"**.**"\t"**.**"2"**.**"\t"**.**"1"**.**"\t"**.**"41"**.**"\t"**.**"ATCTGAGCGGGCTGGCAAGGCCCAAAGGACCAGGTGAGGCC"**.**"\t"**.**"E21-C-T"**.**"\n"**.**  "L-I-19/20"**.**"\t"**.**"2"**.**"\t"**.**"2"**.**"\t"**.**"47"**.**"\t"**.**"/5Phos/CAGGAGACACAGCAGGGCGAACGCGCCAAGTACCCAAGAC"**.**"\t"**.**"E21-C"**.**"\n"**.**  "L-I-19/20"**.**"\t"**.**"1"**.**"\t"**.**"3"**.**"\t"**.**"34"**.**"\t"**.**"/5Phos/TTGATAAATGGCTTATTATCTTCATAG"**.**"\t"**.**"E20"**.**"\n"**.**  "L-I-19/20"**.**"\t"**.**"1"**.**"\t"**.**"4"**.**"\t"**.**"52"**.**"\t"**.**"/5Phos/CCAGGTAAGTCCGTATTATAGTAGCCATCACAAACACGGTAATTA"**.**"\t"**.**"E19"**.**"\n"**.**  "L-I-19/20"**.**"\t"**.**"2"**.**"\t"**.**"5"**.**"\t"**.**"56"**.**"\t"**.**"/5Phos/TTAATGTTTTCCAACCTGCTAGAATCTGGGCCTCCCTCGCGCCATCAGA"**.**"\t"**.**"E18"**.**"\n"**.**  "L-I-E22-24"**.**"\t"**.**"2"**.**"\t"**.**"1"**.**"\t"**.**"44"**.**"\t"**.**"ATCTGAGCGGGCTGGCAAGGCCCGGGCATTGTTCAGTGAATGTC"**.**"\t"**.**"E25"**.**"\n"**.**  "L-I-E22-24"**.**"\t"**.**"1"**.**"\t"**.**"2"**.**"\t"**.**"39"**.**"\t"**.**"ATTATGTCTGTCTGCTCAGGCTATCTGGCCTAAGAGTAG"**.**"\t"**.**"E24/25k"**.**"\n"**.**  "L-I-E22-24"**.**"\t"**.**"1"**.**"\t"**.**"3"**.**"\t"**.**"18"**.**"\t"**.**"TTGTCTCTCTGTAGGTTG"**.**"\t"**.**"E24/25a"**.**"\n"**.**  "L-I-E22-24"**.**"\t"**.**"1"**.**"\t"**.**"4"**.**"\t"**.**"53"**.**"\t"**.**"CTCTAAATGATATAAATCTTGAAGATGGGCTCTCGGACCATGTGATGTCAGTC"**.**"\t"**.**"E24/25b"**.**"\n"**.**  "L-I-E22-24"**.**"\t"**.**"1"**.**"\t"**.**"5"**.**"\t"**.**"55"**.**"\t"**.**"CTGGCGAGTTAAACGCAGGAGACTACTTTTGTTAGTAGGTAATTGCCAGTCATCC"**.**"\t"**.**"E24"**.**"\n"**.**  "L-I-E22-24"**.**"\t"**.**"2"**.**"\t"**.**"6"**.**"\t"**.**"47"**.**"\t"**.**"/5Phos/CTGTGTGCCAGAGGAGTACCGAGCACTGGTGCGGGTGGTG"**.**"\t"**.**"E22-c"**.**"\n"**.**  "L-I-E22-24"**.**"\t"**.**"2"**.**"\t"**.**"7"**.**"\t"**.**"48"**.**"\t"**.**"/5Phos/GATGCCTTCACAGAGCTGTGCGCCTCCCTCGCGCCATCAGA"**.**"\t"**.**"E22-c-t"**.**"\n";  exit;  }  *################ BEGIN Subroutine to PROVIDE POD HELP DATA ######*  **sub podhelp** {  **my** $scriptname**=** shift@\_;  **my** $help**=** shift@\_;  **my** $help\_print**=** shift@\_;  **my** $man\_print**=** shift@\_;  **my** $perlname**=**$scriptname;  **my** $htmlname**=**$scriptname;  **my** $manname**=**$scriptname;  **if** ($help **eq** 1) {system "pod2text $scriptname | less"; die}  **if** ($help\_print **eq** 1)  {  $htmlname **=~** s/\.pl/\.html/;  system "pod2html $perlname --title=$perlname --outfile=$htmlname";  **print** "\n\t$htmlname printed in cwd.\n\n";  exit}  **if** ($man\_print **eq** 1)  {  $manname **=~** s/\.pl/\.man/;  system "pod2man $perlname $manname";  **print** "\n\t$manname printed in $dir.\n\n";  system "man -l $manname|less";  exit}  }  *=head1 Input file format*  *1) Input files are in tab-delimiated formats*  *a) First column should contain a Unique set name*  *This name is appended to all possible assembles of*  *this set of ligamers*  *b) Second column contains a ligamer code*  *2 = Always include (i.e. terminal or connecting)*  *1 = sometimes include (i.e. cassette ligamers)*  *c) Order code - 1,2,3,4*  *This tells the program how to arrange the ligamers*  *1 goes before 2 goes before 3, etc...*  *d) The input size of the ligamer*  *i.e. "20","60" - some number of nt for that*  *particular ligamer*  *e) The sequence of the ligamer used*  *"ATGCGATGCATC"*  *if it starts with a /5Phos/ modification code*  *this will be removed*    *=head1 Output file format*  *1) comma-separated list*  *a) FTL-Isoform-Unique-Name*  *b) Input Order arrangement*  *b) Input Code arrangement*  *c) Input ligamer size arrangement*  *d) Assembled FTL Sequences*  *c) Assembler Input Ligamer name arrangement*  *=cut* |