*#.....................License.................................*

*# MysiRNA-Designer a Software for Rational siRNA Design*

*# Copyright (C) 2011 <M.Mysara et al>*

*# This program 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 2 of the License, or*

*# (at your option) any later version.*

*# This program is distributed in the hope that it will be useful,*

*# but WITHOUT ANY WARRANTY; without even the implied warranty of*

*# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the*

*# GNU General Public License for more details.*

*# You should have received a copy of the GNU General Public License*

*# along with this program; if not, write to the Free Software*

*# Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA*

*#......................Packages Used..........................*

**use** **Getopt::**Std**;**

**use** Cwd**;**

**use** strict**;**

**use** **Bio::DB::**GenBank**;**

**use** **Bio::**Seq**;**

**use** **Bio::**SeqIO**;**

**use** **Bio::Tools::Run::**RemoteBlast**;**

**use** **Bio::Tools::Run::Alignment::**Clustalw**;**

**use** **Bio::**AlignIO**;**

**use** **Bio::**SimpleAlign**;**

**use** **Spreadsheet::**ParseExcel**;**

**use** **Win32::**OLE qw(in with)**;**

**use** **Win32::OLE::**Const 'Microsoft Excel'**;**

**use** **Modern::**Perl**;**

**use** **Bio::DB::**EUtilities**;**

**use** **Bio::**SearchIO**;**

require **LWP::**UserAgent**;**

*#.................adjusting the PATH to the current PATH...........................*

**my** $dir **=** cwd**;** **use** Cwd 'chdir'**;** chdir "/tmp"**;**

*#......................Creating directories ......................*

mkdir "results"**;**

mkdir "RNAxsresult"**;**

mkdir "MysiRNA\_Designer\_Final\_results"**;** system "del /Q /s MysiRNA\_Designer\_Final\_results > temp.txt"**;**

mkdir "blastresult"**;**

mkdir "Multi\_score\_Accepted"**;** system "del /Q /s Multi\_score\_Accepted > temp.txt"**;**

*#...................Varriables...................*

**my** $num**=0;***#wholemain*

**my** $trackernum**=0;***# fasta1*

**my** %opts**;**

**my** $header**;**

**my** $database **=** **new** **Bio::DB::**GenBank**;**

**my** $trans\_num**=0;**

*#....................................*

getopt**(**'studowpei'**,\**%opts**);**

**if(**defined**(**$opts**{**s**})&&**$opts**{**o**}&&**$opts**{**w**})**

**{**

**my** $off\_target\_filtration**=1;***# to perform off-target filtration or not as per used requirement*

**if(!**defined**(**$opts**{**t**}))** **{**$opts**{**t**}=0.01157;** **}**

**if(!**defined**(**$opts**{**u**}))** **{**$opts**{**u**}=0.001002;** **}**

**if(!**defined**(**$opts**{**p**}))** **{**$opts**{**p**}=**"N"**;** **}**

**if(!**defined**(**$opts**{**e**}))** **{**$opts**{**e**}=**"ENST\_UTR.txt"**;** **}**

**if(!**defined**(**$opts**{**i**}))** **{**$opts**{**i**}=**"ENST.txt"**;** **}**

**if(!**defined**(**$opts**{**d**}))** **{**$off\_target\_filtration**=0;**$opts**{**d**}=**"None"**;** **}**

*#............................................*

open Main**,** $opts**{**s**};**

**while** **(my** $acc**=**<Main>**){**

$num**++;**

**print** '

||||||||||||||||||||||||||||||||||||||||||||||||

|| Welcome To MysiRNA-Designer ||

|| A Software For Rational siRNA Design ||

|| Copyright (C) 2011 <M.Mysara et al> ||

||||||||||||||||||||||||||||||||||||||||||||||||

MysiRNA-Designer version 1, Copyright (C) 2011, M.Mysara et al

MysiRNA-Designer comes with ABSOLUTELY NO WARRANTY.

This is free software, and you are welcome to redistribute it

under certain conditions; type `notepad copying.txt\' for details.'**;**

*#............Delete Buffer................*

system "del /Q /s results > temp.txt"**;**

system "del /Q /s RNAxsresult > temp.txt"**;**

system "del /Q /s blastresult > temp.txt"**;**

system "del /Q /s Multi\_score\_Accepted > temp.txt"**;**

system "del /Q /s MysiRNA\_Designer\_Final\_results > temp.txt"**;**

*#..............................................................*

**my** $seq **=** $database**->**get\_Seq\_by\_id**(**$acc**);**

**print** "\nSeq: "**,** $seq**->**accession\_number**(),** " -- "**,** $seq**->**desc**(),** "\n\n"**;**

**my** $Desc**;**

**my** $Desc\_**;**

**my** $habal**=**$seq**->**desc**();**

**if** **(**$habal**=~**/(^[\w\W]\*, transcript variant)/**)** **{**

$Desc**=**$1**;**

**}**

*#make inputs for scoring (Iscore)*

**my** $out **=** **Bio::**SeqIO**->**newFh **(** **-**file **=>** ">"**.**$ENV**{**'PWD'**}.**"/results/fasta1result.fasta"**,** **-**format **=>** 'fasta'**);**

open FH**,**">"**,**$ENV**{**'PWD'**}.**'\results\iscoreinput.fasta'**;** **print** FH ""**;**close FH**;**

open FH**,**">>"**,**$ENV**{**'PWD'**}.**'\results\iscoreinput.fasta'**;**

**print** $out $seq**;**

**print** FH $seq**->**seq**;**

*#Make SNPs Database search input*

**my** $out\_ **=** **Bio::**SeqIO**->**newFh **(** **-**file **=>** ">"**.**$ENV**{**'PWD'**}.**"/results/fasta1result.gb"**,** **-**format **=>** 'GenBank'**);**

**print** $out\_ $seq**;**

close **(**FH**);**

open FH**,**">"**,**$ENV**{**'PWD'**}.**'\AlignIO\_.fasta'**;** **print** FH ""**;**close FH**;**

open FH**,**">>"**,**$ENV**{**'PWD'**}.**'\AlignIO\_.fasta'**;**

**print** FH ">habal\n"**,**$seq**->**seq**;**

close **(**FH**);**

**print** "fasta1 done \n"**;**

*#Blast in case of Multitranscripts, and getting the sequence info for all transcripts*

open FILE**,** ">"**,** $ENV**{**'PWD'**}.**'\results\blast1result.fasta'**;**

**if** **(**$seq**->**desc**=~**/transcript variant/**){**

**my** $prog **=** "blastn"**;**

**my** $db **=** "nr"**;**

**my** $e\_val **=** "1e-5"**;**

**my** $remoteBlast **=** **Bio::Tools::Run::**RemoteBlast**->new(-**prog **=>** $prog**,**

**-**data **=>** $db**,**

**-**expect **=>** $e\_val**);**

*# TO DEFINE SPECIFIC ORGANISM: $Bio::Tools::Run::RemoteBlast::HEADER{'ENTREZ\_QUERY'} = 'Homo sapiens [ORGN]';*

