```perl
#!/usr/bin/perl
use strict;
use warnings;
use File::Basename;

###############################################
# SCRIPT TO QUICKLY BLAST S5_genome_XXX sequence
###############################################
# e.g RUN ON VITAL-IT (need a database...)
# BEFORE RUNNING THE SCRIPT, TYPE IN TERMINAL:
# module add Blast/ncbi-blast/latest;
# USAGE:
# ./blastSeq.pl cdsfile n sequence_idX
#   -> cdsfile => the file with all CDS to retrieve the sequences of the corresponding id
#   -> n => the number of query locus
#   -> sequence_idX => n sequence ids
# EXAMPLE:
# ./blastSeq.pl  Pseud_S5_CDS.fasta 2 S5_genome_4542 S5_genome_900
# OUTPUT
# -> txt file created in current directory for all blast query
# -> print on terminal screen the 5 top results for each query

my $cds = $ARGV[0];
my @query_id = @ARGV[2..($ARGV[1]+1)];
my %all_seq;
my $seq = "";
my $id;
my $query_file = "S5_query.fasta";

# for vital-it:
# do not know why but it doesn't work !!!
# should be type before launching the script
system("module add Blast/ncbi-blast/latest;");

open(CDS, $cds) || die;
while(my $line = <CDS>){
    chomp($line);
    if($line =~ /^>/){

    if(length $seq){          # add to the dict, but do not do that for the 1st iteration
        $all_seq{$id} = $seq;
        $seq = "";

#       print ("hello");
#       last;
    }
        ($id = $line) =~ s/^>//; # store the id without the > that starts the line

    } else{
    $seq .= $line;  # concatenate the sequence parts

    }
}
# do not forget the last sequence
$all_seq{$id} = $seq;
close(CDS);


system("rm -f $query_file");   # because we append, make sure to start from scratch
system("touch $query_file");


foreach my $pos(@query_id){
#    print("$pos\n");
    open(my $queryF, '>>', $query_file) or die("open: $$");
    print($queryF ">$pos\n$all_seq{$pos}\n" );
#    print $x;
}


####### BLAST QUERY
#####################
my $command;
my $blast_output = join("_", @query_id)."_blast.txt";

#$command = "formatdb -i swissprot -p T -o T";
#system($command);

#$command = "blastx -db swiss -query $query_file -out $blast_output";
# -p programm name, -d database, -i query file name
#-max_target_seqs 1


$command = "blastx -db EXPASY/UPKB/UniProtKB -query $query_file -out $blast_output";

system($command);

open(BLAST, $blast_output) || die;
my $k = 0;
my $next;
while(my $line = <BLAST>){
    chomp($line);
    if($line =~ /Query=/){
    print "$line\n";
    $k = 0;        # I want to print the 5 best alignments
    }
    if ($line =~ /Sequences producing/){    # this line will be printed, if not wanted, put at the end
of the loop
    $next = 1;
    }
    if(length $line and $next and $k<6 ){   #line not empty # 5 seq + 1 line Sequences producing...
    print "$line\n";
    $k ++;
    }
    if($k==6){
    $next = 0;
    }
}
close(BLAST);
```

```perl
#!/usr/bin/perl
use strict;
use warnings;
use File::Basename;
