```
#/bin/tcsh

#       Appendix S1. Detailed target enrichment probe design protocol


#                       Building_exon_probes.sh
#       Workflow and script for development of Hyb-Seq target probes


###############################################################################
#
#                   *** PLEASE CONTINUE READING ***
#
#THIS FILE IS BOTH A DETAILED DESCRIPTION OF THE HYB-SEQ PROBE DESIGN PROCESS,
#AND A PROGRAM THAT CAN BE RUN IN A LINUX ENVIRONMENT.
#
#COMMENTS ARE CONTAINED ON LINES BEGINNING WITH A # SYMBOL.
#
###############################################################################


#Disclaimer:
#Although these commands should function with just the input of two fasta
#files, genome.fasta and transcriptome.fasta, their proper execution is not
#guaranteed. Given the idiosyncrasies in file formats and operating
#environments, these commands are meant more as a starting point, to be
#modified as needed by the user. For example, the presence of spaces or tabs in
#the ID line of the fasta files may cause problems downstream.

#Copyright (c) 2014
#K. Weitemier, S.C.K. Straub, R. Cronn, M. Fishbein, R. Schmickl, A. McDonnell,
#and A. Liston.
#This script is free software: you can redistribute it and/or modify it under
#the terms of the GNU General Public License as published by the Free Software
#Foundation, either version 3 of the License, or (at your option) any later
#version. A copy of this license is available at <http://www.gnu.org/licenses/>.
#Great effort has been taken to make this software perform its said
#task, however, this software comes with ABSOLUTELY NO WARRANTY,
#not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

#TO RUN THIS SCRIPT (in a Linux environment):
#Copy this script into a new directory along with:
#1) A fasta file of genomic contigs, named "genome.fasta"
#2) A fasta file of transcriptome contigs, named "transcriptome.fasta"
#3) the file provided with this script named "grab_singleton_clusters.py" and
#4) the file provided with this script named "blat_block_analyzer.py"
#You may need to make this script and the other two programs executable, or able
#to be recognized as programs. To do this, run the following command:
#
#  chmod +x Building_exon_probes.sh grab_singleton_clusters.py blat_block_analyzer.py
#
#Finally, to run the script, type the following command:
#
```

```
#  ./Building_exon_probes.sh
#
#Additional notes: This script calls two third-party programs, BLAT and
#CD-HIT-EST. These need to be installed on your system so that they can be
#called simply from the commands "blat" and "cd-hit-est", respectively. It is
#also necessary to have Python 2.x installed to run grab_singleton_clusters.py
#and blat_block_analyzer.py.

###############################################################################
#Match genome and transcriptome sequences
###############################################################################
#Here we use BLAT to identify transcripts that have high identity to genomic
#contigs. An important consideration when preparing the input genome and
#transcriptome assemblies is to first remove those contigs that match the
#chloroplast or mitochondrial genomes, so that they are not targeted in the same
#pool as nuclear loci. The presence of multiple copies of these genomes in
#prepared libraries, relative to the nuclear genome, will skew the ratio of
#captured fragments by an equal proportion (e.g., if 100 copies of a targeted
#chloroplast locus are present relative to a targeted nuclear locus, the pool of
#enriched fragments, and sequenced reads, will over-represent the chloroplast
#locus by about 100:1). High copy loci such as the chloroplast will represent a
#large fraction of off-target sequenced reads and can likely be assembled
#without enrichment. If enrichment of organellar or other high copy regions is
#desirable (as might be required for very high multiplexing), target enrichment
#should be done in a separate reaction using a separate set of probes. After
#enrichment, the individual pools representing different targeted fractions
#(e.g., nuclear and chloroplast targets) can be re-mixed at desired ratios for
#multiplex sequencing.

#As a default in this step the matching identity is set very high (99%) because
#the transcripts and genomic contigs are expected to come from the same
#individual. If this is not the case, particularly if the transcriptome and
#genome originate from different taxa, you should reduce the stringency by
#modifying the "-minIdentity" flag.

#The fasta files should be formatted so that each sequence takes up exactly two
#lines: the ID line and the sequence line.

#This step may take a very long time.

echo """Comparing genome and transcriptome. This may take a very long time."""

date

blat genome.fasta transcriptome.fasta -tileSize=8 -minIdentity=99 -noHead -out=pslx
genome_v_transcriptome.pslx


###############################################################################
#Find and extract transcriptome sequences with only one match against the genome
###############################################################################
#This step removes transcripts that have matches against more than one genomic
#contig, based on the assumption that multiple hits may indicate loci with
#several copies in the genome. However, this step may exclude loci that are
#truly single copy in cases where a locus is split across more than one genomic
```

#contig in a fragmented genome assembly.

```
echo """Finding and extracting matches with a single hit."""

date

cut -f10 genome_v_transcriptome.pslx | sort | uniq -c | grep '      1 ' | sed -e
's/      1 /\\\</' -e 's/$/\\\>/' > single_hits_vs_genome.txt

grep -f single_hits_vs_genome.txt genome_v_transcriptome.pslx >
single_hit_genome_v_transcriptome.pslx
```

```
################################################################################
```