**my** $r **=** $remoteBlast**->**submit\_blast**(**$ENV**{**'PWD'**}.**'\results\fasta1result.fasta'**);**

*#Submit blast jobs to ncbi blast queue on sequence(s)*

**while** **(my** @reqIDs **=** $remoteBlast**->**each\_rid **)** **{**

**print** STDERR join**(**" "**,** "\nINFO: RIDs: "**,** @reqIDs**),** "\n"**;**

**foreach** **my** $reqID **(**@reqIDs**)** **{**

*# each search results*

**my** $rc **=** $remoteBlast**->**retrieve\_blast**(**$reqID**);**

**if** **(!** ref **(**$rc**)){**

**if** **(**$rc **<** **0)** **{** *# no match*

$remoteBlast**->**remove\_rid**(**$reqID**);**

**}**

*# Search is not done yet, wait 10 sec, and try to retrieve again*

**print** STDERR "."**;**

sleep **(10);**

**}**

*# got some blast hit*

**else** **{**

**my** $result **=** $rc**->**next\_result**;** *# get the blast output*

**while(my** $hit **=** $result**->**next\_hit**)** **{**

$Desc\_**=0;**

**my** $haball**=**$hit**->**description**();**

**my** $accc **=** $hit**->**accession**();**

**if** **((**$haball**=~**/(^[\w\W]\*, transcript variant)/**)){**

$Desc\_ **=**$1**;}***#&& $accc=~/[A-Z][A-Z]\_/*

**if((**$Desc **eq** $Desc\_**)){**

*# print out the accession etc of all hits*

**print** FILE $hit**->**accession**,**"\n"**;** *#$hit->description, "\t", $hit->name, "\t ", $hit->description, "\t", $hit->significance, "\t", $hit->score, "\t", $hit->frac\_identical, "\n";*

$trackernum **++;**

**}**

**}**

**print** STDERR "\nINFO: removing $reqID\n"**;**

$remoteBlast**->**remove\_rid**(**$reqID**);** *# remove this RID since we*

*# already got the results*

**}**

**}**

**}**

**}**

**else{**

**print** FILE $seq**->**accession\_number**();**

**}**

close FILE**;**

*#To get the number of transcripts*

**my** $file3**=**$ENV**{**'PWD'**}.**'\results\blast1result.fasta'**;**

open FH1**,**$file3**;**

**my** @array**=**<FH1>**;**

**print** @array**,**"\n"**;**

$trans\_num**=**@array**;**

**print** $trans\_num**,**"\n"**;**

close FH1**;**

**if** **(**$trans\_num**>1){**

*#Getting sequence fasta for all transcripts*

open FH1**,** "<"**,** $ENV**{**'PWD'**}.**'\results\blast1result.fasta'**;**

open FH2**,**">"**,**$ENV**{**'PWD'**}.**'\results\fasta2result.fasta'**;**

**print** FH2 ""**;**

**my** $kk**;**

**while** **(**defined **(**$kk**=**<FH1>**)){**

**my** $seq **=** $database**->**get\_Seq\_by\_id**(**$kk**);**

**print** "Seq: "**,** $seq**->**accession\_number**(),** " -- "**,** $seq**->**desc**(),** "\n\n"**;**

**my** $out **=** **Bio::**SeqIO**->**newFh **(** **-**file **=>** ">>"**.**$ENV**{**'PWD'**}.**'\results\fasta2result.fasta'**,** **-**format **=>** 'fasta'**);**

**print** $out $seq**;**

**}**

close **(**FH1**);**

**print** "Multi varriant\n"**;**

**my** $file1**=**$ENV**{**'PWD'**}.**'\results\fasta2result.fasta'**;**

**my**$file2**=**$ENV**{**'PWD'**}.**'\results\clustalwresult'**;**

*#Calling Local ClustalW (needs ClustalW to be installed in ur PC and varriables intialization in PATH and ClustPath)*

**my** $filename **=** $ENV**{**'PWD'**}.**'\results\clustalwresult.aln'**;**

*# Build a clustalw alignment factory*

**my** @params **=** **(**'output' **=>** 'clustal'**,**'TYPE'**=>**'DNA'**,**'outfile'**=>**"$filename"**);**

**my** $factory **=** **Bio::Tools::Run::Alignment::**Clustalw**->new(**@params**);**

*# Pass the factory a list of sequences to be aligned.*

**my** $inputfilename **=** $ENV**{**'PWD'**}.**'\results\fasta2result.fasta'**;**

**my** $aln **=** $factory**->**align**(**$inputfilename**);** *# $aln is a SimpleAlign object.*

**my** $file4**=**$ENV**{**'PWD'**}.**'\results\clustalwresult.aln'**;**

**my** $file5**=**$ENV**{**'PWD'**}.**'\results\consresult.fasta'**;**

*#use cons.exe tool*

system "cons.exe -sequence $file4 -outseq $file5 -identity $trans\_num -setcase 0"**;**

*#analysisng cons.exe results*

**my** $lines**;**

open FH1**,**"<"**,**"$file5"**;**

**while** **(my** $line**=**<FH1>**){**

**if** **(**$line**=~**/^>/**){**

*#Do Nothing "Header"*

**}**

**else{**

chomp $line**;**

$lines**=**$lines**.**$line**;**

**}**

**}**

close FH1**;**

open FH1**,**">"**,**"$file5"**;**

**print** FH1 $lines**;**

close FH1**;**

open FH1**,**"<"**,**"$file5"**;**

open FH2**,**">"**,**$ENV**{**'PWD'**}.**'\results\AlignIO.fasta'**;print** FH2 ""**;** close FH2**;**

