MOLECULAR EPIDEMIOLOGICAL STUDY ON INFECTIOUS PANCREATIC NECROSIS VIRUS ISOLATES FROM AQUAFARMS IN SCOTLAND OVER THREE DECADES

Kristina Ulrich

UNIVERSITY of STIRLING

Thesis submitted for the degree of Doctor of Philosophy

Faculty of Natural Sciences

Institute of Aquaculture

APPENDIX

Table of Contents

V Appendix - Scripts

V.1 MiSeq data process

V.1.1 MiSeq reads process

```
#PATHES
PROGRAMS=/home/bioinf/stefanie/programs/
IPNV=/home/bioinf/stefanie/databases/IPNV/


#Uncompress files
for F in SRC/*gz; do gunzip $F; done
for F in SRC/*fqz; do  G=${F%.*}; fqz_comp -d $F > $G; done

#Quality check with fastqc
for F in SRC/*fastq; do fastqc $F -o fastqc; done

#Filter with PrinSeq
for F in SRC/*R1*fastq; do G=${F/R1/R2}; H=${F##*/}; perl $PROGRAMS/prinseq-lite-
0.20.4/prinseq-lite.pl -min_qual_score 10 -min_qual_mean 20 -ns_max_n 0 -noniupac -derep 14 -
lc_method dust -lc_threshold 30 -verbose -fastq $F -fastq2 $G -out_good raw/$H -out_bad null -
log prinseq-lite.log -graph_data prinseq-lite.gd; done &>> raw/prinseq.log

#Trim with Trimmomatic
for F in raw/*1.fastq; do G=${F/_1.fastq/_2.fastq}; H=${F#*/}; I=${G#*/}; nice java -jar
$PROGRAMS/Trimmomatic-0.35/trimmomatic-0.35.jar PE -threads 15 -phred33 $F $G trimmomatic/$H
trimmomatic/$H.unpaired.fastq trimmomatic/$I
trimmomatic/$I.unpaired.fastq ILLUMINACLIP:$PROGRAMS/adapters.fasta:2:40:15; done

#Remove contamination with deconseq
for F in trimmomatic/*.fastq; do G=${F#*/}; H=${G%%.*}; mkdir decon/tmp; cd decon/tmp; split -
l 40000 ../../$F; parallel 'perl $PROGRAMS/deconseq-standalone-0.4.3/deconseq.pl -f {} -dbs
phix,salmon,trout -dbs_retain viruses -i 90 -out_dir . -keep_tmp_files -group 1 -id {}' ::::
x*; cat *clean.fq > ../$G.clean.fq; cat *cont.fq > ../$G.cont.fq; for X in *tsv; do echo $X;
Y=${X#*_} ; echo $Y; cat *$Y > ../all.$Y ; done ; cd -;  rm -r decon/tmp; done

#Reduce coverage with khmer
for F in ../decon/*_1*lean.fq; do G=${F/_1/_2}; $PROGRAMS/khmerEnv/bin/normalize-by-
median.py -C 20 -k 20 -N 4 -x 4e8 $F $G; done &> log
cd ..

#Denovo assembly with spades
for F in reduce_coverage/*1.fastq.clean*fq.keep.fastq; do
G=${F/1.fastq.clean/2.fastq.clean}; H=${F#*/}; I=${H%%.*}; $PROGRAMS/SPAdes-3.7.1-
Linux/bin/spades.py -1 $F -2 $G -o spades/$I -t 8; done &> spades_reduced_coverage.log

#reference based assembly with tanoti

for F in ../SRC/*R1*.fastq; do G=${F/R1/R2}; echo $G; H=${F##*/}; echo $H; I=${H%.*}; echo $I;
/home/bioinf/stefanie/programs/Tanoti-1.2-Linux/tanoti -r $IPNV/all.c.fa   -i $F $G -o $I.sam
-P 8 -u 1; done > tanoti.log
```

```
for F in *.sam; do echo $F; samtools view -bS $F > ${F/sam/bam}; done
for F in *c.bam; do echo $F; samtools sort $F -o  $F.sorted; done
for F in *.bam.sorted; do echo $F; samtools index $F;done

cp ../../databases/IPNV/all.c.fa ipnv.fa
for F in *.bam.sorted; do H=${F##*/}; I=${H%%.*}; samtools mpileup  -uf ipnv.c.fa  $F |
$PROGRAMS/bcftools/bcftools call -c | vcfutils.pl vcf2fq > $I.cons.fq; done
for F in *cons.fq; do seqret $F -out ${F/.fq/.fa}; done
for F in *cons.fa; do awk '/>/{print ID"\n"SEQ; ID=$0; SEQ="";} !/>/{SEQ=SEQ""$0}END{print
ID"\n"SEQ} ' $F > $F.oneline; done
for F in *cons.fa.oneline; do G=${F/_L001_R1_001.cons.fa.oneline/};  awk
'/>/{split($1,a,">");FN=a[2]; print "echo \">"G"\" >> "FN"_all.fa"}!/>/{print "echo \""$1"\"
>> "FN"_all.fa"}' G=$G $F; done  | sh


#reference based assembly with stampy
mkdir stampy
cd stampy
$PROGRAMS/stampy-1.0.28/stampy.py -G ipnv $IPNV/all.fa
$PROGRAMS/stampy-1.0.28/stampy.py -g ipnv -H ipnv
for F in ../SRC/*R1*fastq; do G=${F/R1/R2}; H=${F##*/}; I=${H%%_*};  $PROGRAMS /stampy-
1.0.28/stampy.py -g ipnv -h ipnv -M $F $G > $I.sam ; done

for F in *.sam; do samtools view -bS $F > ${F/sam/bam}; done
for F in *bam; do samtools sort $F -o $F.sorted; done
for F in *sorted; do samtools index $F;done

cp ../../databases/IPNV/all.c.fa ipnv.fa
for F in *.bam.sorted; do H=${F##*/}; I=${H%%.*}; samtools mpileup  -uf ipnv.c.fa  $F |
$PROGRAMS/bcftools/bcftools call -c | vcfutils.pl vcf2fq > $I.cons.fq; done
for F in *cons.fq; do seqret $F -out ${F/.fq/.fa}; done
for F in *cons.fa; do awk '/>/{print ID"\n"SEQ; ID=$0; SEQ="";} !/>/{SEQ=SEQ""$0}END{print
ID"\n"SEQ} ' $F > $F.oneline; done
for F in *cons.fa.oneline; do G=${F/_L001_R1_001.cons.fa.oneline/};  awk
'/>/{split($1,a,">");FN=a[2]; print "echo \">"G"\" >> "FN"_all.fa"}!/>/{print "echo \""$1"\"
>> "FN"_all.fa"}' G=$G $F; done  | sh
cd ..
```

V.1.2 Workflow for coverage plots generation

```
#sequences assembled by spades
#segment A
#/Volumes/Experiments/stefanie/viruses/ends/eval_1/methods/spades/ref_all_NC_001915.fa.oneline
.formatted
#segment B
#/Volumes/Experiments/stefanie/viruses/ends/eval_1/methods/spades/ref_all_NC_001916.fa.oneline
.formatted
#create fasta index
for F in *fa; do samtools faidx $F; done
#bowtie2 index building
for F in *fa; do bowtie2-build $F ${F/.fa}; done
#get reads
cp /home/bioinf/stefanie/IPNV/RAW/*/*fqz .
#decompress reads
for F in *fqz;  do fqz_comp -d $F ${F/.fqz/}; done
#mapping
for F in *fa; do G=${F/.fa/}; bowtie2 -1 ${F/.fa/_L001_R1_001.fastq} -2
${F/.fa/_L001_R2_001.fastq} -x $G -S $G.sam -p 20 &> $G.log; done
#sam2bam
for F in *.sam; do samtools view -bS $F > ${F/sam/bam}; done
#sort bam
for F in *bam; do samtools sort $F -o $F.sorted; done
#bedgraph
for F in *bam.sorted; do G=${F#*/}; H=${G%%.*}; genomeCoverageBed -ibam $F -g $H.fa.fai -bga >
$F.with_zeroes.bedgraph; done
#generate plots
bash coverage.sh
#blast
for F in *fa; do blastn -db $IPNV/all.fna -query $F -outfmt 6 > $F.bl; done

######Coverage.sh###
for F in *bedgraph; do awk '/segment_A/{if($3-$2==1){print $1"\t"$2"\t"$4} if($3-
$2>1){for(i=$2;i<$3; i++){print $1"\t"i"\t"$4}} }' $F | awk '{print $2"\t"$3}' >
$F.segment_A.coverage.txt; done
for F in *bedgraph; do awk '/segment_B/{if($3-$2==1){print $1"\t"$2"\t"$4} if($3-
$2>1){for(i=$2;i<$3; i++){print $1"\t"i"\t"$4}} }' $F | awk '{print $2"\t"$3}' >
$F.segment_B.coverage.txt; done
for F in  *sorted.with_zeroes.bedgraph.segment_A.coverage.txt; do G=${F/.coverage.txt};
H=${G##*.}; Rscript coverage.c.R $F ${F/.txt/.pdf} $H pdf; done
for F in  *sorted.with_zeroes.bedgraph.segment_B.coverage.txt; do G=${F/.coverage.txt};
H=${G##*.};  Rscript coverage2.c.R $F ${F/.txt/.pdf} $H pdf; done
for F in *pdf; do  mv $F ${F/.bam.sorted.with_zeroes.bedgraph./_}; done
for F in *coverage.pdf; do mv $F ${F/.coverage/_coverage}; done

####coverage.c.R#######
library(ggplot2)
library(scales)
library[1]
library(gridExtra)
# Filenames
args <- commandArgs(TRUE)
input <- args[1];
```

```
output <- args[2];
type <- args[3];
format <- args[4];
maxk <- 0;

roundUp <- function(x,to=10)
{
    to*(x%/%to + as.logical(x%%to))
}
colours = c("#68B5B9", "#CF746D", "#91A851");
colour = ("#68B5B9");
if (format=="png") {
    textsize <- c(40)
    pointsize <- c(5)
    pointalpha <- c(0.5)
    pointshape <- c(1)
    pointwidth <- c(3)
    xvjust <- c(1.2)
    yvjust <- c(1.8)
} else {
    textsize <- c(14)
    pointsize <- c(2)
    pointalpha <-c(0.4)
    pointshape <- c(1)
    pointwidth <- c(1)
    xvjust <- c(0.2)
    yvjust <- c(0.8)
}
     colourcode = colour;
# Plot coverage vs position
data_coverage = read.table(input, col.name=c("Position", "Coverage"))
if (format=="png") {
    png(output, width=1600, height=400)
    print(ggplot(data_coverage, aes(x=data_coverage$Position, y=data_coverage$Coverage)) +
geom_line(color=colourcode) + ggtitle(type) + theme(text = element_text(size=textsize)) +
xlab("Position") + ylab("Mean coverage") + expand_limits(y = 0) + theme(plot.margin =
unit(c(0.02,0.02,0.04,0.02), "npc")) + theme(axis.title.x=element_text(vjust=-xvjust)) +
theme(axis.title.y=element_text(vjust=yvjust)))
        } else {
                    # pdf_coverage <- paste(graphsdir, "/", refid, "/", refid,
"_",type,"_coverage.pdf", sep="");
                     pdf(output, width=16, height=4)
                     print(ggplot(data_coverage, aes(x=data_coverage$Position,
y=data_coverage$Coverage)) + geom_line(color=colourcode) + ggtitle(type) + theme(text =
element_text(size=textsize)) + xlab("Position") + ylab("Mean coverage") + coord_cartesian(xlim
= c(0, 3097)) +  scale_x_continuous(expand = c(0, 0)) + scale_y_continuous(expand = c(0,
0))  + theme(plot.margin = unit(c(0.02,0.02,0.04,0.02), "npc")) +
theme(axis.title.y=element_text(vjust=0.2)) + theme(axis.title.x=element_text(vjust=-0.2)) +
expand_limits(y = 0) + theme(plot.margin = unit(c(0.02,0.02,0.04,0.02), "npc")) +
theme(axis.title.x=element_text(vjust=-xvjust)) +
theme(axis.title.y=element_text(vjust=yvjust)) )
}
garbage <- dev.off()
```

V.2 MinION data process

```
###poretools###
for F in SRC/*; do G=${F#*/}; poretools  fastq $F/*fast5 --type 2D > $G.passed.fastq; done
for F in SRC/*; do G=${F#*/}; poretools  fasta $F/*fast5 --type 2D > $G.passed.fasta; done
for F in SRC/*; do G=${F#*/}; poretools  stats $F/*fast5 --type 2D > $G.stats.txt; done
for F in SRC/*; do G=${F#*/}; poretools winner $F/*fast5 > $G.winner.fasta; done

###nanook###
mkdir nanook
cd nanook
export NANOOK_DIR=/home/bioinf/stefanie/programs/NanoOK/
for F in /data/minion/run7_20161207/downloads/pass/*; do G=${F##*/}; mkdir $G; done
for F in /data/minion/run7_20161207/downloads/pass/*; do G=${F##*/}; mkdir $G/fast5; done
for F in /data/minion/run7_20161207/downloads/pass/*; do G=${F##*/}; mkdir $G/fast5/pass; done
for F in /data/minion/run7_20161207/downloads/pass/*; do G=${F##*/};mkdir $G/fast5/fail; done
for F in ../SRC/*; do G=${F##*/};  cp $F/*fast5 $G/fast5/pass; done
for  F in *; do G=${F##*/}; /home/bioinf/stefanie/programs/NanoOK/bin/nanook extract -s $F;
done
rm -r BC*
cp ../../metrichor.run3/ref_based/Noda/FJ789783_FJ789784.fa .
lastdb -Q 0 FJ789783_FJ789784 FJ789783_FJ789784.fa
for F in NB*; do  /home/bioinf/stefanie/programs/NanoOK/bin/nanook align -s $F -r
FJ789783_FJ789784.fa; done
for F in NB*; do /home/bioinf/stefanie/programs/NanoOK/bin/nanook analyse -s $F -r
FJ789783_FJ789784.fa -passonly ; done
for F in NB*; do /home/bioinf/stefanie/programs/NanoOK/bin/nanook analyse -s $F -r
FJ789783_FJ789784.fa -passonly -2donly ; done


###bwa###
bwa index FJ789783_FJ789784.fa
for F in ../*.fastq; do echo $F;  bwa mem -x ont2d FJ789783_FJ789784.fa $F | samtools view -T
FJ789783_FJ789784.fa -bS - | samtools sort -T $F.FJ.bwa -o $F.FJ.bwa.bam -; done
for F in NB*passed**FJ.bwa.bam; do echo $F; samtools mpileup  -uf FJ789783_FJ789784.fa $F |
/home/bioinf/stefanie/programs/bcftools/bcftools call -c | vcfutils.pl vcf2fq > $F.cons.fq;
done
for F in NB*passed**FJ.bwa.bam.cons.fq; do echo $F; seqret $F -out ${F/cons.fq/cons.fa}; done
```

V.3 Genome annotation

```
#!/usr/bin/perl

#BEGIN{foreach (@INC) {s/\/usr\/local\/packages/\/local\/platform/}};
#use lib (@INC,$ENV{"PERL_MOD_DIR"});
#no lib "$ENV{PERL_MOD_DIR}/i686-linux";
#no lib ".";

=head1 NAME

split_multifasta.pl - split a single FASTA file containing multiple sequences into separate
files.

=head1 SYNOPSIS

USAGE: split_multifasta.pl
            --input_file=/path/to/some_file.fsa
            --output_dir=/path/to/somedir
          [ --output_list=/path/to/somefile.list
            --output_subdir_size=1000
            --output_subdir_prefix=fasta
            --seqs_per_file=1
            --compress_output=1
          ]


 split_multifasta.pl --in snapdmel.aa --output_dir=./ --f=snaa --seqs_per_file=1000

=head1 OPTIONS

B<--input_file,-i>
    The input multi-fasta file to split.

B<--output_dir,-o>
    The directory to which the output files will be written.

B<--output_list,-s>
    Write a list file containing the paths of each of the regular output files.  This may be
useful
    for later scripts that can accept a list as input.

B<--output_file_prefix,-f>
    If defined, each file created will have this string prepended to its name.  This is
ignored unless
    writing multiple sequences to each output file using the --seqs_per_file option with a
value greater
    than 1, else each file created will just be a number.

B<--output_subdir_size,-u>
    If defined, this script will create numbered subdirectories in the output directory, each
    containing this many sequences files.  Once this limit is reached, another subdirectory
    is created.
```

B<--output_subdir_prefix,-p>
    To be used along with --output_subdir_size, this allows more control of the names of the
    subdirectories created. Rather than just incrementing numbers (like 10), each
subdirectory
    will be named with this prefix (like prefix10).