###############################################
# SCRIPT TO CALCULATE SOME GENE PARAMETERS
###############################################
##### Spring 2016 - MLS - UNIL - Marie Zufferey
# USAGE :
# ./stat_CDS.pl  my_cds.fasta outfile.txt
#   -> my_cds.fasta  => file with the CDS
#   -> outfile.txt   => file in which output written !!!! if already exist, will be overwritten !
# output as follow (tab-separated):
#   -> the tag of the sequence, its length, the ratio of purine and the ratio of GC-content (for the
whole gene sequence):
#   SEQ_TAG     LENGTH      ratioGC     ratioPu

my $cds = $ARGV[0];
my $outfile = $ARGV[1];

print("WARNING: outfile will be overwritten !\n");
system("rm -f $outfile");
system("touch $outfile");

# write the header in the output file
open(my $out, '>>', $outfile) or die("open: $$");
print($out "Seq_tag\tLength\tratioGC\tratioPu\n");
close($out);

my ($id, $seq, $nG, $nC, $nA, $size, $nPu, $nGC);

open(CDS, $cds) || die;
while(my $line = <CDS>){
    chomp($line);
    if($line =~ /^>/){  # => it is the ID line

        if(length $seq){          # we have a new ID (all IDs except the 1st)

            # calculate and write the sequence information in output file
            $nG = ($seq =~ tr/gG//);
            $nC = ($seq =~ tr/cC//);
            $nA = ($seq =~ tr/aA//);
            $size = length($seq);
            $nPu = ($nA+$nG)/$size;
            $nGC = ($nC+$nG)/$size;
            open(my $out, '>>', $outfile) or die("open: $$");
            print($out "$id\t$size\t$nGC\t$nPu\n" );
            close($out);

            # reinitialize the seq
            $seq = "";

        }
        ($id = $line) =~ s/^>//; # store the id without the > that starts the line

    } else{         # => it is one seq line (can be separated with \n ...)
    $seq .= $line;  # concatenate the sequence parts

    }
}
# do not forget the last sequence  # TODO put in subroutine to avoid repetition of code...

# calculate and write the sequence information in output file
$nG = ($seq =~ tr/gG//);
$nC = ($seq =~ tr/cC//);
$nA = ($seq =~ tr/aA//);
$size = length($seq);
$nPu = ($nA+$nG)/$size;
$nGC = ($nC+$nG)/$size;
open($out, '>>', $outfile) or die("open: $$");
print($out "$id\t$size\t$nGC\t$nPu\n" );
close($out);

close(CDS);
```

```
#foreach my $pos(@query_id){
#    print("$pos\n");
 #   open(my $queryF, '>>', $query_file) or die("open: $$");
 #   print($queryF ">$pos\n$all_seq{$pos}\n" );
#    print $x;
#}
```

```perl
#!/usr/bin/perl
use strict;
use warnings;
use File::Basename;

###############################################
# SCRIPT TO CALCULATE SOME GENE PARAMETERS -> 3d codon position only !
###############################################
##### Spring 2016 - MLS - UNIL - Marie Zufferey
# USAGE :
# ./stat_CDS_3dpos.pl  my_cds.fasta outfile.txt
#   -> my_cds.fasta  => file with the CDS
#   -> outfile.txt   => file in which output written !!!! if already exist, will be overwritten !
# output as follow (tab-separated):
#   -> the tag of the sequence, its length, the ratio of purine and the ratio of GC-content (for the 
3d codon position only):
#   SEQ_TAG     LENGTH      ratioGC     ratioPu

my $cds = $ARGV[0];
my $outfile = $ARGV[1];


print("WARNING: outfile will be overwritten !\n");
system("rm -f $outfile");
system("touch $outfile");

# write the header in the output file
open(my $out, '>>', $outfile) or die("open: $$");
print($out "Seq_tag\tn3dpos\tratioGC\tratioPu\n");
close($out);

my ($id, $seq, $nG, $nC, $nA, $size, $nPu, $nGC);

open(CDS, $cds) || die;
while(my $line = <CDS>){
    chomp($line);
    if($line =~ /^>/){  # => it is the ID line

        if(length $seq){          # we have a new ID (all IDs except the 1st)

            # select every 3d character only
            $seq =~ s/..(.)/$1/g;
            # calculate and write the sequence information in output file
            $nG = ($seq =~ tr/gG//);
            $nC = ($seq =~ tr/cC//);
            $nA = ($seq =~ tr/aA//);
            $size = length($seq);
            $nPu = ($nA+$nG)/$size;
            $nGC = ($nC+$nG)/$size;
            open(my $out, '>>', $outfile) or die("open: $$");
            print($out "$id\t$size\t$nGC\t$nPu\n" );
            close($out);

            # reinitialize the seq
            $seq = "";

        }
        ($id = $line) =~ s/^>//; # store the id without the > that starts the line

    } else{         # => it is one seq line (can be separated with \n ...)