open FH2**,**">>"**,**$ENV**{**'PWD'**}.**'\results\AlignIO.fasta'**;**

**while** **(my** $seq**=**<FH1>**){**

**if** **(**$seq**=~**/^>/**){}**

**else{**

**for** **my** $i**(0..(**length**(**$seq**)-19)){**

**my** $substring**=**substr**(**$seq**,**$i**,19);**

**if** **(**$substring**=~**/N/i**){**

*#if not 100% conserved Do notting*

**}**

**else** **{**

*#if 100% conserved switch T => U*

$substring **=~**s/T/U/gi**;print** FH2 $substring**,**"\n" **;**

**}**

**}**

**}**

**}**

close FH1**;**

close FH2**;**

**print** "consresult.pl done!!!\n"**;**

**}**

**else** **{**

**print** "one varriant\n"**;**

**my** $del**=**$ENV**{**'PWD'**}.**'\results\AlignIO.fasta'**;**

system "del $del > temp.txt"**;**

**}**

**my** $file6**=**$ENV**{**'PWD'**}.**'\results\AlignIO\_.fasta'**;**

system "move AlignIO\_.fasta $file6 > temp.txt"**;**

system "copy RNAduplex.exe \.\\RNAxsresult > temp.txt"**;**

system "copy RNAfold.exe \.\\RNAxsresult > temp.txt"**;**

system "copy RNAplfold.exe \.\\RNAxsresult > temp.txt"**;**

system "copy cyggcc\_s-1.dll \.\\RNAxsresult > temp.txt"**;**

system "copy cygwin1.dll \.\\RNAxsresult > temp.txt"**;**

system"perl RNAxs.pl -s $file6 -u $opts{u} -f 0 -g 0 -e 0 -t $opts{t} -a 0 -w 80 -l 40"**;**

**my** $RNAxsresult**=**$ENV**{**'PWD'**}.**'\\'**.**'RNAxsresult'**.**'\\output.csv'**;**

**my** $RNAxsresult\_**=**$ENV**{**'PWD'**}.**'\\'**.**'results'**.**'\\Tar\_acc\_result.txt'**;**

*#Analyzing RNAxs results*

open FH**,**">"**,**"$RNAxsresult\_"**;**

open FH1**,**"<"**,**"$RNAxsresult"**;**

**while** **(my** $line**=**<FH1>**){**

**if** **(**$line**=~**/^([A-Za-z]\*),/**){**

**print** FH ">habal\n"**,**$1**,**"\n"**;**

**}**

**}**

close **(**FH**);**

close **(**FH1**);**

*#Thermo-iscore evaluation*

**my** $Thermo\_iscore **=** $ENV**{**'PWD'**}.**'\iscore\_thermo.xls'**;**

**my** $Excel **=** **Win32::**OLE**->**GetActiveObject**(**'Excel.Application'**)||** **Win32::**OLE**->new(**'Excel.Application'**,** 'Quit'**);** *# use the Excel application if it's open, otherwise open new*

**my** $Book **=** $Excel**->**Workbooks**->**Open**(**$Thermo\_iscore**);** *# open the file*

$Excel**->**Run**(**"SeqInput"**);**

$Excel**->**Run**(**"Secondary\_Structure"**);** *#macro\_name*

$Book**->**Save**;** *#optional - save any changes made by the macro*

$Book**->**Close**;**

**print** "Iscore finished \n"**;**

*#scores filtration*

scoring90 **(**$num**,**$trans\_num**);**

*#snps filtration*

snps**(**$num**);**

*#Off-target filtration*

**if** **(**$off\_target\_filtration**==1){**

off\_target**(**$trans\_num**,**$num**,**$opts**{**d**},**$opts**{**i**},**$opts**{**e**});}**

**if(**$off\_target\_filtration**==0){**

No\_off\_target**(**$trans\_num**,**$num**,**$opts**{**d**},**$opts**{**i**},**$opts**{**e**});}**

*#MysiRNA-Model*

MysiRNA\_Model**(**$num**,**$opts**{**w**},**$opts**{**s**},**$opts**{**d**},**$opts**{**o**},**$opts**{**t**},**$opts**{**u**},**$opts**{**p**});**

**my** $file7**=**$ENV**{**'PWD'**}.**'\\'**.**'results'**.**'\\survey.log'**;**

chomp**(**$acc**);**

chomp**(**$num**);**

**my** $Outfile**=**$num**.**"\_MysiRNA\_Designer\_results\_"**.**$acc**;**

system "del /Q /s $opts{o}\\$Outfile > temp.txt"**;**

system "move survey.log $file7 > temp.txt"**;**

system "Xcopy /e/-Y/I/R .\\results $opts{o}\\$Outfile\\tools\_results > temp.txt"**;**

system "Xcopy /e/-Y/I/R .\\blastresult $opts{o}\\$Outfile\\blastresult > temp.txt"**;**

system "Xcopy /e/-Y/I/R .\\Multi\_score\_Accepted $opts{o}\\$Outfile\\Multi\_score\_Accepted > temp.txt"**;**

system "Xcopy /e/-Y/I/R .\\RNAxsresult $opts{o}\\$Outfile\\RNAxsresult > temp.txt"**;**

system "Xcopy /e/-Y/I/R .\\MysiRNA\_Designer\_Final\_results $opts{o}\\$Outfile\\MysiRNA\_Designer\_Final\_results > temp.txt"**;**

**}**

close Main**;**

**}**

**else{**

usage**();**

**}**

**sub** usage**{**

**print**

"

||||||||||||||||||||||||||||||||||||||||||||||||

|| Welcome To MysiRNA-Designer ||

|| A Software For Rational siRNA Design ||

|| Copyright (C) 2011 <M.Mysara et al> ||

||||||||||||||||||||||||||||||||||||||||||||||||

MysiRNA-Designer version 1, Copyright (C) 2011, M.Mysara et al

MysiRNA-Designer comes with ABSOLUTELY NO WARRANTY.

This is free software, and you are welcome to redistribute it under

certain conditions; type `notepad copying.txt\' for details.;

Use the Following Mandatory Options:\n

-s File containing mRNA(s) Refseq Accession(s) one/line,

<Example, D:\\Acc.txt containing 'NM\_001211' accession number >

-o out put PATH.

<Example, D:\\ >

-w WEKA installation PATH.

<Example, C:\\Weka-3-7 >

Use the Following Non-Mandatory Options:\n

-d To perform off-target filtration, insert mRNA refseq dataset name

without its extension and complete PATH, <Example, D:\\refseq\_rna >.

-t RNAxs threshold on the 8 nts accessibility.

-u RNAxs threshold on the 16 nts accessibility.

-p Use MysiRNA-Model high specificity threshold (Y or N).