B<--compress_output,-c>
    Output fasta files will be gzipped when written.

B<--debug,-d>
    Debug level. Use a large number to turn on verbose debugging.

B<--log,-l>
    Log file

B<--help,-h>
    This help message

=head1 DESCRIPTION

This script is used to split a single FASTA file containing multiple sequences into separate files containing one sequence each.

=head1 INPUT

The input is defined with --input_file and should be a single fasta file. File extensions are ignored. When creating this multi-entry FASTA file, one should take care to make the first *word* after the > symbol a unique value, as it will be used as the file name for that sequence.
For example:

```
    >gi53791237 Tragulus javanicus p97bcnt gene for p97Bcnt
    ACAGGAGAAGAGACTGAAGAGACACGTTCAGGAGAAGAGCAAGAGAAGCCTAAAGAAATGCAAGAAGTTA
    AACTCACCAAATCACTTGTTGAAGAAGTCAGGTAACATGACATTCACAAACTTCAAAACTAGTTCTTTAA
    AAAGGAACATCTCTCTTTTAATATGTATGCATTATTAATTTATTTACTCATTGGCGTGGAGGAGGAAATG

    >gi15387669 Corynebacterium callunae pCC1 plasmid
    ATGCATGCTAGTGTGGTGAGTATGAGCACACACATTCATGGGCACCGCCGGGGTGCAGGGGGGCTTGCCC
    CTTGTCCATGCGGGGTGTGGGGCTTGCCCCGCCGATAGAGACCGGCCACCACCATGGCACCCGGTCGCGG
    GGTGATCGGCCACCACCACCGCCCCCGGCCACTCTCCCCCTGTCTAGGCCATATTTCAGGCCGTCCACTG
```

Whitespace is ignored within the input file. See the OUTPUT section for more on creation of output files.

=head1 OUTPUT

The name of each output sequence file is pulled from the FASTA header of that sequence. The first *word* after the > symbol will be used as the file name, along with the extension .fsa. The word is defined as all the text after the > symbol up to the first whitespace.

If the above example were your input file, two files would be created:

```
    gi53791237.fa
    gi15387669.fa
```

Any characters other than a-z A-Z 0-9 . _ - in the ID will be changed into an underscore. This only occurs in the file name; the original FASTA header within the file will be unmodified.

You can pass a path to the optional --output_list to create a text file containing the full paths
to each of the FASTA files created by this script.

Two other optional arguments, --output_subdir_size and --output_subdir_prefix, can be used on input sets that are too large to write out to one directory. This depends on the limitations
of your file system, but you usually don't want 100,000 files written in the same directory.

If you have an FASTA file containing 95000 sequences, and use the following option:

```
--output_dir=/some/path
--output_subdir_size=30000
```

The following will be created:

directory	file count
/some/path/1/	30000
/some/path/2/	30000
/some/path/3/	30000
/some/path/4/	5000

If you choose to create a list file (and you probably want to), it will contain these proper paths.

You may not want the subdirectories to simply be numbers, as above, so you can use the --output_subdir_prefix option. For example:

```
--output_dir=/some/path
--output_subdir_size=30000
--output_subdir_prefix=fasta
```

The following will be created:

directory	file count
/some/path/fasta1/	30000
/some/path/fasta2/	30000
/some/path/fasta3/	30000
/some/path/fasta4/	5000

Finally, you can write multiple sequences to each output file using the --seqs_per_file option, which
can be used along with --outupt_subdir_size and --output_subdir_prefix. The main difference to note
is that, if you use --seqs_per_file, the fasta file created will no longer be named using values

```
taken from the header, since it will contain multiple headers.  Instead, the file will simply
be
named using sequential numbers starting at 1 (like 1.fsa).  For example:

    --output_dir=/some/path
    --output_subdir_size=3000
    --output_subdir_prefix=fasta
    --seqs_per_file=10

The following will be created:

    directory              file count
    --------------------------------
    /some/path/fasta1/     3000
    /some/path/fasta2/     3000
    /some/path/fasta3/     3000
    /some/path/fasta4/      500

=head1  CONTACT

    Joshua Orvis
    jorvis@tigr.org

=cut

use strict;
use Getopt::Long;
# qw(:config no_ignore_case no_auto_abbrev pass_through);
use Pod::Usage;
# BEGIN {
# use Ergatis::Logger;
# }

my %options = ();
my $results = GetOptions (\%options,
                          'input_file|i=s',
                          'output_dir|o=s',
                          'output_file_prefix|f=s',
                          'output_list|s=s',
                          'output_subdir_size|u=s',
                          'output_subdir_prefix|p=s',
                          'seqs_per_file|n|e=s',
                          'compress_output|c=s',
                          'log|l=s',
                          'debug=s',
                          'help|h') || pod2usage();

# my $logfile = $options{'log'} || Ergatis::Logger::get_default_logfilename();
# my $logger = new Ergatis::Logger('LOG_FILE'=>$logfile,
#                                  'LOG_LEVEL'=>$options{'debug'});
# $logger = $logger->get_logger();


my $logfile = $options{'log'} || "log.file";
```

```
my $logger = new logger('LOG_FILE'=>$logfile,
                        'LOG_LEVEL'=>$options{'debug'});

## display documentation
if( $options{'help'} ){
    pod2usage( {-exitval => 0, -verbose => 2, -output => \*STDERR} );
}

## make sure everything passed was peachy
&check_parameters(\%options);

## open the list file if one was passed
my $listfh;
if (defined $options{output_list}) {
    open($listfh, ">$options{output_list}") || $logger->logdie("couldn't create
$options{output_list} list file");
}

my $first = 1;
my $seq = '';
my $header;

my $sfh;

## load the sequence file
if ($options{'input_file'} =~ /\.(gz|gzip)$/) {
    open ($sfh, "<:gzip", $options{'input_file'})
      || $logger->logdie("can't open sequence file:\n$!");
} else {
    open ($sfh, "<$options{'input_file'}")
      || $logger->logdie("can't open sequence file:\n$!");
}

my $sub_dir = 1;
my $seq_file_count = 0;

## keep track of how many sequences are in the current output file
my $seqs_in_file = 0;
my $group_filename_prefix = 1;

## holds the output file handle
my $ofh;

while (<$sfh>) {
    ## if we find a header line ...
    if (/^\>(.*)/) {

        ## write the previous sequence before continuing with this one
        unless ($first) {
            &writeSequence(\$header, \$seq);

            ## reset the sequence
            $seq = '';
        }
```

```
        $first = 0;
        $header = $1;

    ## else we've found a sequence line
    } else {
        ## skip it if it is just whitespace
        next if (/^\s*$/);

        ## record this portion of the sequence
        $seq .= $_;
    }
}

## don't forget the last sequence
&writeSequence(\$header, \$seq);

exit;

sub check_parameters {
    my $options = shift;

    ## make sure input_file and output_dir were passed
    unless ( $options{input_file} && $options{output_dir} ) {
        $logger->logdie("You must pass both --input_file and --output_dir");
    }

    ## make sure input_file exists
    if (! -e $options{input_file} ) {
        if ( -e "$options{input_file}.gz" ) {
            $options{input_file} .= '.gz';
        } else {
            $logger->logdie("the input file passed ($options{input_file}) cannot be read or
does not exist");
        }
    }

    ## make sure the output_dir exists
    if (! -e "$options{output_dir}") {
        $logger->logdie("the output directory passed could not be read or does not exist");
    }

    ## seqs_per_file, if passed, must be at least one
    if (defined $options{seqs_per_file} && $options{seqs_per_file} < 1) {
        $logger->logdie("seq_per_file setting cannot be less than one");
    }

    ## handle some defaults
    $options{output_subdir_size}   = 0  unless ($options{output_subdir_size});
    $options{output_subdir_prefix} = '' unless ($options{output_subdir_prefix});
    $options{seqs_per_file}        = 1  unless ($options{seqs_per_file});
    $options{output_file_prefix} = '' unless ($options{output_file_prefix});
}
```

```
sub writeSequence {
    my ($header, $seq) = @_;

    ## the id used to write the output file will be the first thing
    ##  in the header up to the first whitespace.  get that.
    $$header =~ /^(\S+)/ || $logger->logdie( "can't pull out an id on header $$header" );
    my $id = $1;

    ## because it is going to be the filename, we're going to take out the characters that are
bad form to use
    ## legal characters = a-z A-Z 0-9 - . _
    $id =~ s/[^a-z0-9\-_.]/_/gi;

    my $dirpath;

    ## if we're writing more than one sequence to a file, change the id from
    ##  fasta header to the current group file name
    if ($options{seqs_per_file} > 1) {
        $id = $group_filename_prefix;

        ## did the user ask for a file prefix?
        if ( $options{output_file_prefix} ) {
            $id = $options{output_file_prefix} . $id;
        }
    }


    ## the path depends on whether we are using output subdirectories
    if ($options{output_subdir_size}) {
        $dirpath = "$options{'output_dir'}/$options{output_subdir_prefix}$sub_dir";
    } else {
        $dirpath = "$options{'output_dir'}";
    }

    ## did the user ask for a file prefix?
    my $filepath = "$dirpath/$id.fa";

    ## take any // out of the filepath
    $filepath =~ s|/+|/|g;

    ## write the sequence
    $logger->debug("Writing sequence to $filepath") if ($logger->is_debug());

    ## open a new output file if we need to
    ##  if we're writing multiple sequences per file, we only open a new
    ##  one when $seqs_in_file = 0 (first sequence)
    if ($seqs_in_file == 0) {

        ## if the directory we want to write to doesn't exist yet, create it
        mkdir($dirpath) unless (-e $dirpath);


        if ($options{'compress_output'}) {
            open ($ofh, ">:gzip", $filepath.".gz")
```

```
              || $logger->logdie("can't create '$filepath.gz':\n$!");
        } else {
            open ($ofh, ">$filepath") || $logger->logdie("can't create '$filepath':\n$!");

        }
        $seq_file_count++;

        ## add the file we just wrote to the list, if we were asked to
        if (defined $options{output_list}) {
            print $listfh "$filepath\n";
        }
    }

    ## if we're doing output subdirs and hit our size limit, increment to the next dir
    if ($options{output_subdir_size} && $options{output_subdir_size} == $seq_file_count) {
        $seq_file_count = 0;
        $sub_dir++;
    }

    ## write the sequence
    print $ofh ">$$header\n$$seq\n";
    $seqs_in_file++;

    ## if we hit the limit of how many we want in each file, set the next file name and
    ##  reset the count of seqs within the file
    if ($options{seqs_per_file} == $seqs_in_file) {
        $seqs_in_file = 0;
        $group_filename_prefix++;
    }
}


package logger;

sub new {
  my $packname= shift;
  my %args= @_;
  my $self= \%args;
  bless($self,$packname);
  return $self;
}

sub get_logger {
  my $self= shift;
  return $self;
}

sub logdie {
  my $self= shift;
  die @_;
}

sub debug {
  my $self= shift;
```

```
  warn @_;
}

sub is_debug {
  shift->{LOG_LEVEL} || 0;
}

1;
```