#Keep only matches hitting at least 95% of the transcript.

```
################################################################################

echo """Finding and extracting transcripts with single, substantial hits."""

date

awk '{if (($13-$12) >= ($11 * 0.95)) print $0}'
single_hit_genome_v_transcriptome.pslx >
whole_gene_single_hit_genome_v_transcriptome.pslx
```

#Extract these transcripts from the transcriptome assembly

```
cut -f10 whole_gene_single_hit_genome_v_transcriptome.pslx >
single_whole_hits_vs_genome.txt

sed -i'' -e 's/^/^>/' -e 's/$/\\\>/' single_whole_hits_vs_genome.txt

grep -A1 -f single_whole_hits_vs_genome.txt transcriptome.fasta | sed '/^--$/d' >
single_transcript_hits.fasta
```

```
################################################################################
```

#Cluster any nested transcripts

```
################################################################################
```

#In some cases, one transcript may exactly match, but be nested within, another
#transcript. This program finds such transcripts and retains only the largest
#sequence.

```
echo """Clustering transcripts with 100% identity."""

date

cd-hit-est -i single_transcript_hits.fasta -o
single_transcript_hits_cluster_100.fasta -d 0 -c 1.0 -p 1 >
cluster_100_single_transcript_hits_log.txt
```

```
################################################################################
```

#Remove transcripts with 90% or greater similarity

```
################################################################################
```

#This step removes transcripts that share high identity. This should help reduce
#the number of targeted loci with multiple copies within the genome, and reduce
#the chance of capturing similar, but off-target, loci.

```
echo """Clustering and removing transcripts with 90% identity."""

date

cd-hit-est -i single_transcript_hits_cluster_100.fasta -o
single_transcript_hits_cluster_90.fasta -d 0 -c 0.9 -p 1 -g 1 >
cluster_90_single_transcript_hits_log.txt

python grab_singleton_clusters.py -i single_transcript_hits_cluster_90.fasta.clstr
-o unique_single_transcript_hits_cluster_90.fasta.clstr

grep -v '>Cluster' unique_single_transcript_hits_cluster_90.fasta.clstr | cut -d' '
-f2 | sed -e 's/\.\.\./\\\>/' -e 's/^/^/' > unique_single_transcript_hits

grep -A1 -f unique_single_transcript_hits single_transcript_hits_cluster_100.fasta
| sed '/^--$/d' > unique_single_transcript_hits.fasta

sed -i'' -e 's/\^>/\\\</' unique_single_transcript_hits

grep -f unique_single_transcript_hits
whole_gene_single_hit_genome_v_transcriptome.pslx > unique_single_hits.pslx


################################################################################
#Find loci and exons that meet length requirements
################################################################################
#This step finds every remaining locus and all the remaining exons that make up
#that locus. If an exon is above a specified length, the program adds it to the
#total length for that locus. It outputs the sequences of all the large exons
#for each locus that exceeds a specified minimum length. The default minimum
#length for exons is 120 bp, which matches the length of the probe oligos used
#in the target capture reaction. It can be modified by altering the "-s" flag.
#The default minimum length for loci is 960 bp. This was chosen because it is a
#multiple of 120 and was considered a large enough length for reliable gene tree
#estimation. It can be modified by altering the "-l" flag.

echo """Finding loci and exons that meet length requirements."""

date

python blat_block_analyzer.py -i unique_single_hits.pslx -o
large_enough_unique_single_hits.fasta -l 960 -s 120


################################################################################
#Remove exons with 90% or greater similarity
################################################################################
#Finally, any remaining individual exons that share >=90% identity are removed.
#This is to prevent any problems caused by the cross-enrichment of similar
#exons across different loci.

echo """Removing individual exons with high identity."

date

cd-hit-est -i large_enough_unique_single_hits.fasta -o
large_enough_unique_single_hits_cluster90.fasta -d 0 -c 0.9 -p 1 -g 1 >
cluster_90_large_enough_unique_single_hits_log.txt

python grab_singleton_clusters.py -i
large_enough_unique_single_hits_cluster90.fasta.clstr -o
```

```
unique_blocks_large_single_hits_cluster90.fasta.clstr

grep -v '>Cluster' unique_blocks_large_single_hits_cluster90.fasta.clstr | cut -d'
' -f2 | sed -e 's/\.\.\./\\\>/' -e 's/^/^/' > unique_blocks_large_single_hits

grep -A1 -f unique_blocks_large_single_hits large_enough_unique_single_hits.fasta |
sed '/^--$/d' > blocks_for_probe_design.fasta

echo """Process complete."""

date


#The final output file, blocks_for_probe_design.fasta, contains sequences of
#each exon for each locus that are thought to be low-copy within the genome and
#meet the minimum length standards. The sequences that are output are ultimately
#derived from the original genome assembly (as opposed to the transcriptome
#assembly, if there are differences between the two).
#The ID line of each sequence is comma delimited and contains the name of the
#locus it originates from (from the transcriptome ID), the name of the genomic
#contig it matches followed by an underscore and a number indicating the exon
#within the locus (exon numbering within a locus is arbitrary), and the length
#of the exon. As an example, the following ID line identifies the second exon
#found within transciptome locus m.33568 and genome contig 5193133, which has a
#length of 186 bp:
#
#  >m.33568,5193133_2,186
#  ctca...

#End of File.
```