-e Updated 3UTR Database, else use default

-i Updated refseq2ensumble ID Database, else use default

For Queries about the installaion, type \'notepad README.txt\'

For Queries about the Copy rights, type \'notepad COPYING.txt\'

"**;**

**}**

**sub** scoring90**{**

*#Local varriable*

**my** $filename **=** $ENV**{**'PWD'**}.**'\Iscore\_thermo.xls'**;**

**my** $trackernum**=0;**

**my** $Tracker**=2;** *# to include siRNA number at the header, START WITH TWO AS THE 1ST TWO siRNA are rejected*

*#////////////////*

*#testing*

open TEST**,** ">>"**,**$ENV**{**'PWD'**}.**'\Multi\_score\_Accepted\\habal.csv'**;**

*#///////////*

open FH1**,** ">"**,**$ENV**{**'PWD'**}.**'\Multi\_score\_Accepted\\'**.**$\_**[0].**"\_Multi\_score\_Approved\_siRNA.fasta"**;**

**print** FH1 ""**;**

close **(**FH1**);**

open FH2**,** ">>"**,**$ENV**{**'PWD'**}.**'\Multi\_score\_Accepted\\'**.**$\_**[0].**"\_Multi\_score\_Approved\_siRNA.fasta"**;**

open FH6**,** ">>"**,**$ENV**{**'PWD'**}.**'\blastresult\\'**.**$\_**[0].**"\_Whole\_siRNA\_off\_targets.csv"**;**

*#............... open the conserved siRNA and read it in array...........*

**my** @cons\_sirna**;**

**my** $sirna\_arr\_length**;**

**if** **(**$\_**[1]>1){**

open FH3**,** "<"**,**$ENV**{**'PWD'**}.**'\results\AlignIO.fasta'**;**

@cons\_sirna**=**<FH3>**;**

$sirna\_arr\_length**=**scalar**(**@cons\_sirna**);**

**print** $sirna\_arr\_length**,**"\t"**;**

close **(**FH3**);**

**}**

**my** $string**;**

*#............... open the target\_Accessibility siRNA and read it in array...........*

open FH4**,** "<"**,**$ENV**{**'PWD'**}.**'\results\Tar\_acc\_result.txt'**;**

**my** @tar\_Acc**=**<FH4>**;**

**my** $tar\_acc\_length**=**scalar**(**@tar\_Acc**);**

**print** $tar\_acc\_length**,**"\t"**;**

close **(**FH4**);**

**my** $tar\_acc\_string**;**

*#..................open thermo-iscore results.......*

**my** $parser **=** **Spreadsheet::**ParseExcel**->new();**

**my** $workbook **=** $parser**->**parse**(**$filename**);**

**my** @Habal**;**

**my** @Habal\_**;**

**if** **(** **!**defined $workbook **)** **{**

die $parser**->**error**(),** ".\n"**;**

**}**

**for** **my** $worksheet **(**$workbook**->**worksheet**(**'Designer'**))** **{**

**my** **(** $row\_min**,** $row\_max **)** **=** $worksheet**->**row\_range**();**

**my** **(** $col\_min**,** $col\_max **)** **=** $worksheet**->**col\_range**();**

**print** "Row min: "**,**$row\_min**,**" "**,**"Row max: "**,**$row\_max**,**" "**,**"Col min: "**,**$col\_min**,**" "**,**"Col max: "**,**$col\_max**,**"\n"**;**

**my** $cell **=** $worksheet**->**get\_cell**(** **0,** **1** **);**

**my** $Maximum**=** $cell**->**value**();**

**my** $cell\_\_ **=** $worksheet**->**get\_cell**(** **15,** **3** **);**

**print** "Value = "**,** $cell\_\_**->**value**(),** "\n"**;**

**my** $counter**=1;**

**my** $i**=0;**

**for** **my** $row **(** **18** **..** $Maximum**-3** **)** **{**

$Tracker**++;** *#add 1 to the current tracker*

**print** TEST $Tracker**.**","**;**

**my** $track**=0;**

**my** $track1**=0;**

*#.................................................................................................*

**my** $celll **=** $worksheet**->**get\_cell**(** $row**,** **1);** *#to get the siRNA sense*

**my** $celll\_AS**=**$worksheet**->**get\_cell**(** $row**,** **2);** *#to get the anitsense*

**my** $sirna**=**$celll**->**value**();**

**my** $sirna\_AS**=**$celll\_AS**->**value**();**

**my** $sirna\_**=**substr**(**$sirna**,0,19);***# to remove the extra 2 nt*

**print** TEST "$sirna\_AS,$sirna\_,"**;**

*#.......................................consider only siRNA conserved by comparing to the @cons\_sirna*

**if** **(**$\_**[1]==1){**

$track**=1;**

**print** TEST "OK\_TRANSCRIPT,"**;**

**}**

**else{**

**for** **my** $numb**(0..(**$sirna\_arr\_length**-1)){**

$string**=**$cons\_sirna**[**$numb**];**

chomp **(**$string**);**

$string**=~**s/T/U/gi**;**

*#print $string,"\t";*

$string**=**substr**(**$string**,0,19);***#no need but to double check*

**if** **(**$sirna\_ **=~** /$string/i**){**

$track**=1;**

**}**

**}**

**}**

*# do the whole scoring else don't do anything*

*#..........................Target access filteration.................................*

**for** **my** $numbb**(0..(**$tar\_acc\_length**-1)){**

$tar\_acc\_string**=**$tar\_Acc**[**$numbb**];**

chomp **(**$tar\_acc\_string**);**

$tar\_acc\_string**=~**s/T/U/gi**;**

*#print $tar\_acc\_string,"\t";*

$tar\_acc\_string**=**substr**(**$tar\_acc\_string**,0,19);***#no need but to double check*

**if** **(**$sirna\_ **=~** /$tar\_acc\_string/i**){**

$track1**=1;**

**print** TEST "ok\_accessibility,"**;**

**}**

**else{**

**print** TEST "false\_accessibility,"**;**

**}**

**}**

*# do the whole scoring else dont do anything*

*#...........ThermoComp filteration.....................................................................................*