V.4 CAI

V.4.1 CAI calculation

```
################################################################################
#                                                                              #
#                        CAIcal/E-CAI PERL version 1.4                         #
#                              (July 2009)                                     #
#                                                                              #
#  This version of CAIcal/E-CAI calculates CAI or RCDI                          #
#  from DNA sequences or the expected value determined by randomly generating   #
#  sequences.                                                                  #
#                                                                              #
#               Created by Pere Puigbo (puigboap".'@'."ncbi.nlm.nih.gov)       #
#                                                                              #
#             Freely available at http://genomes.urv.es/CAIcal.                #
#                                                                              #
#             P.PuigbÚ, I.G.Bravo and S.Garcia-VallvÈ © 2007 URV               #
#                                                                              #
################################################################################
#
# Start credits and parametres
&credits();
&getParameters();

######################
# Get the parameters  #
######################
#
############################################################
# FUNCTION 0.a - Read parameters and execute CAIcal
############################################################
sub getParameters() {
        %parameters = @ARGV;

        if ($ARGV[0] eq '-help') {
                &readme();
                print "\n";
                exit;
        }
        # Get new parameters
        # introduced by the user
        $comp_parametres = 0;
        foreach $p(sort keys(%parameters)) {
                chomp $parameters{$p} ;
                if ($p eq '-e') {
                        if ($parameters{$p} eq '') {
                        }
                        elsif ( ($parameters{$p} eq 'rcdi')|| ($parameters{$p} eq 'ercdi')||
($parameters{$p} eq 'rcdi_and_ercdi') ||($parameters{$p} eq 'cai') || ($parameters{$p} eq
'expected') || ($parameters{$p} eq 'cai_and_expected')) {
                                print "$p $parameters{$p}
.................................................. ok\n";
                                $crida =  $parameters{$p};
                        }
```

```
                        else {
                                print "$p $parameters{$p}
..................................................... error\n";
                                exit;
                        }

                }
                elsif ($p eq '-f') {
                        if ($parameters{$p} eq '') {
                        }else {
                                #if (system "ls -f $parameters{$p}") {
                                #       print "$p $parameters{$p}
..................................................... error\n";
                                #       exit;
                                #}else {
                                        $fitxer = $parameters{$p};
                                        print "$p $parameters{$p}
..................................................... ok\n";
                                #}
                        }
                }
                elsif ($p eq '-h') {
                        if ($parameters{$p} eq '') {
                        }else {
                                #if (system "ls -f $parameters{$p}") {
                                #       print "$p $parameters{$p}
..................................................... error\n";
                                #       exit;
                                #}else {
                                        $hoste= $parameters{$p};
                                        print "$p $parameters{$p}
..................................................... ok\n";
                                #}
                        }
                }
                elsif ($p eq '-g') {
                        if ($parameters{$p} eq '') {
                        }else {
                                if ( ($parameters{$p} == 1) || ($parameters{$p} == 4) ||
($parameters{$p} == 11) || ($parameters{$p} == 2) || ($parameters{$p} == 3)|| ($parameters{$p}
== 5)|| ($parameters{$p} == 6)|| ($parameters{$p} == 9)|| ($parameters{$p} == 10)||
($parameters{$p} == 11)|| ($parameters{$p} == 12)|| ($parameters{$p} == 13)|| ($parameters{$p}
== 14)|| ($parameters{$p} == 15)) {
                                        $tipusGenoma= $parameters{$p};
                                        print "$p $parameters{$p}
..................................................... ok\n";
                                }
                                else {
                                        $tipusGenoma = $parameters{$p};
                                        print "$p $parameters{$p}
..................................................... error\n";
                                        exit;
                                }
                        }
```

```
		}
		elsif ($p eq '-c') {
			if ($parameters{$p} eq '') {
			}else {
				if ( ($parameters{$p} == 90) || ($parameters{$p} == 95) ||
($parameters{$p} == 99) ) {
					$errorC= $parameters{$p};
					print "$p $parameters{$p}
.................................................... ok\n";
				}
				else {
					print "$p $parameters{$p}
.................................................... error\n";
					exit;
				}
			}
		}
		elsif ($p eq '-p') {
			if ($parameters{$p} eq '') {
			}else {
				if ( ($parameters{$p} == 90) || ($parameters{$p} == 95) ||
($parameters{$p} == 99) ) {
					$population= $parameters{$p};
					print "$p $parameters{$p}
.................................................... ok\n";
				}
				else {
					print "$p $parameters{$p}
.................................................... error\n";
					exit;
				}
			}
		}
		elsif ($p eq '-gc') {
			if ($parameters{$p} eq '') {
				$intGC = "";
			}else {
				if ( ($parameters{$p} > 0 ) && ($parameters{$p} <= 100)  ) {
					$intGC= $parameters{$p};
					print "$p $parameters{$p}
.................................................... ok\n";
				}
				else {
					print "$p $parameters{$p}
.................................................... error\n";
					exit;
				}
			}
		}
		elsif ($p eq '-o1') {
			if ($parameters{$p} eq '') {
			}else {
				$fileCAI1= $parameters{$p};
```

```
                        print "$p $parameters{$p}
..................................................... ok\n";
                }
        }
        elsif ($p eq '-o2') {
                if ($parameters{$p} eq '') {
                }else {
                        $fileCAI2= $parameters{$p};
                        print "$p $parameters{$p}
..................................................... ok\n";
                }
        }
        elsif ($p eq '-o3') {
                if ($parameters{$p} eq '') {
                }else {
                        $fileExpected= $parameters{$p};
                        print "$p $parameters{$p}
..................................................... ok\n";
                }
        }
        elsif ($p eq '-n') {
                if ($parameters{$p} eq '') {
                }else {
                        if ( ($parameters{$p} =~ /[0-9]+/) && ($parameters{$p} !~ /[a-
z]+/) ){
                                $numSequences= $parameters{$p};
                                print "$p $parameters{$p}
..................................................... ok\n";
                        }else {
                                print "$p $parameters{$p}
..................................................... error\n";
                                exit;
                        }
                }
        }
        elsif ($p eq '-l') {
                if ($parameters{$p} eq '') {
                }else {
                        if ( ($parameters{$p} =~ /[0-9]+/) && ($parameters{$p} !~ /[a-
z]+/) ){
                                $long= $parameters{$p};
                                print "$p $parameters{$p}
..................................................... ok\n";
                        }else {
                                print "$p $parameters{$p}
..................................................... error\n";
                                exit;
                        }
                }
        }elsif ($p eq '-m') {
                # Methods: montecarlo, poisson, random.
                if ($parameters{$p} eq '') {
                }else {
```

```
                        if ( ($parameters{$p} eq 'markov') || ($parameters{$p} eq
'poisson') || ($parameters{$p} eq 'random') ) {
                                if ($method eq "markov") {
                                        $method = "markov";
                                }else {
                                        $method = $parameters{$p};
                                }
                                print "$p $parameters{$p}
..................................................... ok\n";
                        }else {
                                print "$p $parameters{$p}
..................................................... error\n";
                                exit;
                        }
                }
        }

        elsif ($p eq '-s') {
                if ($parameters{$p} eq '') {
                }
                elsif  ( ($parameters{$p} ne 'y') && ($parameters{$p} ne 'n') ) {
                        print "$p $parameters{$p}
..................................................... error\n";
                        exit;
                }
                else {
                        $splitseq = $parameters{$p};
                        print "$p $parameters{$p}
..................................................... ok\n";
                }
        }

        $comp_parametres++;
}

#####################
# Default parameters: list of options that are executed by default in CAIcal
#####################

# By default the CAIcal program calculates CAI and E-CAI values
if ($crida eq "") {
        $crida = "cai_and_expected";
        print "-e $crida ..............................................
default\n";
}

# Exectuion of Example.ffn by default
if ($fitxer eq "") {
        $fitxer = "example.ffn";
        #if (system "ls -f $fitxer") {
        #       &readme();
        #       print "\n";
        #       exit;
        #} else {
```

```
                print "-f $fitxer ..............................................
default\n";
        #}
    }

    # By default the program CAIcal program calculates only an E-CAI value for all
sequences
    if ($splitseq eq "") {
        $splitseq = "n";
        print "-s $splitseq ...................................................
default\n";
    }

    # The CAIcal program uses the mean codon usage of human as a reference table
    if ($hoste eq "") {
        $hoste = "human";
        print "-h $hoste ......................................................
default\n";
    }

    # The Default genetic code is the Standard Genetic code (1)
    if ($tipusGenoma eq "") {
        $tipusGenoma = 1;
        print "-g $tipusGenoma ................................................
default\n";
    }

    # By default the CAIcal program uses a 95% of confidence
    if ($errorC eq "") {
        $errorC = 95;
        print "-c $errorC .....................................................
default\n";
    }

    # By default the CAIcal program uses a 99% of coverage
    if ($population eq "") {
        $population = 99;
        print "-p $population .................................................
default\n";
    }

    # By default the output files are called: cai, random_sequences_and_cai and expected
    if ($fileCAI1 eq "") {
        $fileCAI1 = "cai";
        print "-o1 $fileCAI1 ................................. default\n";
    }
    if ($fileCAI2 eq "") {
        $fileCAI2 = "random_sequences_and_cai";
        print "-o2 $fileCAI2 ............................... default\n";
    }

    if ($fileExpected eq "") {
        $fileExpected = "expected";
```

```
		print "-o3 $fileExpected ..............................................
default\n";
	}

	# By default the CAIcal program generates 1000 random sequences with 300 of length
	if ($numSequences eq "") {
		$numSequences = 1000;
		print "-n $numSequences ...................................................
default\n";
	}

	if ($long eq "") {
		$long = 300;
		print "-l $long ...................................................
default\n";
	}


	# By default the CAIcal program uses the markov method to generate random sequences
	if ($method eq '') {
		$method = "markov";
		print "-m markov ..................................................
default\n";
	}

	# By default the CAIcal program executes the Example
	if ($comp_parametres == 0) {
		print "\nCAIcal is running using an example. Files created: 1) cai, 2)
random_sequences_and_cai and 3) expected. Please read in the following 'readme' the parameters
to use CAIcal.\n";
		print "\n";
		&readme;
		print "\n";
	}

	#####################
	#Programs execution
	#####################
	if ($splitseq eq "y") {
		if ( ($crida eq 'cai') || ($crida eq 'expected') || ($crida eq
'cai_and_expected') ) {
			# Execution of the split option
			# Parameters required: input file with sequences, type of genetic code,
codon usage reference table, output file 1, percentage of confidence, percentage of coverage,
output file 2, number of sequences, sequence length, method, G+C introduced (or blank) and
Output file 3
			&split_seq($fitxer, $tipusGenoma, $hoste, $fileCAI1, $errorC,
$population, $fileCAI2, $numSequences, $long, $method, $intGC, $fileExpected);
		}else {
			# Execution of the split option
			# Parameters required: input file with sequences, type of genetic code,
codon usage reference table, output file 1, percentage of confidence, percentage of coverage,
output file 2, number of sequences, sequence length, method, G+C introduced (or blank) and
Output file 3
```

```
                        &split_seq_rcdi($fitxer, $tipusGenoma, $hoste, $fileCAI1, $errorC,
$population, $fileCAI2, $numSequences, $long, $method, $intGC, $fileExpected);
                }
        }
        else {
                if ($crida eq 'cai') {
                        # Execution of CAI calculation
                        #Parameters required: input file with sequences in fasta format, type of
genetic code, file with codon usage reference table and Output file
                        print "\nCalculating CAIs.";
                        &sequencies($fitxer, $tipusGenoma, $hoste, $fileCAI1 );
                        print "\n..................................... Done.\n";
                        sleep(1);
                }
                elsif ($crida eq 'expected') {
                        # Execution of E-CAI calculation
                        # Parameters required: input file with sequences in fasta format, type
of genetic code, file with codon usage reference table, percentage of confidence, percentage
of coverage, Output file 2, number of sequences, length, method and G+C introduced (or blank)
                        print "\nExpected CAI calculation.\n";
                        if ($method eq "random") {
                                &CalcMitjanaCAIrandom2($fitxer, $tipusGenoma, $hoste, $errorC,
$population, $fileCAI2, $numSequences, $long, $method, $intGC);
                        }else {
                                &CalcMitjanaCAIrandom($fitxer, $tipusGenoma, $hoste, $errorC,
$population, $fileCAI2, $numSequences, $long, $method, $intGC);
                        }
                        print "..................................... Done.\n";
                        sleep(1);
                }
                elsif ($crida eq 'cai_and_expected') {

                        # Execution of CAI calculation
                          #Parameters required: input file with sequences in fasta format, type
of genetic code, file with codon usage reference table and Output file
                        print "\nCalculating CAIs.";
                        &sequencies($fitxer, $tipusGenoma, $hoste, $fileCAI1);
                        print "\n..................................... Done.\n";
                        print "Calculating expected value.\n";
                        sleep(1);

                        # Execution of E-CAI calculation
                          # Parameters required: input file with sequences in fasta format, type
of genetic code, file with codon usage reference table, percentage of confidence, percentage
of coverage, Output file 2, number of sequences, length, method and G+C introduced (or blank)
                        if ($method eq "random") {
                                &CalcMitjanaCAIrandom2($fitxer, $tipusGenoma, $hoste, $errorC,
$population, $fileCAI2, $numSequences, $long, $method);
                        }else {
                                &CalcMitjanaCAIrandom($fitxer, $tipusGenoma, $hoste, $errorC,
$population, $fileCAI2, $numSequences, $long, $method);
                        }
                        print "..................................... Done.\n";
                        sleep(1);
```

```
	}elsif ($crida eq 'rcdi') {
		# Execution of rcdi calculation
		#Parameters required: input file with sequences in fasta format, type of
genetic code, file with codon usage reference table and Output file
		my $calcrcdi = "Y";
		if ($fileCAI1 eq "cai") {
			$fileCAI1 = "rcdi";
			print "-o1 $fileCAI1 ..................................... ok\n";
		}
		print "\nCalculating RCDIs.";
		&sequencies_rcdi($fitxer, $tipusGenoma, $hoste, $fileCAI1 );
		print "\n..................................... Done.\n";
		sleep(1);
	}elsif ($crida eq 'ercdi') {
		my $calcrcdi = "Y";
		# Execution of E-RCDI calculation
		# Parameters required: input file with sequences in fasta format, type
of genetic code, file with codon usage reference table, percentage of confidence, percentage
of coverage, Output file 2, number of sequences, length, method and G+C introduced (or blank)

		print "\nExpected RCDI calculation.\n";
		$fileCAI2 = "random_sequences_and_rcdi";
		print "-o2 $fileCAI2
.................................................. ok\n";

		my $calcrcdi = "Y";
		&CalcMitjanaCAIrandom($fitxer, $tipusGenoma, $hoste, $errorC,
$population, $fileCAI2, $numSequences, $long, $method, $intGC, $calcrcdi);

		print "..................................... Done.\n";
		sleep(1);
	}elsif ($crida eq 'rcdi_and_ercdi') {
		# Execution of rcdi calculation
		#Parameters required: input file with sequences in fasta format, type of
genetic code, file with codon usage reference table and Output file
		my $calcrcdi = "Y";
		if ($fileCAI1 eq "cai") {
			$fileCAI1 = "rcdi";
			print "-o1 $fileCAI1 ..................................... ok\n";
		}
		print "\nCalculating RCDIs.";
		&sequencies_rcdi($fitxer, $tipusGenoma, $hoste, $fileCAI1 );
		print "\n..................................... Done.\n";
		sleep(1);

		# Execution of E-RCDI calculation
		# Parameters required: input file with sequences in fasta format, type
of genetic code, file with codon usage reference table, percentage of confidence, percentage
of coverage, Output file 2, number of sequences, length, method and G+C introduced (or blank)
		print "\nExpected RCDI calculation. \n";
		$fileCAI2 = "random_sequences_and_rcdi";
		print "-o2 $fileCAI2
.................................................. ok\n";
```

```
                        my $calcrcdi = "Y";
                        &CalcMitjanaCAIrandom($fitxer, $tipusGenoma, $hoste, $errorC,
$population, $fileCAI2, $numSequences, $long, $method, $intGC, $calcrcdi);

                        print "..................................... Done.\n";
                        sleep(1);
                }
        }
        print "\nThanks for using CAIcal ................................ Bye.\n\n";
}


########################################
# Expected CAI calculation markov/poisson #
########################################
#
############################################################
# FUNCTION 1.b - CAI and E-CAI calculation
############################################################
#
# Input parameters: input file with sequences in fasta format, type of genetic code, file with
codon usage reference table and Output file
# Result: output file with E-CAI value

sub CalcMitjanaCAIrandom {
        my ($fitxer, $tipusGenoma, $hoste, $errorC, $population, $fileCAI, $numSequences,
$long, $method, $intGC, $calcrcdi) = @_;

        open (OUT1, ">$fileCAI") || die "cannot open $fileCAI\n";

        if ($calcrcdi eq "Y") {
                print OUT1 "RCDI\tSEQUENCE\n";
        }else {
                print OUT1 "CAI\tSEQUENCE\n";
        }

        # Genetic Code
        my @codigenetic_aux = &geneticCode($tipusGenoma);
        my %codigenetic = @codigenetic_aux;
        my @CodonsSTOP = &stopCodons(@codigenetic_aux);
        my @CodonsPerAA_aux = &codonsPerAA(@codigenetic_aux);
        my %CodonsPerAA = @CodonsPerAA_aux;
        my @varAA_aux = &varAA(@CodonsPerAA_aux);
        my %varAA = @varAA_aux;
        my @CodonsUnics = &CodonsUnics(@codigenetic_aux);
        my @CodonsVariables = &CodonsVariables(@codigenetic_aux);
        my $countVariables = &nCodonsVariables(@CodonsVariables);


        # Reading Input file to  get sequences
        open (IN, "<$fitxer") || die "cannot open $fitxer\n";
        my $sequence__ = "";
        my $nom__ == "";
        my $inicial__ = 0;
```

```
	my %sequenciesORIGINALS = ();
	while (<IN>) {
		chomp;
		if (/^>/) {
			if ($inicial__ == 1) {
				$sequenciesORIGINALS{$nom__} = $sequence__;
				$sequence__ = "";
			}
			$nom__ = $_;
		}
		else {
			$sequence__ .= $_;
		}
		$inicial__ = 1;
	}
	$sequenciesORIGINALS{$nom__} = $sequence__;
	close IN;

	#Calculation of mean amino acids composition and standard deviations
	my $nombreSEQ = 0;
	my %aminoacidsTotal__ = ();
	my $GC_total=0;
	my $GC_1s = 0;
	my $GC_2s = 0;
	my $GC_3s= 0;
	my $codonsSEQ = 0;
	my %aminoacidsParcial__ = ();
	my %nombreCodons = ();
	my %aminoacidsParcial__ = ();
	my %GC1Parcial__ = ();
	my %GC2Parcial__ = ();
	my %GC3arcial__ = ();

	foreach $nom__(keys(%sequenciesORIGINALS)) {
		$sequenciesORIGINALS{$nom__} =~ tr/actguU/ACTGTT/;
		my $stringSEQ = $sequenciesORIGINALS{$nom__};
		$missatge = 'ok';

		# reading error messages
		($missatge, $opmissatge) = &errors($nom__, $stringSEQ, $tipusGenoma);


		if ($missatge eq 'ok') {
			# If there are not errors

			#Calculation of amino acids frequency
			$stringSEQ =~ s/(...)/$1 /g;
			$codonsSEQ = 0;
			foreach $codo__(sort keys(%codigenetic)) {
				$AA__ = $codigenetic{$codo__};
				$aminoacidsParcial__{$nombreSEQ}{$AA__} += $stringSEQ =~
s/($codo__ )/$1/g;
			}
```

```
                        #Calculation of codon usage of each sequence
                        $codonsSEQ = $stringSEQ =~ s/(... )/$1/g;
                        $nombreCodons{$nombreSEQ} = $codonsSEQ;
                        foreach $AA__(sort keys(%CodonsPerAA)) {
                                $aminoacidsParcial__{$nombreSEQ}{$AA__} /=
$nombreCodons{$nombreSEQ};
                                $aminoacidsParcial__{$nombreSEQ}{$AA__} *= 100;
                                $aminoacidsTotal__{$AA__} +=
$aminoacidsParcial__{$nombreSEQ}{$AA__};
                        }

                        #G+C calculation
                        $GC_parcial = $stringSEQ =~ s/([C|G])/$1/g;
                        $GC_parcial /= ($nombreCodons{$nombreSEQ}*3);
                        $GC_total += $GC_parcial;

                        #G+C at the three positions of the codon calculation
                        $stringSEQ2 = $stringSEQ;
                        for $cOdO(@CodonsSTOP) {
                                $stringSEQ2 =~ s/$cOdO //g;
                        }
                        for $cOdO(@CodonsUnics) {
                                $stringSEQ2 =~ s/$cOdO //g;
                        }
                        $codonsSEQ2 = $stringSEQ2 =~ s/(... )/$1/g;

                        $GC1_parcial = $stringSEQ2 =~ s/([C|G]..\s)/$1/g;
                        $GC1Parcial__{$nombreSEQ} = $GC1_parcial;
                        $GC1_parcial /= $codonsSEQ2;
                        $GC_1s += $GC1_parcial;