    $seq .= $line;  # concatenate the sequence parts

    }
}
# do not forget the last sequence  # TODO put in subroutine to avoid repetition of code...

# calculate and write the sequence information in output file
$seq =~ s/..(.)/$1/g;
$nG = ($seq =~ tr/gG//);
$nC = ($seq =~ tr/cC//);
$nA = ($seq =~ tr/aA//);
$size = length($seq);
$nPu = ($nA+$nG)/$size;
$nGC = ($nC+$nG)/$size;
```

```
open($out, '>>', $outfile) or die("open: $$");
print($out "$id\t$size\t$nGC\t$nPu\n" );
close($out);

close(CDS);


#foreach my $pos(@query_id){
#    print("$pos\n");
 #   open(my $queryF, '>>', $query_file) or die("open: $$");
 #   print($queryF ">$pos\n$all_seq{$pos}\n" );
#    print $x;
#}
```

```perl
#!/usr/bin/perl
use strict;
use warnings;
use File::Basename;

###############################################
# SCRIPT TO CALCULATE SOME GENE PARAMETERS -> first 2 codon positions only !
###############################################
##### Spring 2016 - MLS - UNIL - Marie Zufferey
# USAGE :
# ./stat_CDS_12dpos.pl  my_cds.fasta outfile.txt
#   -> my_cds.fasta  => file with the CDS
#   -> outfile.txt   => file in which output written !!!! if already exist, will be overwritten !
# output as follow (tab-separated):
#   -> the tag of the sequence, its length, the ratio of purine and the ratio of GC-content (for the
first 2 codon positions only):
#   SEQ_TAG     LENGTH      ratioGC     ratioPu

my $cds = $ARGV[0];
my $outfile = $ARGV[1];


print("WARNING: outfile will be overwritten !\n");
system("rm -f $outfile");
system("touch $outfile");

# write the header in the output file
open(my $out, '>>', $outfile) or die("open: $$");
print($out "Seq_tag\tn12pos\tratioGC\tratioPu\n");
close($out);

my ($id, $seq, $nG, $nC, $nA, $size, $nPu, $nGC);

open(CDS, $cds) || die;
while(my $line = <CDS>){
    chomp($line);
    if($line =~ /^>/){  # => it is the ID line

        if(length $seq){          # we have a new ID (all IDs except the 1st)

            # select every 3d character only
            $seq =~ s/(..)./$1/g;
            # calculate and write the sequence information in output file
            $nG = ($seq =~ tr/gG//);
            $nC = ($seq =~ tr/cC//);
            $nA = ($seq =~ tr/aA//);
            $size = length($seq);
            $nPu = ($nA+$nG)/$size;
            $nGC = ($nC+$nG)/$size;
            open(my $out, '>>', $outfile) or die("open: $$");
            print($out "$id\t$size\t$nGC\t$nPu\n" );
            close($out);

            # reinitialize the seq
            $seq = "";

        }
        ($id = $line) =~ s/^>//; # store the id without the > that starts the line

    } else{         # => it is one seq line (can be separated with \n ...)