**if(**$track**==1&&**$track1**==1){**

**print** TEST "enter,"**;**

*#if (($Thermocomp[$row-20]>.6) && ($Thermocomp[$row-20]<1)){*

**my** $Thermocomp **=** $worksheet**->**get\_cell**(** $row**,** **121);** *#Thermo21*

**my** $Thermocomp\_ **=** $worksheet**->**get\_cell**(** $row**,** **120);** *#Thermo19*

**if** **(**$Thermocomp**->**value**()** **>=0.70){**

**print** TEST "ok\_thermo,"**;**

**my** $iscore**=**$worksheet**->**get\_cell**(**$row**,116);** *#26\*4+nn #iscore*

**if** **(**$iscore**->**value**()** **>=70){**

**print** TEST "ok\_iscore,"**;**

**my** $dG**=** $worksheet**->**get\_cell**(** $row**,** **112);**

**my** $S\_Bio**=** $worksheet**->**get\_cell**(** $row**,** **115);** *#s-Biopredsi*

**if** **(**$S\_Bio**->**value**()** **>=** **0.7){**

**print** TEST "ok\_bio,"**;**

**my** $DISR **=** $worksheet**->**get\_cell**(** $row**,** **119);** *#Disr*

**if** **(**$DISR**->**value**()** **>=** **70){**

**print** TEST "ok\_Dsir,"**;**

**my** $rey **=** $worksheet**->**get\_cell**(** $row**,** **117);***#reynold*

**if** **(**$rey **->**value**()** **>** **1.9** **&&**$rey **->**value**()<** **9.15){**

**print** TEST "ok\_rey,"**;**

**my** $Ui **=** $worksheet**->**get\_cell**(** $row**,** **111);***#Ui-Tie*

**if** **(**$Ui**->**value**()** **=~**/[Ia,Ib,II,III]/**){**

**print** TEST "ok\_ui,"**;**

**my** $Ama **=** $worksheet**->**get\_cell**(** $row**,** **112);***#Amarzguioui*

**if(**$Ama**->**value**()** **>** **-1.21** **&&** $Ama**->**value**()<5.3){**

**print** TEST "ok\_ama,"**;**

**my** $kat **=** $worksheet**->**get\_cell**(** $row**,** **118);***#katoh*

**if** **(**$kat**->**value**()** **>42.03** **&&**$kat**->**value**()** **<** **97.01){**

**print** TEST "ok\_ka,"**;**

**my** $Hs **=** $worksheet**->**get\_cell**(** $row**,** **113);***#Hsieh*

**if** **(**$Hs**->**value**()** **>** **-1.11** **&&** $Hs**->**value**()** **<** **3.11){**

**print** TEST "ok\_hs,"**;**

**my** $Taka **=** $worksheet**->**get\_cell**(** $row**,** **114);***#Takasaki*

**if** **(**$Taka**->**value**()** **>-10.22&&**$Taka**->**value**()** **<14.06){**

*#print "SCORE!!! \n";*

*#Getting Total G*

**my** $G **=** $worksheet**->**get\_cell**(** $row**,** **108);**

**print** FH2 ">XX\_000000 siRNAnum:"**,**$Tracker**,**"\_iscore:"**,**$iscore**->**value**(),**"\_SBiopredsi:"**,**$S\_Bio**->**value**(),**"\_DISR:"**,**$DISR**->**value**(),**"\_reynold:"**,** $rey **->**value**(),**"\_UiTie:"**,**$Ui**->**value**(),**"\_Amarzguioui:"**,**$Ama**->**value**(),**"\_Katoh:"**,**$kat**->**value**(),**"\_Hiesh:"**,**$Hs**->**value**(),**"\_Takasaki:"**,**$Taka**->**value**(),**"\_ThermoComp21:"**,**$Thermocomp**->**value**(),**"\_Thermo19:"**,**$Thermocomp\_**->**value**(),**"\_Gwhole:"**,**$G**->**value**(),**"\_siRNA\_AntiSense:"**,**$sirna\_AS**,**"\n"**,**$sirna\_**,**"\n"**;**

**print** FH6 $Tracker**,**$iscore**->**value**(),**","**,**$S\_Bio**->**value**(),**","**,**$DISR**->**value**(),**","**,**$rey **->**value**(),**","**,**$Ui**->**value**(),**","**,**$Ama**->**value**(),**","**,**$kat**->**value**(),**","**,**$Taka**->**value**(),**","**,**$Thermocomp**->**value**(),**","**,**$Thermocomp\_**->**value**(),**","**,**$G**->**value**(),**","**,**$sirna\_**,**"\n"**;**

**print** TEST "accepted,"**;**

$trackernum**++;**

**}**

**}**

**}**

**}**

**}**

**}**

**}**

**}**

**}**

**next** **unless** $cell**;**

$i**++;**

$counter **++;**

*#................ ...................*

**}**

*# }*

*#.......................................*

**}**

**print** TEST "\n"**;**

**}**

**}**

*#.................................................................*

**print** FH6 '\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_,\_'**;** **print** FH6 "\n"**;**

close FH6**;**

close **(**FH2**);**

**print** "\n SCORING DONE \n"**;**

close TEST**;**

**}**

**sub** snps**{**

**my** $id**;**

open FH**,**$ENV**{**'PWD'**}.**"/results/fasta1result.gb"**;**

**while** **(my** $line**=**<FH>**){**

**if** **(**$line **=~**/^VERSION/**){**

$line**=~**/GI:([0-9]+)/**;***#VERSION NM\_001211.5 GI:168229167*

$id **=** $1**;**

**}**

**}**

**print** $id**,**"\n"**;**

**my** $eutil **=** **Bio::DB::**EUtilities**->new(-**eutil **=>** 'elink'**,**

**-**id **=>** $id**,**

**-**email **=>** 'setyourown@foo.bar'**,**

**-**verbose **=>** **1,**

**-**dbfrom **=>** 'nuccore'**,**

**-**db **=>** 'snp'**,**

**-**cmd **=>** 'neighbor\_history'**,**

**);**

**my** $hist **=** $eutil**->**next\_History **||** die "No history data returned"**;**

$eutil**->**set\_parameters**(-**eutil **=>** 'efetch'**,**

**-**history **=>** $hist**,**

**-**retmode **=>** 'text'**,**

*# 'chr', 'flt', 'brief', 'rsr', 'docset'*

**-**rettype **=>** 'brief'

**);**

$eutil**->**get\_Response**(-**file **=>**$ENV**{**'PWD'**}.**"/results/snps.txt"**,-**format**=>**'fasta'**);**

**my** $counter**=0;**

open FH1**,** $ENV**{**'PWD'**}.**"/results/snps.txt"**;**

open FH2**,**">"**,**$ENV**{**'PWD'**}.**"/results/snps\_region.fasta"**;** **print** FH2 ""**;** close FH2**;**

open FH2**,**">>"**,**$ENV**{**'PWD'**}.**"/results/snps\_region.fasta"**;**

**while** **(my** $line\_**=**<FH1>**){**

$line\_**=~**s/T/U/gi**;**

*#getting all possibilities of SNPs, and rejecting any siRNA with one or more snps*

**if** **(**$line\_**=~**/([A,G,C,U]{18})\[([A,U,C,G,-]\*)\/([A,U,C,G,-]\*)\/([A,U,C,G,-]\*)\/([A,U,C,G,-]\*)\/([A,U,C,G,-]\*)\/([A,U,C,G,-]\*)\]([A,G,C,U]{18})/i**){print** FH2 $1**,**$2**,**$8**,**"\n"**;** **print** FH2 $1**,**$3**,**$8**,**"\n"**;print** FH2 $1**,**$4**,**$8**,**"\n"**;print** FH2 $1**,**$6**,**$8**,**"\n"**;print** FH2 $1**,**$5**,**$8**,**"\n"**;print** FH2 $1**,**$7**,**$8**,**"\n"**;}***#$line\_=~/[A,G,C,U]{18}\[([A,U,C,G,-])\*[\/[A,U,C,G,-]\*]\*\]([A,G,C,U]{18})/i*