                        $GC2_parcial = $stringSEQ2 =~ s/(.[C|G].\s)/$1/g;
                        $GC2Parcial__{$nombreSEQ} = $GC2_parcial;
                        $GC2_parcial /= $codonsSEQ2;
                        $GC_2s += $GC2_parcial;

                        $GC3_parcial = $stringSEQ2 =~ s/(..[C|G]\s)/$1/g;
                        $GC3Parcial__{$nombreSEQ} = $GC3_parcial;
                        $GC3_parcial /= $codonsSEQ2;
                        $GC_3s += $GC3_parcial;

                        $nombreSEQ++;
                }else {
                        # If there are errors. See error messages

                        if ($opmissatge == 1) {
                                print "$missatge ";
                                print " This sequence is not used to calculates the expected
CAI.\n";
                        }elsif ($opmissatge ==2) {
                                #Calculates amino acids frequency
                                $stringSEQ =~ s/(...)/$1 /g;
                                $codonsSEQ = 0;
                                %aminoacidsParcial__ = ();
```

```
                                foreach $codo__(sort keys(%codigenetic)) {
                                        $AA__ = $codigenetic{$codo__};
                                        $aminoacidsParcial__{$nombreSEQ}{$AA__} += $stringSEQ =~
s/($codo__ )/$1/g;
                                }

                                #Codon of the sequences
                                $codonsSEQ = $stringSEQ =~ s/(... )/$1/g;
                                $nombreCodons{$nombreSEQ} = $codonsSEQ;
                                foreach $AA__(sort keys(%CodonsPerAA)) {
                                        $aminoacidsParcial__{$nombreSEQ}{$AA__} /=
$nombreCodons{$nombreSEQ};
                                        $aminoacidsParcial__{$nombreSEQ}{$AA__} *= 100;

                                        $aminoacidsTotal__{$AA__} +=
$aminoacidsParcial__{$nombreSEQ}{$AA__};
                                }
                                #G+C calculation
                                $GC_parcial = $stringSEQ =~ s/([C|G])/$1/g;
                                $GC_parcial /= ($nombreCodons{$nombreSEQ}*3);
                                $GC_total += $GC_parcial;

                                #G+C at the three positions of the codon calculation
                                $stringSEQ2 = $stringSEQ;
                                for $cOdO(@CodonsSTOP) {
                                        $stringSEQ2 =~ s/$cOdO //g;
                                }
                                for $cOdO(@CodonsUnics) {
                                        $stringSEQ2 =~ s/$cOdO //g;
                                }
                                $codonsSEQ2 = $stringSEQ2 =~ s/(... )/$1/g;

                                $GC1_parcial = $stringSEQ2 =~ s/([C|G]..\s)/$1/g;
                                $GC1Parcial__{$nombreSEQ} = $GC1_parcial;
                                $GC1_parcial /= $codonsSEQ2;
                                $GC_1s += $GC1_parcial;

                                $GC2_parcial = $stringSEQ2 =~ s/(.[C|G].\s)/$1/g;
                                $GC2Parcial__{$nombreSEQ}= $GC2_parcial;
                                $GC2_parcial /= $codonsSEQ2;
                                $GC_2s += $GC2_parcial;

                                $GC3_parcial = $stringSEQ2 =~ s/(..[C|G]\s)/$1/g;
                                $GC3Parcial__{$nombreSEQ} = $GC3_parcial;
                                $GC3_parcial /= $codonsSEQ2;
                                $GC_3s += $GC3_parcial;

                                $nombreSEQ++;
                                print "$missatge ";
                                print " However this sequence is used to calculate the expected
CAI.\n";
                        }
                }
        }
```

```
# CHi-square test for GC
# This test checks the homogeneity if G+C composition
my $i__ = 0;
my $df_gc = 3-1;
my $chi_testAAH0 = 0;
my $chi_testAAH1 = 0;
while ($i__ < $nombreSEQ) {
	my $chi_exp_gc = 0;
	# GC1
	if ($GC1Parcial__{$i__} == 0) {
	}else {
		$obs_gc1 = ($GC_1s/$nombreSEQ) * $nombreCodons{$i__};
		$esp_gc1 = $GC1Parcial__{$i__};
		$chi_exp_gc1 = (($obs_gc1 - $esp_gc1)**2)/$esp_gc1;
		$chi_exp_gc += $chi_exp_gc1;
	}
	# GC2
	if ($GC2Parcial__{$i__} == 0) {
	}else {
		$obs_gc2 = ($GC_2s/$nombreSEQ) * $nombreCodons{$i__};
		$esp_gc2 = $GC2Parcial__{$i__};
		$chi_exp_gc2 = (($obs_gc2 - $esp_gc2)**2)/$esp_gc2;
		$chi_exp_gc += $chi_exp_gc2;
	}

	# GC3
	if ($GC3Parcial__{$i__} == 0) {
	}else {
		$obs_gc3 = ($GC_3s/$nombreSEQ) * $nombreCodons{$i__};
		$esp_gc3 = $GC3Parcial__{$i__};
		$chi_exp_gc3 = (($obs_gc3 - $esp_gc3)**2)/$esp_gc3;
		$chi_exp_gc += $chi_exp_gc3;
	}

	# Chi-square test
	$chi_table_gc = &chi_square_test($df_gc);
	if ($chi_exp_gc <= $chi_table_gc) {
		$chi_testGCH0++;
	}else {
		$chi_testGCH1++
	}
	$i__++;
}
#Chi-Square Goodness-of-Fit Test for GC
# Result of the test:
$Perc_chi_GCH1 = ( $chi_testGCH0 / ($chi_testGCH0+$chi_testGCH1) ) * 100;

# CHi-square test for AA compostion
 # This test checks the homogeneity if AA composition
foreach $AA__(keys(%aminoacidsTotal__)) {
	$aminoacidsTotal__{$AA__} /= $nombreSEQ;
}
# Standard deviation and chi-square test
```

```
	my $i__ = 0;
	my %aminoacidsTotal__SD = ();
	$parcial = 0;
	my $chi_testAAH0 = 0;
	my $chi_testAAH1 = 0;
	while ($i__ < $nombreSEQ) {
		$chi_parcial = 0;
		$chi_exp = 0;
		$chi_table= 0;
		$contAA__=0;
		foreach $AA__(keys(%CodonsPerAA)) {
			if ($aminoacidsParcial__{$i__}{$AA__} == 0) {
			}else {
				#SD
				$parcial = ( $aminoacidsParcial__{$i__}{$AA__} -
$aminoacidsTotal__{$AA__} )**2 ;
				$aminoacidsTotal__SD{$AA__} += $parcial;

				#chi-square test
				$obs_chi =
($aminoacidsParcial__{$i__}{$AA__}/100)*$nombreCodons{$i__};
				$esp_chi = ($aminoacidsTotal__{$AA__}/100)*$nombreCodons{$i__};
				$chi_parcial = (($obs_chi - $esp_chi )**2)/$esp_chi;
				$chi_exp += $chi_parcial;
				$contAA__++;
			}
		}
		# Chi-square significance for AA
		$df  = $contAA__ - 1;
		$chi_table = &chi_square_test($df);
		if ($chi_exp <= $chi_table) {
			$chi_testAAH0++;
		}else {
			$chi_testAAH1++;
		}
		$i__++;
	}

	# Chi-Square Goodness-of-Fit Test for AA
	# Result of Chi-square test:
	$Perc_chi_AAH1 = ( $chi_testAAH0 / ($chi_testAAH0+$chi_testAAH1) ) * 100;

	foreach $AA__(keys(%aminoacidsTotal__SD)) {
		$previ = $aminoacidsTotal__SD{$AA__} / $nombreSEQ;
		$aminoacidsTotal__SD{$AA__} = sqrt($previ);
	}

	#G+C composition used to generate random sequences
	if ($intGC  == "") {
		$GC_total /= $nombreSEQ;
		$GC_1s /= $nombreSEQ;
		$GC_2s /= $nombreSEQ;
		$GC_3s /= $nombreSEQ;
```

```
		$GC_total *= 100;
		$GC_1s *= 100;
		$GC_2s *= 100;
		$GC_3s *= 100;
	}else {
		$GC_total = $intGC;
		$GC_1s = $intGC;
		$GC_2s = $intGC;
		$GC_3s = $intGC;
	}

	# Generates -n random sequences with a specific length (-l)
	# Variables required: $nombreSEQ, %aminoacidsTotal__, %aminoacidsTotal__SD, $GC_total,
$GC_1s, $GC_2s, $GC_3s
		my $c=0;
		my $totalCAI=0;
		my @totsCAIs = ();
		# while number of new sequences is less than -n
		while ($c < $numSequences) {
			$c++;
			my $segNTstring ="";
			my $segAAstring ="";
			my $patternG3 = "..G";
			my $patternC3 = "..C";

			###############
			#    MARKOV   #
			###############
			if ($method eq 'markov') {
				# while length of sequence is less than $long proceed
				while ( (length($segNTstring)/3) < $long) {
					#Get the random value
					my $R = int(rand(100))+1;
					my $suma =0;
					# Search one AA at "random"
					foreach $aa(keys(%aminoacidsTotal__)) {
						if ($aa eq '.') {
						}
						else {
							my $freq = $aminoacidsTotal__{$aa};
							# Paa = Mitjana +  nael * SD
							my $sd = $aminoacidsTotal__SD{$aa};
							if (int(rand(10)) >= 5) {
								$nale = int(rand(101))/100;
							}else {
								$nale = int(rand(-101))/100;
							}
							$Paa = $freq + ($nale * $sd);

							$suma += $Paa;

							if ($suma > $R) {
								$segAAstring .= $aa;
								# Search one codon at "random"
```

```
$codigenetic{$codo};

($tipusGenoma =~ /1|3|4|6|10|11|12|15/) {
	#$RcodonsR = int(rand(101));
($RcodonsR > $GC_1s) {
	$compR1 = "A..";
	$compR2 = "T..";
{
	$compR1 = "C..";
	$compR2 = "G..";
((($codo =~ /$compR1/)||($codo =~ /$compR2/)) {
	if (($codo =~ /$patternG3/)||($codo =~ /$patternC3/)) {
		$codons{$codo} = $GC_3s;
	}
	else {
		$codons{$codo} = 100-$GC_3s;
	}
((($codo =~ /$patternG3/)||($codo =~ /$patternC3/)) {
	$codons{$codo} = $GC_3s;
{
	$codons{$codo} = 100-$GC_3s;
```

```
my %codons= ();

$RcodonsR = int(rand(101));
$RcodonsL = int(rand(101));
$RcodonsS = int(rand(101));
foreach $codo(keys(%codigenetic)) {
	my $cod_aa =
	if ($cod_aa eq $aa) {
		if ($aa eq 'R'){
			if
				if
				}
				else
				}
				if
			}
		}else {
			if
			}
			else
			}
```

```
                                                                        }
                                                                }
                                                                elsif ($aa  eq 'L')
{
                                                                        if
($tipusGenoma !~ /3/) {
        #$RcodonsL = int(rand(101));
                                                                                if
($RcodonsL > $GC_1s) {
        $compL1 = "A..";
        $compL2 = "T..";
                                                                                }
                                                                                else
{
        $compL1 = "C..";
        $compL2 = "G..";
                                                                                }
                                                                                if
(($codo =~ /$compL1/)||($codo =~ /$compL2/)) {
        if (($codo =~ /$patternG3/)||($codo =~ /$patternC3/)) {
                $codons{$codo} = $GC_3s;
        }
        else {
                $codons{$codo} = 100-$GC_3s;
        }
                                                                                }
                                                                        }else {
                                                                                if
(($codo =~ /$patternG3/)||($codo =~ /$patternC3/)) {
        $codons{$codo} = $GC_3s;
                                                                                }
                                                                                else
{
        $codons{$codo} = 100-$GC_3s;
                                                                                }
                                                                        }
                                                                }
                                                                else {
                                                                        if (($codo =~
/$patternG3/)||($codo =~ /$patternC3/)) {
```

```
        $codons{$codo} = $GC_3s;
                                                                        }
                                                                        else {

        $codons{$codo} = 100-$GC_3s;
                                                                        }
                                                                }
                                                        }
                                                }
                                                # Sum frequencies
                                                my $sumaFreqCodons = 0;
                                                foreach $codi(keys(%codons)) {
                                                        $fri = $codons{$codi};
                                                        $sumaFreqCodons += $fri;
                                                }
                                                $Rcodons =
int(rand($sumaFreqCodons)+1);
                                                my $sumaCodons = 0;
                                                my $trobat = 0;
                                                foreach $codi(keys(%codons)) {
                                                        $fri = $codons{$codi};
                                                        $sumaCodons += $fri;
                                                        if ($sumaCodons >= $Rcodons)
{
                                                                $segNTstring .=
$codi;
                                                                last;
                                                        }
                                                }
                                                last;
                                        }
                                }
                        }
                }

        } elsif ($method eq 'poisson') {
        ###############
        #   POISSON   #
        ###############
                # Create AA sequence
                #$long = 100;
                my $llseq = 0;
                while ($llseq < $long) {
                        foreach $aa(keys(%aminoacidsTotal__)) {
                                if ($aa eq '.') {
                                }
                                else {
                                        my $freq = $aminoacidsTotal__{$aa};
                                        $Paa = &calcPoisson($freq);
                                        while ($Paa > 0) {
                                                $segAAstring .= $aa;
                                                $Paa--;
                                        }
```

```
                                        }
                                }
                                $llseq = length($segAAstring);
                        }

                        @seqAAarray = split '', $segAAstring;
                        foreach $aa(@seqAAarray) {
                                # Search codon at random
                                my %codons= ();
                                $RcodonsR = int(rand(101));
                                $RcodonsL = int(rand(101));
                                $RcodonsS = int(rand(101));
                                foreach $codo(keys(%codigenetic)) {
                                        my $cod_aa = $codigenetic{$codo};
                                        if ($cod_aa eq $aa) {
                                                if ($aa eq 'R'){
                                                        if ($tipusGenoma =~
/1|3|4|6|10|11|12|15/) {
                                                                #$RcodonsR = int(rand(101));
                                                                if ($RcodonsR > $GC_1s) {
                                                                        $compR1 = "A..";
                                                                        $compR2 = "T..";
                                                                }
                                                                else {
                                                                        $compR1 = "C..";
                                                                        $compR2 = "G..";
                                                                }
                                                                if (($codo =~
/$compR1/)||($codo =~ /$compR2/)) {
                                                                        if (($codo =~
/$patternG3/)||($codo =~ /$patternC3/)) {

        $codons{$codo} = $GC_3s;
                                                                }
                                                                else {
                                                                        $codons{$codo} =
100-$GC_3s;
                                                                }
                                                        }
                                                        }else {
                                                                if (($codo =~
/$patternG3/)||($codo =~ /$patternC3/)) {
                                                                        $codons{$codo} =
$GC_3s;
                                                                }
                                                                else {
                                                                        $codons{$codo} =
100-$GC_3s;
                                                                }
                                                        }
                                                }
                                                elsif ($aa  eq 'L') {
                                                        if ($tipusGenoma !~ /3/) {
                                                                #$RcodonsL = int(rand(101));
```

```
                                                        if ($RcodonsL > $GC_1s) {
                                                                $compL1 = "A..";
                                                                $compL2 = "T..";
                                                        }
                                                        else {
                                                                $compL1 = "C..";
                                                                $compL2 = "G..";
                                                        }
                                                        if (($codo =~
/$compL1/)||($codo =~ /$compL2/)) {
                                                                if (($codo =~
/$patternG3/)||($codo =~ /$patternC3/)) {

        $codons{$codo} = $GC_3s;
                                                                }
                                                                else {

        $codons{$codo} = 100-$GC_3s;
                                                                }
                                                        }
                                                }else {
                                                        if (($codo =~
/$patternG3/)||($codo =~ /$patternC3/)) {
                                                                $codons{$codo} =
$GC_3s;
                                                        }
                                                        else {
                                                                $codons{$codo} =
100-$GC_3s;
                                                        }
                                                }
                                        }
                                        else {
                                                if (($codo =~ /$patternG3/)||($codo
=~ /$patternC3/)) {
                                                        $codons{$codo} = $GC_3s;
                                                }
                                                else {
                                                        $codons{$codo} = 100-$GC_3s;
                                                }
                                        }
                                }
                        }
                        # Sum frequencies
                        my $sumaFreqCodons = 0;
                        foreach $codi(keys(%codons)) {
                                $fri = $codons{$codi};
                                $sumaFreqCodons += $fri;
                        }
                        $Rcodons = int(rand($sumaFreqCodons)+1);
                        my $sumaCodons = 0;
                        my $trobat = 0;
                        foreach $codi(keys(%codons)) {
                                $fri = $codons{$codi};
```

```
                                        $sumaCodons += $fri;
                                        if ($sumaCodons >= $Rcodons) {
                                                $segNTstring .= $codi;
                                                last;
                                        }
                                }
                        }
                }

                if ($calcrcdi eq "Y") {
                        ######################################################
                        # RCDI calculation and print the sequence into an output file
                        ######################################################
                        my $RCDI = &calcularElRCDI($segNTstring, $tipusGenoma, $hoste);
                        printf OUT1 "%6.3f\t$segNTstring\n", $RCDI;
                        @totsCAIs[$c] = $RCDI;
                        $totalCAI += $RCDI;

                }else {
                        ######################################################
                        # CAI calculation and print the sequence into an output file
                        ######################################################
                        my $CAI = &calcularElCAI($segNTstring, $tipusGenoma, $hoste);
                        printf OUT1 "%6.3f\t$segNTstring\n", $CAI;
                        @totsCAIs[$c] = $CAI;
                        $totalCAI += $CAI;
                }
        }
        close OUT1;
        # Mean CAI calculation
        my $MitjanaCAI = $totalCAI / $numSequences;

        # Standard Deviation of CAI
        my $DesvEstCAI = 0;
        my $difCAI = 0;
        foreach $CAI(@totsCAIs) {
                if ($CAI == 0) {
                }else {
                        $difCAI_ =  ($CAI - $MitjanaCAI)**2;
                        $DesvEstCAI += $difCAI_;
                }
        }
        $DesvEstCAI = sqrt($DesvEstCAI/$numSequences);

        # Confidence and Coverage
        $ErrorCAI = &CalcConfidence($numSequences, $errorC, $population);
        my $LimitSupCAI = $MitjanaCAI + ($ErrorCAI*$DesvEstCAI);

        # Performing Normality test Kolmogorov-Smirnov
        print "$MitjanaCAI, $DesvEstCAI,  $numSequences,\n";
        my ($NORMAL, $maxD, $c95) = &KStest($MitjanaCAI, $DesvEstCAI,  $numSequences,
@totsCAIs);
        #print "($NORMAL, $maxD, $c95)\n";
```

```
	########################
	# RESULTS
	########################
	if ($calcrcdi eq "Y") {
		#####
		print "* Number of random sequences:\t$numSequences\n";
		printf "* RCDI Average:\t%6.3f\n", $MitjanaCAI;
		printf "* Standard deviation:\t%6.3f\n", $DesvEstCAI;
		printf "* Upper (one-side) Tolerance Limit (eRCDI):\t%6.3f\n", $LimitSupCAI;
		#####
		open (OUT2, ">$fileExpected") || die "cannot open $fileExpected";
		print OUT2 "EXPECTED RCDI at "."$errorC"."% confident that contain
"."$population"."% of population\n";
		printf OUT2 "Upper (one-side) Tolerance Limit (eRCDI):\t%6.3f\n", $LimitSupCAI;
		printf OUT2 "RCDI Average:\t%6.3f\n", $MitjanaCAI;
		printf OUT2 "Standard deviation:\t%6.3f\n", $DesvEstCAI;
		print OUT2 "Number of random sequences: $numSequences\n";
		if ($NORMAL eq 'TRUE') {
			printf OUT2 "Kolmogorov-Smirnov test for the expected RCDI(alpha = 5%):
%6.5f < Critical Value(%6.5f) [NORMALITY]\n", $maxD, $c95;
		}else {
			printf OUT2 "Kolmogorov-Smirnov test for the expected RCDI (alpha = 5%):
%6.5f >= Critical Value(%6.5f) [NOT NORMALITY]\n", $maxD, $c95;
		}
		printf OUT2 "Chi-Square Goodness-of-Fit test for AA (alpha = 5%): the %3.1f",
$Perc_chi_AAH1;
		print OUT2 "% of sequences fit the AA distribution.\n";

		if ($intGC  == "") {
			printf OUT2 "Chi-Square Goodness-of-Fit test for GC (alpha = 5%): the
%3.1f", $Perc_chi_GCH1;
			print OUT2 "% of sequences fit the GC distribution.\n";
		}
	}else {
		#####
		print "* Number of random sequences:\t$numSequences\n";
		printf "* CAI Average:\t%6.3f\n", $MitjanaCAI;
		printf "* Standard deviation:\t%6.3f\n", $DesvEstCAI;
		printf "* Upper (one-side) Tolerance Limit (eCAI):\t%6.3f\n", $LimitSupCAI;
		#####
		open (OUT2, ">$fileExpected") || die "cannot open $fileExpected";
		print OUT2 "EXPECTED CAI at "."$errorC"."% confident that contain
"."$population"."% of population\n";
		printf OUT2 "Upper (one-side) Tolerance Limit (eCAI):\t%6.3f\n", $LimitSupCAI;
		printf OUT2 "CAI Average:\t%6.3f\n", $MitjanaCAI;
		printf OUT2 "Standard deviation:\t%6.3f\n", $DesvEstCAI;
		print OUT2 "Number of random sequences: $numSequences\n";
		if ($NORMAL eq 'TRUE') {
			printf OUT2 "Kolmogorov-Smirnov test for the expected RCDI (alpha = 5%):
%6.5f < Critical Value(%6.5f) [NORMALITY]\n", $maxD, $c95;
		}else {
			printf OUT2 "Kolmogorov-Smirnov test for the expected RCDI (alpha = 5%):
%6.5f >= Critical Value(%6.5f) [NOT NORMALITY]\n", $maxD, $c95;