    $seq .= $line;  # concatenate the sequence parts

    }
}
# do not forget the last sequence  # TODO put in subroutine to avoid repetition of code...

# calculate and write the sequence information in output file
$seq =~ s/(..)./$1/g;
$nG = ($seq =~ tr/gG//);
$nC = ($seq =~ tr/cC//);
$nA = ($seq =~ tr/aA//);
$size = length($seq);
$nPu = ($nA+$nG)/$size;
$nGC = ($nC+$nG)/$size;
```

```
open($out, '>>', $outfile) or die("open: $$");
print($out "$id\t$size\t$nGC\t$nPu\n" );
close($out);

close(CDS);


#foreach my $pos(@query_id){
#    print("$pos\n");
 #   open(my $queryF, '>>', $query_file) or die("open: $$");
 #   print($queryF ">$pos\n$all_seq{$pos}\n" );
#    print $x;
#}
```

```perl
#!/usr/bin/perl
use strict;
use warnings;
use File::Basename;

##############################################
# SCRIPT TO PARSE BLAT RESULTS
##############################################
##### Spring 2016 - MLS - UNIL - Marie Zufferey
# Script to use after having run blat, e.g.:
#blat -t=dna -q=dna -noHead Pf5_cds_foo.fasta ../PseudS5_query.fasta
# USAGE EXAMPLE:
# ./parse_psl.pl ../data/BLAT_OUTPUT.psl ../data/Pf5_cds.fasta annotation.csv Pf5

my $blat_result = $ARGV[0];
my $db_fasta = $ARGV[1];
my $annotDB = $ARGV[2];
my $nameDB = $ARGV[3];
my $command = "";
my $outfile = "S5vs$nameDB.txt";

##############
# RETRIEVE ANNOTATIONS/FUNCTIONS FROM THE DB (.CSV)
(my $annotDBcut = $annotDB) =~ s/.csv/_cut.csv/;
$command = "cut -d \",\" -f3,9 $annotDB | tail -n +4 > $annotDBcut";  # first 3 lines are comments
system($command);
my %dbID_fction;
my $function ="";
open(CSV, $annotDBcut) || die;
while(my $line = <CSV>){
    chomp($line);
    $line =~ s/\"//g;  # not forget the g for global !
    my @line = split /,/, $line;
    my $geneID = $line[0];
    my $function = $line[1];


    if(not length $function){
        $function = "NA";
    }
    if(exists($dbID_fction{$geneID}) and $dbID_fction{$geneID} ne "NA"){
        my $prev_function = $dbID_fction{$geneID};
        my $new_function = "$prev_function,$function";  # if already exist => function1,function2
        $dbID_fction{$geneID} = $new_function;

    }else{
        $dbID_fction{$geneID} = $function;    # but some $geneID comme more than one time !!!
    }

}
close(CSV);


(my $blat_result_sorted = $blat_result) =~ s/.psl/_sorted.psl/;

system("sort -k 1 -r -n $blat_result > $blat_result_sorted");

### create the file
# S5_genome_id  $nameDB  Function
#S5_genome   PFL_0001    replication initiator


open(my $tempf, '>', $outfile) or die("open: $$");
my $first_line = "S5_genome_id\t$nameDB\tFunction\n";
print($tempf $first_line);


open(PSL, $blat_result_sorted) || die;

while(my $line = <PSL>){
    chomp($line);
    my @line = split /\t/, $line;
    my $queryID = $line[9];                 # id from our genome (S5_genome_87)
    my $dbIDnr = $line[13];                     # => but the ID we retrieve for the DB is not the
appropriate one (e.g. 1893893)

    $command = "grep \">$dbIDnr\\s\" $db_fasta";  # retrieve the gene id for the number of the DB
    my $retrieveID = `$command`;                 # => >1893893 gene=PFL_0001  # in the csv we have
PFL_0001

    $retrieveID =~ s/^\s+|\s+$//g;           #trim

    my @fastaID = split /=/, $retrieveID;
    my $dbID = $fastaID[1];

    if(exists($dbID_fction{$dbID})){
        print ($tempf "$queryID\t$dbID\t$dbID_fction{$dbID}\n");
    } else {
        print ($tempf "$queryID\t$dbID\tNA\n");
       print "not found";
    }

}
close($tempf);
close(PSL);
```