**if** **(**$line\_**=~**/([A,G,C,U]{18})\[([A,U,C,G,-]\*)\/([A,U,C,G,-]\*)\/([A,U,C,G,-]\*)\/([A,U,C,G,-]\*)\/([A,U,C,G,-]\*)\]([A,G,C,U]{18})/i**){print** FH2 $1**,**$2**,**$7**,**"\n"**;** **print** FH2 $1**,**$3**,**$7**,**"\n"**;print** FH2 $1**,**$4**,**$7**,**"\n"**;print** FH2 $1**,**$5**,**$7**,**"\n"**;print**$1**,**$6**,**$7**,**"\n"**;}***#$line\_=~/[A,G,C,U]{18}\[([A,U,C,G,-])\*[\/[A,U,C,G,-]\*]\*\]([A,G,C,U]{18})/i*

**if** **(**$line\_**=~**/([A,G,C,U]{18})\[([A,U,C,G,-]\*)\/([A,U,C,G,-]\*)\/([A,U,C,G,-]\*)\/([A,U,C,G,-]\*)\]([A,G,C,U]{18})/i**){print** FH2 $1**,**$2**,**$6**,**"\n"**;** **print** FH2 $1**,**$3**,**$6**,**"\n"**;print** FH2 $1**,**$4**,**$6**,**"\n"**;print** FH2 $1**,**$5**,**$6**,**"\n"**;}***#$line\_=~/[A,G,C,U]{18}\[([A,U,C,G,-])\*[\/[A,U,C,G,-]\*]\*\]([A,G,C,U]{18})/i*

**if** **(**$line\_**=~**/([A,G,C,U]{18})\[([A,U,C,G,-]\*)\/([A,U,C,G,-]\*)\/([A,U,C,G,-]\*)\]([A,G,C,U]{18})/i**){print** FH2 $1**,**$2**,**$5**,**"\n"**;** **print** FH2 $1**,**$3**,**$5**,**"\n"**;print** FH2 $1**,**$4**,**$5**,**"\n"**;}***#$line\_=~/[A,G,C,U]{18}\[([A,U,C,G,-])\*[\/[A,U,C,G,-]\*]\*\]([A,G,C,U]{18})/i*

**if** **(**$line\_**=~**/([A,G,C,U]{18})\[([A,U,C,G,-]\*)\/([A,U,C,G,-]\*)\]([A,G,C,U]{18})/i**){print** FH2 $1**,**$2**,**$4**,**"\n"**;** **print** FH2 $1**,**$3**,**$4**,**"\n"**;}***#$line\_=~/[A,G,C,U]{18}\[([A,U,C,G,-])\*[\/[A,U,C,G,-]\*]\*\]([A,G,C,U]{18})/i*

**}**

close FH2**;**

open FH4**,**">"**,**$ENV**{**'PWD'**}.**"/results/SNPs\_free\_siRNAs.fasta"**;** **print** FH4 ""**;**close FH4**;**

**my** $out\_ **=** **Bio::**SeqIO**->**newFh **(** **-**file **=>** ">"**.**$ENV**{**'PWD'**}.**"/results/SNPs\_free\_siRNAs.fasta"**,** **-**format **=>** 'fasta'**);**

**my** $quit**=0;**

**my** $seq**;**

**my** $str **=** **Bio::**SeqIO**->new(-**file**=>**$ENV**{**'PWD'**}.**'\Multi\_score\_Accepted\\'**.**$\_**[0].**'\_Multi\_score\_Approved\_siRNA.fasta' **,** **-**format **=>** 'fasta' **);**

**while** **(my** $input **=** $str**->**next\_seq**()){**

$quit**=0;**

$seq**=**$input**->**seq**;**

open FH3**,**$ENV**{**'PWD'**}.**"/results/snps\_region.fasta"**;**

**while** **(my** $line1**=**<FH3>**){**

**last** **if** $quit**==1;**

**if** **(**$line1**=~**/$seq/**){**

$quit**=1;**

**}**

**}**

**if** **(**$quit**==0){**

**print** $out\_ $input**;**

**print** $seq**,**"\n"**;**

**}**

**}**

**}**

**sub** off\_target**{**

*#....making the result folder*

**my** $num**=1;**

**my** $quit**=0;**

**my** $Tracer**;**

**my** $flage**;**

**my** $blast\_db**=**$\_**[2];**

*#////////////get the current path*

**my** $filename**=**$ENV**{**'PWD'**}.**'\blastresult'**.**'\\'**;**

**my** $filename1**=**$ENV**{**'PWD'**}.**'\results\Blastquery.fasta'**;**

*#..........geting the @ of all input acc*

open FH**,**"<"**,**$ENV**{**'PWD'**}.**'\results\blast1result.fasta'**;**

**my** $t\_acc**;**

**if** **(**$\_**[0]>1){**

**while** **(my**$s\_acc**=**<FH>**){**

**my** $NN**=**chomp**(**$s\_acc**);**$t\_acc**=**$t\_acc**.**$s\_acc**;**

**}**

**}**

*#..........to ensure that Acc of the mRNA and its transcript is rejected*

**else{**

**my** $str\_ **=** **Bio::**SeqIO**->new(-**file**=>**$ENV**{**'PWD'**}.**'\results\fasta1result.fasta' **,** **-**format **=>** 'fasta' **);**

**while** **(my** $input\_ **=** $str\_**->**next\_seq**()){**$t\_acc**=**$input\_**->**id**;** **print** "$t\_acc\n"**;}**

**}**

*#...........to check if the snps worked and produced the snps\_score\_final.fasta or to shift to iscore\_final.fasta*

**my** $checkfile**=**$ENV**{**'PWD'**}.**'\results\SNPs\_free\_siRNAs.fasta'**;**

**my** $checkfile\_**=**$ENV**{**'PWD'**}.**'\Multi\_score\_Accepted\\'**.**$\_**[1].**"\_Multi\_score\_Approved\_siRNA.fasta"**;**

**my** $str**;**

**if** **(-**e "$checkfile"**){**

$str **=** **Bio::**SeqIO**->new(-**file**=>**"$checkfile" **,** **-**format **=>** 'fasta' **);**