		}
```

```
		printf OUT2 "Chi-Square Goodness-of-Fit test for AA (alpha = 5%): the %3.1f",
$Perc_chi_AAH1;
		print OUT2 "% of sequences fit the AA distribution.\n";

		if ($intGC  == "") {
			printf OUT2 "Chi-Square Goodness-of-Fit test for GC (alpha = 5%): the
%3.1f", $Perc_chi_GCH1;
			print OUT2 "% of sequences fit the GC distribution.\n";
		}
	}
	close OUT2;
}


##################################
# CAI calculation  of real  Random sequences
##################################
#
#############################################################
# FUNCTION 1.b (2) - CAI and E-CAI calculation (real random)
#############################################################
#
sub CalcMitjanaCAIrandom2($fitxer, $tipusGenoma, $hoste, $errorC, $population, $fileCAI,
$numSequences, $long, $method) {
	my ($fitxer, $tipusGenoma, $hoste, $errorC, $population, $fileCAI, $numSequences,
$long) = @_;

	open (OUT1, ">$fileCAI") || die "cannot open $fileCAI\n";
	print OUT1 "CAI\tSEQUENCE\n";

	# Genetic Code
	my @codigenetic_aux = &geneticCode($tipusGenoma);
	my %codigenetic = @codigenetic_aux;
	my @CodonsSTOP = &stopCodons(@codigenetic_aux);
	my @CodonsPerAA_aux = &codonsPerAA(@codigenetic_aux);
	my %CodonsPerAA = @CodonsPerAA_aux;
	my @varAA_aux = &varAA(@CodonsPerAA_aux);
	my %varAA = @varAA_aux;
	my @CodonsUnics = &CodonsUnics(@codigenetic_aux);
	my @CodonsVariables = &CodonsVariables(@codigenetic_aux);
	my $countVariables = &nCodonsVariables(@CodonsVariables);


	# Generates -n randome sequenc with a -l length
	# Variables required: $nombreSEQ, %aminoacidsTotal__, %aminoacidsTotal__SD, $GC_total,
$GC_1s, $GC_2s, $GC_3s
		my $c=0;
		my $totalCAI=0;
		my @totsCAIs = ();
		# while number of new sequences is less than -n
		my @CodonsRandom = (@CodonsUnics, @CodonsVariables);
		print "@CodonsRandom\n";
		while ($c < $numSequences) {
			$c++;
```

```
                        $segNTstring = "";
                        while ( (length($segNTstring)/3) < $long) {
                                $r_number = int(rand($countVariables));
                                $segNTstring .= @CodonsRandom[$r_number];
                        }
                        # CAI calculation and print the sequence into a file
                        my $CAI = &calcularElCAI($segNTstring, $tipusGenoma, $hoste);
                        printf OUT1 "%6.3f\t$segNTstring\n", $CAI;
                        @totsCAIs[$c] = $CAI;
                        $totalCAI += $CAI;
                }
                close OUT1;
                #CAI average calculation
                my $MitjanaCAI = $totalCAI / $numSequences;

                #SD of CAI
                my $DesvEstCAI = 0;
                my $difCAI = 0;
                foreach $CAI(@totsCAIs) {
                        if ($CAI == 0) {
                        }else {
                                $difCAI_ =  ($CAI - $MitjanaCAI)**2;
                                $DesvEstCAI += $difCAI_;
                        }
                }
                $DesvEstCAI = sqrt($DesvEstCAI/$numSequences);
                #Confidence parameters
                $ErrorCAI = &CalcConfidence($numSequences, $errorC, $population);
                my $LimitSupCAI = $MitjanaCAI + ($ErrorCAI*$DesvEstCAI);


        #####
        print "* Number of random sequences:\t$numSequences\n";
        printf "* CAI Average:\t%6.3f\n", $MitjanaCAI;
        printf "* Standard deviation:\t%6.3f\n", $DesvEstCAI;
        printf "* Upper (one-side) Tolerance Limit:\t%6.3f\n", $LimitSupCAI;
        #####
        open (OUT2, ">$fileExpected") || die "cannot open $fileExpected";
        print OUT2 "EXPECTED CAI at "."$errorC"."% confident that contain "."$population"."% of
population\n";
        printf OUT2 "Upper (one-side) Tolerance Limit:\t%6.3f\n", $LimitSupCAI;
        printf OUT2 "CAI Average:\t%6.3f\n", $MitjanaCAI;
        printf OUT2 "Standard deviation:\t%6.3f\n", $DesvEstCAI;
        print OUT2 "Number of random sequences: $numSequences\n";
        close OUT2;
}


###################
# CAI calculation #
###################
#
############################################################
# FUNCTION 2.b - CAI calculation
```

```
############################################################
#
# Parameters required: input file with sequences in fasta format, type of genetic code, file
with codon usage reference table and Output file
# Return CAI value

sub calcularElCAI() {
	my $seq = shift;
	my $tipusGenoma = shift;
	my $hoste = shift;

	$seq =~ tr/actguU/ACTGTT/;
	my %Codo = ('TTT',0, 'TTC',0, 'TTA',0, 'TTG',0, 'CTT',0, 'CTC',0, 'CTA',0, 'CTG',0,
'ATT',0, 'ATC',0, 'ATA',0, 'ATG',0, 'GTT',0, 'GTC',0, 'GTA',0, 'GTG',0, 'TCT',0, 'TCC',0,
'TCA',0,'TCG',0, 'CCT',0, 'CCC',0, 'CCA',0, 'CCG',0, 'ACT',0, 'ACC',0, 'ACA',0, 'ACG',0,
'GCT',0, 'GCC',0,'GCA',0, 'GCG',0, 'TAT',0, 'TAC',0, 'TAA',0, 'TAG',0, 'CAT',0, 'CAC',0,
'CAA',0, 'CAG',0, 'AAT',0,'AAC',0, 'AAA',0, 'AAG',0, 'GAT',0, 'GAC',0, 'GAA',0, 'GAG',0,
'TGT',0, 'TGC',0, 'TGA',0, 'TGG',0,'CGT',0, 'CGC',0, 'CGA',0, 'CGG',0, 'AGT',0, 'AGC',0,
'AGA',0, 'AGG',0, 'GGT',0, 'GGC',0, 'GGA',0,'GGG',0);

	# Genetic Code
	my @codigenetic_aux = &geneticCode($tipusGenoma);
	my %codigenetic = @codigenetic_aux;
	my @CodonsSTOP = &stopCodons(@codigenetic_aux);
	my @CodonsPerAA_aux = &codonsPerAA(@codigenetic_aux);
	my %CodonsPerAA = @CodonsPerAA_aux;
	my @varAA_aux = &varAA(@CodonsPerAA_aux);
	my %varAA = @varAA_aux;
	my @CodonsUnics = &CodonsUnics(@codigenetic_aux);
	my @CodonsVariables = &CodonsVariables(@codigenetic_aux);
	my $countVariables = &nCodonsVariables(@CodonsVariables);

	#Codon usage calculation
	my $seq2 = $seq;
	$seq2 =~ s/(...)/$1 /g;
	my %totalCodo =();
	my %totalAA =();
	my $totalCodons = 0;
	foreach $codo(sort keys(%Codo)) {
		my $num = $seq2 =~ s/$codo /$codo /g;
		$num += 0;
		$totalCodo{$codo} = $num;
		$AA = $codigenetic{$codo};
		$totalAA{$AA} += $num;
		$totalCodons += $num;
	}

	#RSCU calculation
	my %rscu = ();
	foreach $codo(sort keys(%Codo)) {
		$AA = $codigenetic{$codo};
		if ($totalAA{$AA} == 0) {
			$rscu{$codo} = 0;
		}else {
```

```
                        $rscu{$codo} = ($totalCodo{$codo}  / $totalAA{$AA} ) *
$CodonsPerAA{$AA};
                }
        }

        #Codon usage per thousand calculation
        my %us_per_thousand = ();
        foreach $codo(sort keys(%Codo)) {
                if ($totalCodons == 0) {
                        $us_per_thousand{$codo} = 0;
                }
                else {
                        $us_per_thousand{$codo} = ($totalCodo{$codo} / $totalCodons) * 1000;
                }
        }

        # Get the codon usage reference table
        open (INhoste, "<$hoste") || die "cannot open $hoste";
        $linea1 = <INhoste>;
        close INhoste;
        my %ushoste = ();
        my %aahoste = ();
        my %rscuhoste = ();
        my %maxrscuhoste = ();
        my %wihoste = ();
        if ($linea1 =~ /\(/) {
                open (INhoste, "<$hoste") || die "cannot open $hoste";
                while (<INhoste>) {
                        chomp;
                        $lin = $_;
                        $lin =~ s/\t//g;
                        $lin =~ s/\s//g;
                        my @linea = split /\)/, $lin;
                        foreach $l34(@linea) {
                                $l34 =~ /^(\w\w\w)/;
                                $val0 = $1;
                                $val0 =~ tr/actguU/ACTGTT/;
                                $l34 =~ /\((.+)/;
                                $val1 = $1;
                                chomp $val0;
                                chomp $val1;
                                my $AA = $codigenetic{$val0};
                                $ushoste{$val0} = $val1;
                                $aahoste{$AA} += $val1;
                        }
                }
                close INhoste;
        }
        else {
                open (INhoste, "<$hoste") || die "cannot open $hoste";
                while (<INhoste>) {
                        chomp;
                        my @linea = split /\t/, $_;
                        $linea[0] =~ tr/actguU/ACTGTT/;
```

```
                        my $AA = $codigenetic{$linea[0]};
                        $ushoste{$linea[0]} = $linea[1];
                        $aahoste{$AA} += $linea[1];
                }
                close INhoste;
        }

        # Reference RSCU calculation
        foreach $codo(keys(%ushoste)) {
                my $AA = $codigenetic{$codo};

                if ( ($AA ne ".") && ($CodonsPerAA{$AA} > 1) ) {
                        my $codo_dos = $codo;
                        chop $codo_dos;

                        if ($aahoste{$AA} == 0) {
                                $rscuhoste{$codo} = 0;
                        }else {
                                #print
"$codo\t$ushoste{$codo}\t$aahoste{$AA}\t$CodonsPerAA{$AA}\n";
                                $rscuhoste{$codo} = ($ushoste{$codo} / $aahoste{$AA}) *
$CodonsPerAA{$AA};
                        }

                        if ($rscuhoste{$codo} > $maxrscuhoste{$AA}{$codo_dos}) {
                                $maxrscuhoste{$AA}{$codo_dos} = $rscuhoste{$codo};
                        }
                }
        }

        # Reference weigth of each codon calculation
        foreach $codo(keys(%ushoste)) {
                my $AA = $codigenetic{$codo};

                if ( ($AA ne ".") && ($CodonsPerAA{$AA} > 1) ) {
                        my $codo_dos = $codo;
                        chop $codo_dos;

                        if ($rscuhoste{$codo} == 0) {
                                #$rscuhoste{$codo}= 0.5;
                                die "#ERROR - $AA - $CodonsPerAA{$AA} - $codo  = 0# There is
insufficient information for computing CAI usign this reference table";
                        }

                        if ($maxrscuhoste{$AA}{$codo_dos} == 0) {
                                $wihoste{$codo} = 0;
                        }else {
                                #print "##
$codo\t$rscuhoste{$codo}\t$maxrscuhoste{$AA}{$codo_dos}\n";
                                $wihoste{$codo} =
$rscuhoste{$codo}/$maxrscuhoste{$AA}{$codo_dos};
                        }
                }
        }
```

```
	# CAI calculation
	my $n = 0;
	my $wi = 0;
	foreach $codo(@CodonsVariables) {
		#print "$codo $wihoste{$codo}\n";
		$wi += $totalCodo{$codo} * log ($wihoste{$codo});
		$n +=$totalCodo{$codo};
	}

	my $CAI = 0;
	if ($n == 0) {
		$CAI = 0;
	}else {
		$CAI = $wi / $n;
		$CAI = exp($CAI);
	}
	return $CAI;
}


sub calcularElRCDI() {
	my $seq = shift;
	my $tipusGenoma = shift;
	my $hoste = shift;

	$seq =~ tr/actguU/ACTGTT/;
	my %Codo = ('TTT',0, 'TTC',0, 'TTA',0, 'TTG',0, 'CTT',0, 'CTC',0, 'CTA',0, 'CTG',0,
'ATT',0, 'ATC',0, 'ATA',0, 'ATG',0, 'GTT',0, 'GTC',0, 'GTA',0, 'GTG',0, 'TCT',0, 'TCC',0,
'TCA',0,'TCG',0, 'CCT',0, 'CCC',0, 'CCA',0, 'CCG',0, 'ACT',0, 'ACC',0, 'ACA',0, 'ACG',0,
'GCT',0, 'GCC',0,'GCA',0, 'GCG',0, 'TAT',0, 'TAC',0, 'TAA',0, 'TAG',0, 'CAT',0, 'CAC',0,
'CAA',0, 'CAG',0, 'AAT',0,'AAC',0, 'AAA',0, 'AAG',0, 'GAT',0, 'GAC',0, 'GAA',0, 'GAG',0,
'TGT',0, 'TGC',0, 'TGA',0, 'TGG',0,'CGT',0, 'CGC',0, 'CGA',0, 'CGG',0, 'AGT',0, 'AGC',0,
'AGA',0, 'AGG',0, 'GGT',0, 'GGC',0, 'GGA',0,'GGG',0);

	# Genetic Code
	my @codigenetic_aux = &geneticCode($tipusGenoma);
	my %codigenetic = @codigenetic_aux;
	my @CodonsSTOP = &stopCodons(@codigenetic_aux);
	my @CodonsPerAA_aux = &codonsPerAA(@codigenetic_aux);
	my %CodonsPerAA = @CodonsPerAA_aux;
	my @varAA_aux = &varAA(@CodonsPerAA_aux);
	my %varAA = @varAA_aux;
	my @CodonsUnics = &CodonsUnics(@codigenetic_aux);
	my @CodonsVariables = &CodonsVariables(@codigenetic_aux);
	my $countVariables = &nCodonsVariables(@CodonsVariables);

	#Codon usage calculation test
	my $seq2 = $seq;
	$seq2 =~ s/(...)/$1 /g;
	my %totalCodo =();
	my %totalAA =();
	my $totalCodons = 0;
	foreach $codo(sort keys(%Codo)) {
```

```
        my $num = $seq2 =~ s/$codo /$codo /g;
        $num += 0;
        $totalCodo{$codo} = $num;
        $AA = $codigenetic{$codo};
        $totalAA{$AA} += $num;
        $totalCodons += $num;
}
my $N = $totalCodons;

# Get the codon usage reference table
open (INhoste, "<$hoste") || die "cannot open $hoste";
$linea1 = <INhoste>;
close INhoste;
my %ushoste = ();
my %aahoste = ();
my %rscuhoste = ();
my %maxrscuhoste = ();
my %wihoste = ();
if ($linea1 =~ /\(/) {
        open (INhoste, "<$hoste") || die "cannot open $hoste";
        while (<INhoste>) {
                chomp;
                $lin = $_;
                $lin =~ s/\t//g;
                $lin =~ s/\s//g;
                my @linea = split /\)/, $lin;
                foreach $l34(@linea) {
                        $l34 =~ /^(\w\w\w)/;
                        $val0 = $1;
                        $val0 =~ tr/actguU/ACTGTT/;
                        $l34 =~ /\((.+)/;
                        $val1 = $1;
                        chomp $val0;
                        chomp $val1;
                        my $AA = $codigenetic{$val0};
                        $ushoste{$val0} = $val1;
                        $aahoste{$AA} += $val1;
                }
        }
        close INhoste;
}
else {
        open (INhoste, "<$hoste") || die "cannot open $hoste";
        while (<INhoste>) {
                chomp;
                my @linea = split /\t/, $_;
                $linea[0] =~ tr/actguU/ACTGTT/;
                my $AA = $codigenetic{$linea[0]};
                $ushoste{$linea[0]} = $linea[1];
                $aahoste{$AA} += $linea[1];
        }
        close INhoste;
}
```

```
	# Calculation of RCDI
	my $RCDI = 0;
	foreach $codo(sort keys(%codigenetic)) {
		my $AA = $codigenetic{$codo};


		if ($AA eq ".") {
		}elsif ( ($ushoste{$codo} == 0) || ($totalAA{$AA} == 0) || ($aahoste{$AA}) == 0
) {
			$RCDI +=0;
		}else {
			my $CiFa = $totalCodo{$codo} / $totalAA{$AA};
			my $CiFh = $ushoste{$codo} / $aahoste{$AA};
			my $Ni = $totalCodo{$codo};

			my $rcdi = ( $CiFa / $CiFh ) * $Ni;
			$RCDI += $rcdi;
		}
	}

	$RCDI /= $N;

	return $RCDI;
}

#####################
#  CAI calculation  #
#####################
#
############################################################
# FUNCTION 1.a - CAI calculation
############################################################
#
sub sequencies($fitxer, $tipusGenoma, $hoste, $fileCAI ) {
	my ($fitxer, $tipusGenoma, $hoste, $fileCAI ) = @_;

	my %sequences = ();
	my @seq_ = ();
	my $seq = '';
	my $fila = 1;
	# Get the fasta sequences from a file
	open (IN, "<$fitxer") || die "cannot open $fitxer\n";
		while (<IN>) {
			chomp;
			if (/^>/) {
				if ($file == 2) {
					$sequences{$nom} = $seq;
				}
				$nom = $_;
				push @seq_, $_;
				$seq = '';
			}
			else {
				$seq .= $_;
```

```
                        }
                        $file = 2;
                }
                $sequences{$nom} = $seq;
        close IN;
        # CAI calculation and print CAI into a file
        open (OUT1, ">$fileCAI") || die "cannot open $fileCAI\n";
                print OUT1 "NAME\tCAI\n";
                foreach $nom(@seq_) {
                        $segNTstring = $sequences{$nom};
                        $segNTstring =~ tr/actguU/ACTGTT/;
                        $missatge = 'ok';
                        ($missatge, $opmissatge) = &errors($nom, $segNTstring, $tipusGenoma);
                        if ($missatge eq 'ok') {
                                my $CAI = &calcularElCAI($segNTstring, $tipusGenoma, $hoste);
                                printf OUT1 "$nom\t%6.3f\n", $CAI;
                        }else {
                                if ($opmissatge == 1) {
                                        print OUT1 "$nom\t#Error# The length of this sequence is
not divided by three.\n";
                                }elsif ($opmissatge == 2) {
                                        my $CAI = &calcularElCAI($segNTstring, $tipusGenoma,
$hoste);
                                        printf OUT1 "$nom\t%6.3f\t#Warning# This sequence has
more than one stop codon\n", $CAI;
                                }
                        }
                }
        close OUT1;
}

############################################################
# FUNCTION 1.b - RCDI calculation
############################################################
#
sub sequencies_rcdi($fitxer, $tipusGenoma, $hoste, $fileCAI ) {
        my ($fitxer, $tipusGenoma, $hoste, $fileCAI ) = @_;

        my %sequences = ();
        my @seq_ = ();
        my $seq = '';
        my $fila = 1;
        # Get the fasta sequences from a file
        open (IN, "<$fitxer") || die "cannot open $fitxer\n";
                while (<IN>) {
                        chomp;
                        if (/^>/) {
                                if ($file == 2) {
                                        $sequences{$nom} = $seq;
                                }
                                $nom = $_;
                                push @seq_, $_;
                                $seq = '';
                        }
```

```
                        else {
                                $seq .= $_;
                        }
                        $file = 2;
                }
                $sequences{$nom} = $seq;
        close IN;
        # CAI calculation and print CAI into a file
        open (OUT1, ">$fileCAI") || die "cannot open $fileCAI\n";
                print OUT1 "NAME\tRCDI\n";
                foreach $nom(@seq_) {
                        $segNTstring = $sequences{$nom};
                        $segNTstring =~ tr/actguU/ACTGTT/;
                        $missatge = 'ok';
                        ($missatge, $opmissatge) = &errors($nom, $segNTstring, $tipusGenoma);
                        if ($missatge eq 'ok') {
                                my $CAI = &calcularElRCDI($segNTstring, $tipusGenoma, $hoste);
                                printf OUT1 "$nom\t%6.3f\n", $CAI;
                        }else {
                                if ($opmissatge == 1) {
                                        print OUT1 "$nom\t#Error# The length of this sequence is
not divided by three.\n";
                                }elsif ($opmissatge == 2) {
                                        my $CAI = &calcularElRCDI($segNTstring, $tipusGenoma,
$hoste);
                                        printf OUT1 "$nom\t%6.3f\t#Warning# This sequence has
more than one stop codon\n", $CAI;
                                }
                        }
                }
        close OUT1;

}


##################
# Error Messages #
##################
#
############################################################
# FUNCTION 2.a - Error messages
############################################################
#
sub errors() {
        my $name = shift;
        my $sequence = shift;
        my $tipusGenoma = shift;

        # Genetic Code
        my @codigenetic_aux = &geneticCode($tipusGenoma);
        my %codigenetic = @codigenetic_aux;
        my @CodonsSTOP = &stopCodons(@codigenetic_aux);

        #### Warning: sequence with more than one Stop codon
```

```
	$num_stop = 0;
	$sequenceSTOPS = $sequence;
	$sequenceSTOPS =~ s/(...)/$1 /g;
	foreach $_codostop(@CodonsSTOP) {
		$num_stop += $sequenceSTOPS =~ s/$_codostop/$_codostop/g;
	}

	if ($num_stop > 1) {
		$missatge = "#Warning# Sequence $name has more than one stop codon. ";
		$opmissatge = 2;
	}

	##### Error: sequence not divided by 3
	my $longitud = length($sequence);
	if ( ($longitud%3) == 0 ) {

	}
	else {
		$missatge = "#Error# The length of the sequence \'$name\' is not divided by
three. ";
		$opmissatge = 1;
	}

	return ($missatge, $opmissatge);
}

########################
# Confidence coeficient #
########################
#
############################################################
# FUNCTION 2.e - Calculation of confidence level
############################################################
#
sub CalcConfidence () {
	my $n_ =shift;
	my $conf_ = shift;
	my $pop_ = shift;
	# ZP
	my $zp = '';
	if ($conf_ == 90) {
		$zp = 1.281552;
	}
	elsif ($conf_ == 95) {
		$zp = 1.644853;
	}
	elsif ($conf_ == 99) {
		$zp = 2.326347;
	}

	# ZG
	my $zg = '';
	if ($pop_ == 90) {
		$zg = 1.281552;
```

```
	}
	elsif ($pop_ == 95) {
		$zg = 1.644853;
	}
	elsif ($pop_ == 99) {
		$zg = 2.326347;
	}

	#a and b
	my $a_ = 1 - ( ($zg**2)  / ( 2*($n_-1) ) );
	my $b_ = ( ($zp**2) ) - ($zg**2)  / $n_ ;

	#K1 (ErrorCAI_);
	my $ErrorCAI_ = ( $zp + sqrt( ($zp**2) - ($a_*$b_) ) ) / $a_;
	return $ErrorCAI_;
}

####################################
# Normality test Kolmogorov-Smirnov #
# Return NORMAL (true, false)       #
####################################
#
############################################################
# FUNCTION 2.g - KS-test
############################################################
#
sub KStest() {
	my ($MitjanaCAI, $DesvEstCAI,  $numSequences, $blanc, @dataCAIs) = @_;
	my $NORMAL = "FALSE";
	# Sort de data
	my @dataCAIs = sort (@dataCAIs);

	# Critical value
	#95
	my $c95 = 1.36 / sqrt($numSequences);

	# Normal distribution
	my @probabNormDist = ();
	$i=1;
	$maxD = "";
	foreach $dcai(@dataCAIs) {
		if ($dcai eq '') {
		}else {
			# g(Z)
			my $probab = &normalDist($dcai, $MitjanaCAI, $DesvEstCAI);
			#print "$dcai, $MitjanaCAI, $DesvEstCAI --> $probab\n";

			if ($probab ne "") {
				# g(Z)-f(Z)
				$valD = sqrt((	($probab-(($i-1)/$numSequences))
)**2);
				# Find the max D value

				if ($valD > $maxD) {
```

```
                                $maxD = $valD;
                                #print "$maxD\n";
                        }

                        $i++;
                }
        }
    }

    # Normality test
    if ($maxD < $c95) {
        $NORMAL = "TRUE";
    }else {
        $NORMAL = "FALSE";
    }
    return ($NORMAL, $maxD, $c95);
}
###################################
# Normal distribution probability #
###################################
#
#############################################################
# FUNCTION 3.a - Normal distribution
#############################################################
#
sub normalDist {
    my ($x, $mean, $stdev) = @_;
    my %NormalTable = ('-4.00', '0.00003', '-4.01', '0.00003', '-4.02', '0.00003', '-4.03',
'0.00003', '-4.04', '0.00003', '-4.05', '0.00003', '-4.06', '0.00002', '-4.07', '0.00002', '-
4.08', '0.00002', '-4.09', '0.00002', '-3.90', '0.00005', '-3.91', '0.00005', '-3.92',
'0.00004', '-3.93', '0.00004', '-3.94', '0.00004', '-3.95', '0.00004', '-3.96', '0.00004', '-
3.97', '0.00004', '-3.98', '0.00003', '-3.99', '0.00003', '-3.80', '0.00007', '-3.81',
'0.00007', '-3.82', '0.00007', '-3.83', '0.00006', '-3.84', '0.00006', '-3.85', '0.00006', '-
3.86', '0.00006', '-3.87', '0.00005', '-3.88', '0.00005', '-3.89', '0.00005', '-3.70',
'0.00011', '-3.71', '0.00010', '-3.72', '0.00010', '-3.73', '0.00010', '-3.74', '0.00009', '-
3.75', '0.00009', '-3.76', '0.00008', '-3.77', '0.00008', '-3.78', '0.00008', '-3.79',
'0.00008', '-3.60', '0.00016', '-3.61', '0.00015', '-3.62', '0.00015', '-3.63', '0.00014', '-
3.64', '0.00014', '-3.65', '0.00013', '-3.66', '0.00013', '-3.67', '0.00012', '-3.68',
'0.00012', '-3.69', '0.00011', '-3.50', '0.00023', '-3.51', '0.00022', '-3.52', '0.00022', '-
3.53', '0.00021', '-3.54', '0.00020', '-3.55', '0.00019', '-3.56', '0.00019', '-3.57',
'0.00018', '-3.58', '0.00017', '-3.59', '0.00017', '-3.40', '0.00034', '-3.41', '0.00032', '-
3.42', '0.00031', '-3.43', '0.00030', '-3.44', '0.00029', '-3.45', '0.00028', '-3.46',
'0.00027', '-3.47', '0.00026', '-3.48', '0.00025', '-3.49', '0.00024', '-3.30', '0.00048', '-
3.31', '0.00047', '-3.32', '0.00045', '-3.33', '0.00043', '-3.34', '0.00042', '-3.35',
'0.00040', '-3.36', '0.00039', '-3.37', '0.00038', '-3.38', '0.00036', '-3.39', '0.00035', '-
3.20', '0.00069', '-3.21', '0.00066', '-3.22', '0.00064', '-3.23', '0.00062', '-3.24',
'0.00060', '-3.25', '0.00058', '-3.26', '0.00056', '-3.27', '0.00054', '-3.28', '0.00052', '-
3.29', '0.00050', '-3.10', '0.00097', '-3.11', '0.00094', '-3.12', '0.00090', '-3.13',
'0.00087', '-3.14', '0.00084', '-3.15', '0.00082', '-3.16', '0.00079', '-3.17', '0.00076', '-
3.18', '0.00074', '-3.19', '0.00071', '-3.00', '0.00135', '-3.01', '0.00131', '-3.02',
'0.00126', '-3.03', '0.00122', '-3.04', '0.00118', '-3.05', '0.00114', '-3.06', '0.00111', '-
3.07', '0.00107', '-3.08', '0.00103', '-3.09', '0.00100', '-2.90', '0.00187', '-2.91',
'0.00181', '-2.92', '0.00175', '-2.93', '0.00169', '-2.94', '0.00164', '-2.95', '0.00159', '-
2.96', '0.00154', '-2.97', '0.00149', '-2.98', '0.00144', '-2.99', '0.00139', '-2.80',
```

'0.00256', '-2.81', '0.00248', '-2.82', '0.00240', '-2.83', '0.00233', '-2.84', '0.00226', '-
2.85', '0.00219', '-2.86', '0.00212', '-2.87', '0.00205', '-2.88', '0.00199', '-2.89',
'0.00193', '-2.70', '0.00347', '-2.71', '0.00336', '-2.72', '0.00326', '-2.73', '0.00317', '-
2.74', '0.00307', '-2.75', '0.00298', '-2.76', '0.00289', '-2.77', '0.00280', '-2.78',
'0.00272', '-2.79', '0.00264', '-2.60', '0.00466', '-2.61', '0.00453', '-2.62', '0.00440', '-
2.63', '0.00427', '-2.64', '0.00415', '-2.65', '0.00402', '-2.66', '0.00391', '-2.67',
'0.00379', '-2.68', '0.00368', '-2.69', '0.00357', '-2.50', '0.00621', '-2.51', '0.00604', '-
2.52', '0.00587', '-2.53', '0.00570', '-2.54', '0.00554', '-2.55', '0.00539', '-2.56',
'0.00523', '-2.57', '0.00508', '-2.58', '0.00494', '-2.59', '0.00480', '-2.40', '0.00820', '-
2.41', '0.00798', '-2.42', '0.00776', '-2.43', '0.00755', '-2.44', '0.00734', '-2.45',
'0.00714', '-2.46', '0.00695', '-2.47', '0.00676', '-2.48', '0.00657', '-2.49', '0.00639', '-
2.30', '0.01072', '-2.31', '0.01044', '-2.32', '0.01017', '-2.33', '0.00990', '-2.34',
'0.00964', '-2.35', '0.00939', '-2.36', '0.00914', '-2.37', '0.00889', '-2.38', '0.00866', '-
2.39', '0.00842', '-2.20', '0.01390', '-2.21', '0.01355', '-2.22', '0.01321', '-2.23',
'0.01287', '-2.24', '0.01255', '-2.25', '0.01222', '-2.26', '0.01191', '-2.27', '0.01160', '-
2.28', '0.01130', '-2.29', '0.01101', '-2.10', '0.01786', '-2.11', '0.01743', '-2.12',
'0.01700', '-2.13', '0.01659', '-2.14', '0.01618', '-2.15', '0.01578', '-2.16', '0.01539', '-
2.17', '0.01500', '-2.18', '0.01463', '-2.19', '0.01426', '-2.00', '0.02275', '-2.01',
'0.02222', '-2.02', '0.02169', '-2.03', '0.02118', '-2.04', '0.02067', '-2.05', '0.02018', '-
2.06', '0.01970', '-2.07', '0.01923', '-2.08', '0.01876', '-2.09', '0.01831', '-1.90',
'0.02872', '-1.91', '0.02807', '-1.92', '0.02743', '-1.93', '0.02680', '-1.94', '0.02619', '-
1.95', '0.02559', '-1.96', '0.02500', '-1.97', '0.02442', '-1.98', '0.02385', '-1.99',
'0.02330', '-1.80', '0.03593', '-1.81', '0.03515', '-1.82', '0.03438', '-1.83', '0.03362', '-
1.84', '0.03288', '-1.85', '0.03216', '-1.86', '0.03144', '-1.87', '0.03074', '-1.88',
'0.03005', '-1.89', '0.02938', '-1.70', '0.04456', '-1.71', '0.04363', '-1.72', '0.04272', '-
1.73', '0.04181', '-1.74', '0.04093', '-1.75', '0.04006', '-1.76', '0.03920', '-1.77',
'0.03836', '-1.78', '0.03754', '-1.79', '0.03673', '-1.60', '0.05480', '-1.61', '0.05370', '-
1.62', '0.05262', '-1.63', '0.05155', '-1.64', '0.05050', '-1.65', '0.04947', '-1.66',
'0.04846', '-1.67', '0.04746', '-1.68', '0.04648', '-1.69', '0.04551', '-1.50', '0.06681', '-
1.51', '0.06552', '-1.52', '0.06425', '-1.53', '0.06301', '-1.54', '0.06178', '-1.55',
'0.06057', '-1.56', '0.05938', '-1.57', '0.05821', '-1.58', '0.05705', '-1.59', '0.05592', '-
1.40', '0.08076', '-1.41', '0.07927', '-1.42', '0.07780', '-1.43', '0.07636', '-1.44',
'0.07493', '-1.45', '0.07353', '-1.46', '0.07214', '-1.47', '0.07078', '-1.48', '0.06944', '-
1.49', '0.06811', '-1.30', '0.09680', '-1.31', '0.09510', '-1.32', '0.09342', '-1.33',
'0.09176', '-1.34', '0.09012', '-1.35', '0.08851', '-1.36', '0.08691', '-1.37', '0.08534', '-
1.38', '0.08379', '-1.39', '0.08226', '-1.20', '0.11507', '-1.21', '0.11314', '-1.22',
'0.11123', '-1.23', '0.10935', '-1.24', '0.10749', '-1.25', '0.10565', '-1.26', '0.10383', '-
1.27', '0.10204', '-1.28', '0.10027', '-1.29', '0.09852', '-1.10', '0.13566', '-1.11',
'0.13350', '-1.12', '0.13136', '-1.13', '0.12924', '-1.14', '0.12714', '-1.15', '0.12507', '-
1.16', '0.12302', '-1.17', '0.12100', '-1.18', '0.11900', '-1.19', '0.11702', '-1.00',
'0.15865', '-1.01', '0.15625', '-1.02', '0.15386', '-1.03', '0.15150', '-1.04', '0.14917', '-
1.05', '0.14686', '-1.06', '0.14457', '-1.07', '0.14231', '-1.08', '0.14007', '-1.09',
'0.13786', '-0.90', '0.18406', '-0.91', '0.18141', '-0.92', '0.17878', '-0.93', '0.17618', '-
0.94', '0.17361', '-0.95', '0.17105', '-0.96', '0.16853', '-0.97', '0.16602', '-0.98',
'0.16354', '-0.99', '0.16109', '-0.80', '0.21185', '-0.81', '0.20897', '-0.82', '0.20611', '-
0.83', '0.20327', '-0.84', '0.20045', '-0.85', '0.19766', '-0.86', '0.19489', '-0.87',
'0.19215', '-0.88', '0.18943', '-0.89', '0.18673', '-0.70', '0.24196', '-0.71', '0.23885', '-
0.72', '0.23576', '-0.73', '0.23269', '-0.74', '0.22965', '-0.75', '0.22663', '-0.76',
'0.22363', '-0.77', '0.22065', '-0.78', '0.21769', '-0.79', '0.21476', '-0.60', '0.27425', '-
0.61', '0.27093', '-0.62', '0.26763', '-0.63', '0.26434', '-0.64', '0.26108', '-0.65',
'0.25784', '-0.66', '0.25462', '-0.67', '0.25143', '-0.68', '0.24825', '-0.69', '0.24509', '-
0.50', '0.30853', '-0.51', '0.30502', '-0.52', '0.30153', '-0.53', '0.29805', '-0.54',
'0.29460', '-0.55', '0.29116', '-0.56', '0.28774', '-0.57', '0.28434', '-0.58', '0.28095', '-
0.59', '0.27759', '-0.40', '0.34457', '-0.41', '0.34090', '-0.42', '0.33724', '-0.43',