**}**

**else{**

$str **=** **Bio::**SeqIO**->new(-**file**=>**"$checkfile\_" **,** **-**format **=>** 'fasta' **);**

**}**

**while** **(my** $input **=** $str**->**next\_seq**()){**

$Tracer**=1;**

*#.......... reseting*

system"del Temp.fasta > temp.txt"**;**

*#...............getting the sirna seq and 2-7 seq*

**my** $queryseq**=**$input**->**seq**();**

**my** $seed**=**$input**->**subseq**(2,7);**

*#......................*

**print** $queryseq**,**"\n"**;**

open Blastquery**,** ">"**,** $ENV**{**'PWD'**}.**'\results\Blastquery.fasta'**;**

**print** Blastquery ">$t\_acc Homo sapiens, mRNA.\n"**,**$queryseq**;**

close Blastquery**;**

system "blastall.exe -p blastn -d $blast\_db -i $filename1 -o Temp.fasta -W 7 -e 1000 -q -1 -G 1 -E 2 -I T"**;**

**print** "blastall done\n"**;**

*#...................reading the result file*

**my** $report\_obj **=** **new** **Bio::**SearchIO**(-**format **=>** 'blast'**,**

**-**file **=>** "Temp.fasta"**);**

**while(** **my** $result **=** $report\_obj**->**next\_result **){**

**while(** **my** $hit **=** $result**->**next\_hit **){**

*#if the hit with one of the gene transcripts, Do nothing*

**my** $hit\_acc**=**$hit**->**accession**;**

**if** **(**$t\_acc**=~**/$hit\_acc/**){}**

**else{**

**if** **(**$hit**->**description**()** **=~**/^Homo sapiens/ **){**

*#Homosapiens only species*

**print** $hit**->**accession**().**" "**.**$result**->**query\_accession**(),**"\n"**;**

**while(** **my** $hsp **=** $hit**->**next\_hsp **){**

*# searching the hists*

**if** **(** **(**$hsp**->**length**(**'query'**)==19** **&&** $hsp**->**percent\_identity **>** **93** **)||(**$hsp**->**length**(**'query'**)==18** **&&** $hsp**->**percent\_identity **==100))** **{** **print** "1st stage filteration\n"**;** $Tracer**=0;}**

**else{**

**my** @filter**=**$hsp**->**seq\_inds**(**'query'**,**'identical'**);**

*#to convert array to string*

**my** $String**=** join**(**''**,**@filter**);**

**my** @hit\_range**=**$hsp**->**range**(**'hit'**);**

**my** $hit\_range\_String**=**join**(**''**,**@hit\_range**);**

**if** **(** $String**=~**/([0-9]\*)245678([0-9]\*)/**){**

**print** "Seed matching..check if it matches 3UTR \(true -ve\).. or not \(false -ve\)!!!! \n"**;**

**my** $hit\_acc**=**$hit**->**accession**();**

*#......................*

open FHH**,**$\_**[3];**

**my** $enst**;**

**my** $hitid **=** $hit**->**accession**();**

*#to make sure no accession versions are included*

**if(**$hitid**=~**/([\w]\*)\./**){**

**print** $1**,**"\n"**;**

$hitid**=**$1**;**

**}**

**while** **(my** $id**=**<FHH>**){**

**if** **(**$id**=~**/^([\w]\*),$hitid/**){**

$enst**=**$1**;***#to take the ensumbl id corrosponding to refseq accession*

**print**"\.\.\.\. $enst \.\.\.\.\."**;**

**}**

**}**

close FHH**;**

**if(!**$enst**){***#ID still couldn't be found, reject the siRNA*

$Tracer**=0;**

**print** "siRNA's off-target id couldn't be found, rejecting the siRNA\n"**;**

**}**

**else{**

**my** $in **=** **Bio::**SeqIO**->new(-**file **=>** $\_**[4],**

**-**format **=>** 'fasta'**);**

$flage**=0;***#to know when to use ensembl id to get 3UTR in case we couldn't find in lacal database*

**while** **(** **my** $seq **=** $in**->**next\_seq**()** **){**

**if** **(**$seq**->**id**=~**/$enst/**){**

$flage**=1;**

**print** $seed**,**"\n"**;**

**if** **(**$seq**->**seq**=~**/$seed/**){**

**print** "True -ve \n"**;**

$Tracer**=0;**

**}**

**else{**

**print** "False -ve \n"**;**

**}**

**}**

**}**

**}**

*#..........................*

**if** **(**$flage**==0){**

*##Cant find ENSG ID in the ENSG\_UTR database, reject the siRNA*

$Tracer**=0;**

**print** "True -ve \n"**;**

**}**

*#............................*

**}**

**}**

**last** **if** $Tracer **==** **0;**

**}**

**last** **if** $Tracer **==** **0;**

**}**

**last** **if** $Tracer **==** **0;**

**}**

**last** **if** $Tracer **==** **0;**

**}**

**last** **if** $Tracer **==** **0;**

**}**

**if(**$Tracer**==1){**

**my** $out\_ **=** **Bio::**SeqIO**->**newFh **(** **-**file **=>** ">>"**.**$ENV**{**'PWD'**}.**"/MysiRNA\_Designer\_Final\_results/"**.**$\_**[1].**"\_Final\_siRNA.fasta"**,** **-**format **=>** 'fasta'**);**

**print** $out\_ $input**;**

*#open FH6,">>", $ENV{'PWD'}.'\blastresult\Final.fasta';*

*#print FH6 $queryseq,"\n";*

**my** $off\_target**=**$filename**.**"siRNA\_num\_"**.**$num**.**"\_Off\_Target\.fasta"**;**

system "copy Temp.fasta $off\_target > temp.txt"**;**

**}**

$num **++;**

**}**

**print** "off\_target done!!! \n !"**;**

**}**

**sub** No\_off\_target**{**

*#....making the result folder*

**my** $num**=1;**

**my** $quit**=0;**

**my** $Tracer**;**

**my** $flage**;**

**my** $blast\_db**=**$\_**[2];**

*#...............get the current path*

**my** $filename**=**$ENV**{**'PWD'**}.**'\blastresult'**.**'\\'**;**

**my** $filename1**=**$ENV**{**'PWD'**}.**'\results\Blastquery.fasta'**;**

*#.............. geting the @ of all input acc*

open FH**,**"<"**,**$ENV**{**'PWD'**}.**'\results\blast1result.fasta'**;**

**my** $t\_acc**;**

**if** **(**$\_**[0]>1){**

**while** **(my**$s\_acc**=**<FH>**){**

**my** $NN**=**chomp**(**$s\_acc**);**$t\_acc**=**$t\_acc**.**$s\_acc**;**

**}**

**}**

*#................to ensure that Acc of the mRNA and its transcript is rejected*

**else{**

**my** $str\_ **=** **Bio::**SeqIO**->new(-**file**=>**$ENV**{**'PWD'**}.**'\results\fasta1result.fasta' **,** **-**format **=>** 'fasta' **);**