'0.33359', '-0.44', '0.32997', '-0.45', '0.32635', '-0.46', '0.32276', '-0.47', '0.31917', '-0.48', '0.31561', '-0.49', '0.31206', '-0.30', '0.38209', '-0.31', '0.37828', '-0.32', '0.37448', '-0.33', '0.37070', '-0.34', '0.36692', '-0.35', '0.36317', '-0.36', '0.35942', '-0.37', '0.35569', '-0.38', '0.35197', '-0.39', '0.34826', '-0.20', '0.42074', '-0.21', '0.41683', '-0.22', '0.41293', '-0.23', '0.40904', '-0.24', '0.40516', '-0.25', '0.40129', '-0.26', '0.39743', '-0.27', '0.39358', '-0.28', '0.38974', '-0.29', '0.38590', '-0.10', '0.46017', '-0.11', '0.45620', '-0.12', '0.45224', '-0.13', '0.44828', '-0.14', '0.44433', '-0.15', '0.44038', '-0.16', '0.43644', '-0.17', '0.43250', '-0.18', '0.42857', '-0.19', '0.42465', '-0.00', '0.50000', '-0.01', '0.49601', '-0.02', '0.49202', '-0.03', '0.48803', '-0.04', '0.48404', '-0.05', '0.48006', '-0.06', '0.47607', '-0.07', '0.47209', '-0.08', '0.46811', '-0.09', '0.46414', '4.00', '0.99997', '4.01', '0.99997', '4.02', '0.99997', '4.03', '0.99997', '4.04', '0.99997', '4.05', '0.99997', '4.06', '0.99998', '4.07', '0.99998', '4.08', '0.99998', '4.09', '0.99998', '3.90', '0.99995', '3.91', '0.99995', '3.92', '0.99996', '3.93', '0.99996', '3.94', '0.99996', '3.95', '0.99996', '3.96', '0.99996', '3.97', '0.99996', '3.98', '0.99997', '3.99', '0.99997', '3.80', '0.99993', '3.81', '0.99993', '3.82', '0.99993', '3.83', '0.99994', '3.84', '0.99994', '3.85', '0.99994', '3.86', '0.99994', '3.87', '0.99995', '3.88', '0.99995', '3.89', '0.99995', '3.70', '0.99989', '3.71', '0.99990', '3.72', '0.99990', '3.73', '0.99990', '3.74', '0.99991', '3.75', '0.99991', '3.76', '0.99992', '3.77', '0.99992', '3.78', '0.99992', '3.79', '0.99992', '3.60', '0.99984', '3.61', '0.99985', '3.62', '0.99985', '3.63', '0.99986', '3.64', '0.99986', '3.65', '0.99987', '3.66', '0.99987', '3.67', '0.99988', '3.68', '0.99988', '3.69', '0.99989', '3.50', '0.99977', '3.51', '0.99978', '3.52', '0.99978', '3.53', '0.99979', '3.54', '0.99980', '3.55', '0.99981', '3.56', '0.99981', '3.57', '0.99982', '3.58', '0.99983', '3.59', '0.99983', '3.40', '0.99966', '3.41', '0.99968', '3.42', '0.99969', '3.43', '0.99970', '3.44', '0.99971', '3.45', '0.99972', '3.46', '0.99973', '3.47', '0.99974', '3.48', '0.99975', '3.49', '0.99976', '3.30', '0.99952', '3.31', '0.99953', '3.32', '0.99955', '3.33', '0.99957', '3.34', '0.99958', '3.35', '0.99960', '3.36', '0.99961', '3.37', '0.99962', '3.38', '0.99964', '3.39', '0.99965', '3.20', '0.99931', '3.21', '0.99934', '3.22', '0.99936', '3.23', '0.99938', '3.24', '0.99940', '3.25', '0.99942', '3.26', '0.99944', '3.27', '0.99946', '3.28', '0.99948', '3.29', '0.99950', '3.10', '0.99903', '3.11', '0.99906', '3.12', '0.99910', '3.13', '0.99913', '3.14', '0.99916', '3.15', '0.99918', '3.16', '0.99921', '3.17', '0.99924', '3.18', '0.99926', '3.19', '0.99929', '3.00', '0.99865', '3.01', '0.99869', '3.02', '0.99874', '3.03', '0.99878', '3.04', '0.99882', '3.05', '0.99886', '3.06', '0.99889', '3.07', '0.99893', '3.08', '0.99897', '3.09', '0.99900', '2.90', '0.99813', '2.91', '0.99819', '2.92', '0.99825', '2.93', '0.99831', '2.94', '0.99836', '2.95', '0.99841', '2.96', '0.99846', '2.97', '0.99851', '2.98', '0.99856', '2.99', '0.99861', '2.80', '0.99744', '2.81', '0.99752', '2.82', '0.99760', '2.83', '0.99767', '2.84', '0.99774', '2.85', '0.99781', '2.86', '0.99788', '2.87', '0.99795', '2.88', '0.99801', '2.89', '0.99807', '2.70', '0.99653', '2.71', '0.99664', '2.72', '0.99674', '2.73', '0.99683', '2.74', '0.99693', '2.75', '0.99702', '2.76', '0.99711', '2.77', '0.99720', '2.78', '0.99728', '2.79', '0.99736', '2.60', '0.99534', '2.61', '0.99547', '2.62', '0.99560', '2.63', '0.99573', '2.64', '0.99585', '2.65', '0.99598', '2.66', '0.99609', '2.67', '0.99621', '2.68', '0.99632', '2.69', '0.99643', '2.50', '0.99379', '2.51', '0.99396', '2.52', '0.99413', '2.53', '0.99430', '2.54', '0.99446', '2.55', '0.99461', '2.56', '0.99477', '2.57', '0.99492', '2.58', '0.99506', '2.59', '0.99520', '2.40', '0.99180', '2.41', '0.99202', '2.42', '0.99224', '2.43', '0.99245', '2.44', '0.99266', '2.45', '0.99286', '2.46', '0.99305', '2.47', '0.99324', '2.48', '0.99343', '2.49', '0.99361', '2.30', '0.98928', '2.31', '0.98956', '2.32', '0.98983', '2.33', '0.99010', '2.34', '0.99036', '2.35', '0.99061', '2.36', '0.99086', '2.37', '0.99111', '2.38', '0.99134', '2.39', '0.99158', '2.20', '0.98610', '2.21', '0.98645', '2.22', '0.98679', '2.23', '0.98713', '2.24', '0.98745', '2.25', '0.98778', '2.26', '0.98809', '2.27', '0.98840', '2.28', '0.98870', '2.29', '0.98899', '2.10', '0.98214', '2.11', '0.98257', '2.12', '0.98300', '2.13', '0.98341', '2.14', '0.98382', '2.15', '0.98422', '2.16', '0.98461', '2.17', '0.98500', '2.18', '0.98537', '2.19', '0.98574', '2.00', '0.97725', '2.01', '0.97778', '2.02', '0.97831', '2.03', '0.97882', '2.04', '0.97933', '2.05', '0.97982', '2.06', '0.98030', '2.07', '0.98077', '2.08', '0.98124', '2.09', '0.98169', '1.90', '0.97128', '1.91', '0.97193', '1.92', '0.97257', '1.93', '0.97320', '1.94', '0.97381', '1.95', '0.97441', '1.96', '0.97500', '1.97', '0.97558',

```
'1.98', '0.97615', '1.99', '0.97670', '1.80', '0.96407', '1.81', '0.96485', '1.82', '0.96562',
'1.83', '0.96638', '1.84', '0.96712', '1.85', '0.96784', '1.86', '0.96856', '1.87', '0.96926',
'1.88', '0.96995', '1.89', '0.97062', '1.70', '0.95544', '1.71', '0.95637', '1.72', '0.95728',
'1.73', '0.95819', '1.74', '0.95907', '1.75', '0.95994', '1.76', '0.96080', '1.77', '0.96164',
'1.78', '0.96246', '1.79', '0.96327', '1.60', '0.94520', '1.61', '0.94630', '1.62', '0.94738',
'1.63', '0.94845', '1.64', '0.94950', '1.65', '0.95053', '1.66', '0.95154', '1.67', '0.95254',
'1.68', '0.95352', '1.69', '0.95449', '1.50', '0.93319', '1.51', '0.93448', '1.52', '0.93575',
'1.53', '0.93699', '1.54', '0.93822', '1.55', '0.93943', '1.56', '0.94062', '1.57', '0.94179',
'1.58', '0.94295', '1.59', '0.94408', '1.40', '0.91924', '1.41', '0.92073', '1.42', '0.92220',
'1.43', '0.92364', '1.44', '0.92507', '1.45', '0.92647', '1.46', '0.92786', '1.47', '0.92922',
'1.48', '0.93056', '1.49', '0.93189', '1.30', '0.90320', '1.31', '0.90490', '1.32', '0.90658',
'1.33', '0.90824', '1.34', '0.90988', '1.35', '0.91149', '1.36', '0.91309', '1.37', '0.91466',
'1.38', '0.91621', '1.39', '0.91774', '1.20', '0.88493', '1.21', '0.88686', '1.22', '0.88877',
'1.23', '0.89065', '1.24', '0.89251', '1.25', '0.89435', '1.26', '0.89617', '1.27', '0.89796',
'1.28', '0.89973', '1.29', '0.90148 ', '1.10', '0.86434', '1.11', '0.86650', '1.12',
'0.86864', '1.13', '0.87076', '1.14', '0.87286', '1.15', '0.87493', '1.16', '0.87698', '1.17',
'0.87900', '1.18', '0.88100', '1.19', '0.88298', '1.00', '0.84135', '1.01', '0.84375', '1.02',
'0.84614', '1.03', '0.84850', '1.04', '0.85083', '1.05', '0.85314', '1.06', '0.85543', '1.07',
'0.85769', '1.08', '0.85993', '1.09', '0.86214', '0.90', '0.81594', '0.91', '0.81859', '0.92',
'0.82122', '0.93', '0.82382', '0.94', '0.82639', '0.95', '0.82895', '0.96', '0.83147', '0.97',
'0.83398', '0.98', '0.83646', '0.99', '0.83891', '0.80', '0.78815', '0.81', '0.79103', '0.82',
'0.79389', '0.83', '0.79673', '0.84', '0.79955', '0.85', '0.80234', '0.86', '0.80511', '0.87',
'0.80785', '0.88', '0.81057', '0.89', '0.81327', '0.70', '0.75804', '0.71', '0.76115', '0.72',
'0.76424', '0.73', '0.76731', '0.74', '0.77035', '0.75', '0.77337', '0.76', '0.77637', '0.77',
'0.77935', '0.78', '0.78231', '0.79', '0.78524', '0.60', '0.72575', '0.61', '0.72907', '0.62',
'0.73237', '0.63', '0.73566', '0.64', '0.73892', '0.65', '0.74216', '0.66', '0.74538', '0.67',
'0.74857', '0.68', '0.75175', '0.69', '0.75491', '0.50', '0.69147', '0.51', '0.69498', '0.52',
'0.69847', '0.53', '0.70195', '0.54', '0.70540', '0.55', '0.70884', '0.56', '0.71226', '0.57',
'0.71566', '0.58', '0.71905', '0.59', '0.72241', '0.40', '0.65543', '0.41', '0.65910', '0.42',
'0.66276', '0.43', '0.66641', '0.44', '0.67003', '0.45', '0.67365', '0.46', '0.67724', '0.47',
'0.68083', '0.48', '0.68439', '0.49', '0.68794', '0.30', '0.61791', '0.31', '0.62172', '0.32',
'0.62552', '0.33', '0.62930', '0.34', '0.63308', '0.35', '0.63683', '0.36', '0.64058', '0.37',
'0.64431', '0.38', '0.64803', '0.39', '0.65174', '0.20', '0.57926', '0.21', '0.58317', '0.22',
'0.58707', '0.23', '0.59096', '0.24', '0.59484', '0.25', '0.59871', '0.26', '0.60257', '0.27',
'0.60642', '0.28', '0.61026', '0.29', '0.61410', '0.10', '0.53983', '0.11', '0.54380', '0.12',
'0.54776', '0.13', '0.55172', '0.14', '0.55567', '0.15', '0.55962', '0.16', '0.56356', '0.17',
'0.56750', '0.18', '0.57143', '0.19', '0.57535', '0.00', '0.50000', '0.01', '0.50399', '0.02',
'0.50798', '0.03', '0.51197', '0.04', '0.51596', '0.05', '0.51994', '0.06', '0.52393', '0.07',
'0.52791', '0.08', '0.53189', '0.09', '0.53586');
        my $prob = "";
        my $nomValor = ($x - $mean) / $stdev;
        if ($nomValor !~ /\-/) {
                $nomValor =~ /(\d+\.\d\d)/;
                $nomValor = $1;
        }else {
                $nomValor =~ /(\-\d+\.\d\d)/;
                $nomValor = $1;
        }
        $prob = $NormalTable{$nomValor};
        return $prob;
}

###########
# POISSON #
```

```
###########
#PROBAB
#
#############################################################
# FUNCTION 2.d - Poisson distribution
#############################################################
#
sub calcPoisson() {
        my $lambda = shift;
        my $p=0;
        my $P = 0;
        my $factx = 0;
        my $rand_poisson=rand();
        my $X = 0;

        for ($x=0;$x<=100;$x++) {
                if ($x == 0) {
                        $factx = 1;
                }else {
                        $factx = $x * $factx;
                }
                $p =(($lambda**$x)*(exp(-$lambda)))/$factx;
                $P += $p;
                if ($P >= $rand_poisson) {
                        $X = $x;
                        last;
                }
        }
        return $X;
}

###################
# CHI-squared test #
###################
#
#############################################################
# FUNCTION 2.c - Chi-squared test
#############################################################
#
sub chi_square_test() {
        my ($df) = @_;
        my $chi_table = 0;
        my %chi_table95 = ("1" => "3.84", "2" => "5.99","3" => "7.81","4" => "9.49","5" =>
"11.07","6" => "12.59","72" => "14.07","8" => "15.51","9" => "16.92","10" => "18.31","11" =>
"19.68","12" => "21.03","13" => "22.36","14" => "23.68","15" => "25","16" => "26.3","17" =>
"27.59","18" => "28.87","19" => "30.14","20" => "31.41","21" => "32.67","22" => "33.92","23"
=> "35.17","24" => "36.42","25" => "37.65","26" => "38.89","27" => "40.11","28" =>
"41.34","29" => "42.56","30" => "43.77");
        $chi_table = $chi_table95{$df};
        return $chi_table;
}


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