**while** **(my** $input\_ **=** $str\_**->**next\_seq**()){**$t\_acc**=**$input\_**->**id**;** **print** "$t\_acc\n"**;}**

**}**

*#...................to check if the snps worked and produced the snps\_score\_final.fasta or to shift to iscore\_final.fasta*

**my** $checkfile**=**$ENV**{**'PWD'**}.**'\results\SNPs\_free\_siRNAs.fasta'**;**

**my** $checkfile\_**=**$ENV**{**'PWD'**}.**'\Multi\_score\_Accepted\\'**.**$\_**[1].**"\_Multi\_score\_Approved\_siRNA.fasta"**;**

**my** $str**;**

**if** **(-**e "$checkfile"**){**

$str **=** **Bio::**SeqIO**->new(-**file**=>**"$checkfile" **,** **-**format **=>** 'fasta' **);**

**}**

**else{**

$str **=** **Bio::**SeqIO**->new(-**file**=>**"$checkfile\_" **,** **-**format **=>** 'fasta' **);**

**}**

**while** **(my** $input **=** $str**->**next\_seq**()){**

**{**

**my** $out\_ **=** **Bio::**SeqIO**->**newFh **(** **-**file **=>** ">>"**.**$ENV**{**'PWD'**}.**"/MysiRNA\_Designer\_Final\_results/"**.**$\_**[1].**"\_Final\_siRNA.fasta"**,** **-**format **=>** 'fasta'**);**

**print** $out\_ $input**;**

**my** $off\_target**=**$filename**.**"siRNA\_num\_"**.**$num**.**"\_Off\_Target\.fasta"**;**

system "copy Temp.fasta $off\_target > temp.txt"**;**

**}**

$num **++;**

**}**

**print** "off\_target Skipped!!! \n !"**;**

**}**

**sub** MysiRNA\_Model**{**

open FH1**,**">>"**,**$ENV**{**'PWD'**}.**'\MysiRNA\_Designer\_Final\_results\\'**.**$\_**[0].**'\_Final\_MysiRNA.fasta'**;**

open FH2**,** $ENV**{**'PWD'**}.**'\results\blast1result.fasta'**;**

**my** $ID\_total**=**""**;**

**while** **(my** $ID**=**<FH2>**){**

chomp $ID**;**

$ID\_total**=**$ID\_total**.**$ID**.**"\n"**;**

**}**

**print** FH1 "

||||||||||||||||||||||||||||||||||||||||||||||||

|| MysiRNA-Designer Results ||

|| A Software For Rational siRNA Design ||

|| Copyright (C) 2011 <M.Mysara et al> ||

||||||||||||||||||||||||||||||||||||||||||||||||

These siRNA designed to target mRNA(s) with the following Accession:

$ID\_total

Using the following runtime parameters:

mRNA refseq database PATH: $\_[3]

mRNA input accession number PATH: $\_[2]

Output PATH: $\_[4]

WEKA installation PATH: $\_[1]

RNAxs threshold on the 8 nts accessibility: $\_[5]

RNAxs threshold on the 16 nts accessibility: $\_[6]

Use MysiRNA-Model High specificity Threshold: $\_[7]

"**;**

**my** $inputfile**=**$ENV**{**'PWD'**}.**'\MysiRNA\_Designer\_Final\_results\\'**.**$\_**[0].**'\_Final\_siRNA.fasta'**;**

**my** $test**=**$ENV**{**'PWD'**}.**'\results\test.arff'**;**

**my** $str **=** **Bio::**SeqIO**->new(-**file**=>**$inputfile**,-**format **=>** 'fasta' **);**

**my** $Thermo**=0;**

**my** $iscore**=0;**

**my** $DSIR**=0;**

**my** $DG**=0;**

**my** $PATH\_**=**$ENV**{**'PWD'**};**

**while** **(my** $input **=** $str**->**next\_seq**()){**

open FH**,**">"**,**$test**;**

**my** $header**=** $input**->**desc**;**

**my** $seq**=**$input**->**seq**;**

**if** **(**$header**=~**/\_iscore:([0-9]\*[\.]?[0-9]\*)\_/**){**

$iscore**=**$1**;**

**}**

**if** **(**$header**=~**/\_Gwhole:(-[0-9]\*[\.]?[0-9]\*)\_/**){**

$DG**=**$1**;**

**}**

**if** **(**$header**=~**/\_ThermoComp21:([0-9]\*[\.]?[0-9]\*)\_/**){**

$Thermo**=**$1**;**

**}**

**print** FH "\@relation sirnahabal\n\n\@attribute ThermoComposition21 numeric\n\@attribute i-Score numeric\n\@attribute Whole-DG numeric\n\@attribute ' Inhibition' numeric\n\n\@data\n$Thermo,$iscore,$DG,$DSIR\n"**;**

close FH**;**

*#.................*

**my** $model**=**$ENV**{**'PWD'**}.**'\MysiRNA\_Model.model'**;**

**my** $result**=**$ENV**{**'PWD'**}.**'\results\Weka\_result.txt'**;**

chdir "$\_[1]"**;**

system "java weka.classifiers.functions.MultilayerPerceptron -l $model -T $test -p 0 > $result"**;**

chdir "$PATH\_"**;**

*#................*

open FH2**,**$result**;**

**my** @mysirna**=**<FH2>**;**

$mysirna**[5]=~**/ 1 0 ([0-9]\*\.?[0-9]\*) [0-9]\*\.?[0-9]\* /**;**

**my** $MysiRNA\_Score**=**$1**;**

*# $header=~/(\>[\w\W]\*)siRNA\_AntiSense\:([A-Za-z]\*)/;*

$header**=~**s/\_siRNA\_AntiSense:/\_MysiRNA:$MysiRNA\_Score\nAntiSense: /gi**;**

**if** **(**$\_**[7]=~**/N/**){**

**print** FH1 ">"**,**$header**,**"\nSense: $seq\n"**;**

**}**

**if** **(**$\_**[7]=~**/Y/**){**

**if** **(**$MysiRNA\_Score**>=93){**

**print** FH1 ">"**,**$header**,**"\nSense: $seq\n"**;**

**}**

**}**

close FH2**;**

**}**

close FH1**;**

**print** "MysiRNA\_Model DONE!!!!\n"**;**

**}**