```
# Split
######################
# Calculation of CAI and E-CAI for each sequence
#
############################################################
# FUNCTION 2.c - CAI, eCAI and Normalized CAI calculation
############################################################
#
sub split_seq() {
	my ($fitxer, $tipusGenoma, $hoste, $fileCAI1, $errorC, $population, $fileCAI2,
$numSequences, $long, $method, $intGC, $fileExpected) = @_;

	my $result = "RESULT.txt"; #Output file
	my $tmp = "tmp";
	my $aux = "";

	#############################################################################
	#SCRIPT
	#############################################################################
	open (INff, "<$fitxer") || die "cannot open $fitxer";
	open (OUTR, ">$tmp");
	open (result, ">$result");
	print result "NAME\tCAI\teCAI\tNORMALIZED\tSIGNIFICANCE ($errorC %)\n";
	my $cont = 0;

	while (<INff>) {
		chomp;
		if (/^>/) {
			$aux = $_;
			if ($cont == 1) {
				close OUTR;

				&sequencies($tmp, $tipusGenoma, $hoste, $fileCAI1 );
				&CalcMitjanaCAIrandom($tmp, $tipusGenoma, $hoste, $errorC,
$population, $fileCAI2, $numSequences, $long, $method, $intGC);

				my ($cai, $ecai, $normalized, $sig) = &normalized("$fileCAI1",
"$fileExpected");

				print  result  "$namess\t$cai\t$ecai\t$normalized\t$sig\n";

				open (OUTR, ">$tmp");
			}
			$cont = 1;
			$namess = $aux;
			$namess =~ />(.+)/;
			$namess = $1;
			print OUTR ">$namess\n";
		}else {
			print OUTR "$_\n";
		}
	}

	if ($cont == 1) {
```

```
                close OUTR;
                &sequencies($tmp, $tipusGenoma, $hoste, $fileCAI1 );
                &CalcMitjanaCAIrandom($tmp, $tipusGenoma, $hoste, $errorC, $population,
$fileCAI2, $numSequences, $long, $method, $intGC);

                my ($cai, $ecai, $normalized, $sig) = &normalized("$fileCAI1",
"$fileExpected");

                print  result  "$namess\t$cai\t$ecai\t$normalized\t$sig\n";

                open (OUTR, ">$tmp");
        }
        close INff;
        close OUTR;
        close result;

        ############################################################
        # FUNCTION 2.f - Normalized CAI calculation
        ############################################################
        #
        sub normalized() {
                ($caifile, $ecaifile) = @_;
                # CAI
                open (IN2, "<$caifile");
                while (<IN2>) {
                        if (/^>/) {
                                $cai = /(\d+\.\d+)/;
                                $cai = $1;
                        }
                }
                close IN2;

                #ECAI
                open (IN2, "<$ecaifile");
                while (<IN2>) {
                        if (/^Upper/) {
                                $ecai = /(\d+\.\d+)/;
                                $ecai = $1;
                        }
                }
                close IN2;

                # Normalized
                $normalized = $cai/$ecai;

                # Significance
                if ($normalized > 1) {
                        $sig = "*";
                }
                else {
                        $sig = "ns";
                }
                return ($cai, $ecai, $normalized, $sig);
        }
```

```
}

############################################################
# FUNCTION 2.d - RCDI, eRCDI and Normalized RCDI calculation
############################################################
#
sub split_seq_rcdi() {
	my ($fitxer, $tipusGenoma, $hoste, $fileCAI1, $errorC, $population, $fileCAI2,
$numSequences, $long, $method, $intGC, $fileExpected) = @_;

	my $result = "RESULT.txt"; #Output file
	my $tmp = "tmp";
	my $aux = "";

	my $calcrcdi ="Y";

	#############################################################################
	#SCRIPT
	#############################################################################
	open (INff, "<$fitxer") || die "cannot open $fitxer";
	open (OUTR, ">$tmp");
	open (result, ">$result");
	print result "NAME\tRCDI\teRCDI\tNORMALIZED\tSIGNIFICANCE ($errorC %)\n";
	my $cont = 0;

	while (<INff>) {
		chomp;
		if (/^>/) {
			$aux = $_;
			if ($cont == 1) {
				close OUTR;

				&sequencies_rcdi($tmp, $tipusGenoma, $hoste, $fileCAI1 );
				&CalcMitjanaCAIrandom($tmp, $tipusGenoma, $hoste, $errorC,
$population, $fileCAI2, $numSequences, $long, $method, $intGC, $calcrcdi);

				my ($cai, $ecai, $normalized, $sig) = &normalized("$fileCAI1",
"$fileExpected");

				print  result  "$namess\t$cai\t$ecai\t$normalized\t$sig\n";

				open (OUTR, ">$tmp");
			}
			$cont = 1;
			$namess = $aux;
			$namess =~ />(.+)/;
			$namess = $1;
			print OUTR ">$namess\n";
		}else {
			print OUTR "$_\n";
		}
	}
```

```
	if ($cont == 1) {
		close OUTR;
		&sequencies_rcdi($tmp, $tipusGenoma, $hoste, $fileCAI1 );
		&CalcMitjanaCAIrandom($tmp, $tipusGenoma, $hoste, $errorC, $population,
$fileCAI2, $numSequences, $long, $method, $intGC, $calcrcdi);

		my ($cai, $ecai, $normalized, $sig) = &normalized("$fileCAI1",
"$fileExpected");

		print  result  "$namess\t$cai\t$ecai\t$normalized\t$sig\n";

		open (OUTR, ">$tmp");
	}
	close INff;
	close OUTR;
	close result;

	############################################################
	# FUNCTION 2.f - Normalized CAI calculation
	############################################################
	#
	sub normalized() {
		($caifile, $ecaifile) = @_;
		# CAI
		open (IN2, "<$caifile");
		while (<IN2>) {
			if (/^>/) {
				$cai = /(\d+\.\d+)/;
				$cai = $1;
			}
		}
		close IN2;

		#ECAI
		open (IN2, "<$ecaifile");
		while (<IN2>) {
			if (/^Upper/) {
				$ecai = /(\d+\.\d+)/;
				$ecai = $1;
			}
		}
		close IN2;

		# Normalized
		$normalized = $cai/$ecai;

		# Significance
		if ($normalized > 1) {
			$sig = "*";
		}
		else {
			$sig = "ns";
		}
		return ($cai, $ecai, $normalized, $sig);
```

```
	}

}

####################################
#
##############################
# FUNCTION 3.b - Genetic code
##############################
sub geneticCode {
	my ($tipusGenoma) = shift;

	my %codigenetic = ();
	my @codigenetic_aux = ();
	# Genetic codes available in CAIcal.
	# By default the CAIcal program uses the standard genetic code (1)
	if ($tipusGenoma == 11) {
		#genetic code 11
		%codigenetic =
("TTT","F","TCT","S","TAT","Y","TGT","C","TTC","F","TCC","S","TAC","Y","TGC","C","TTA","L","TC
A","S","TAA",".","TGA",".","TTG","L","TCG","S","TAG",".","TGG","W","CTT","L","CCT","P","CAT","
H","CGT","R","CTC","L","CCC","P","CAC","H","CGC","R","CTA","L","CCA","P","CAA","Q","CGA","R","
CTG","L","CCG","P","CAG","Q","CGG","R","ATT","I","ACT","T","AAT","N","AGT","S","ATC","I","ACC"
,"T","AAC","N","AGC","S","ATA","I","ACA","T","AAA","K","AGA","R","ATG","M","ACG","T","AAG","K"
,"AGG","R","GTT","V","GCT","A","GAT","D","GGT","G","GTC","V","GCC","A","GAC","D","GGC","G","GT
A","V","GCA","A","GAA","E","GGA","G","GTG","V","GCG","A","GAG","E","GGG","G");
	}
	elsif ($tipusGenoma == 1) {
		#genetic code 11
		%codigenetic =
("TTT","F","TCT","S","TAT","Y","TGT","C","TTC","F","TCC","S","TAC","Y","TGC","C","TTA","L","TC
A","S","TAA",".","TGA",".","TTG","L","TCG","S","TAG",".","TGG","W","CTT","L","CCT","P","CAT","
H","CGT","R","CTC","L","CCC","P","CAC","H","CGC","R","CTA","L","CCA","P","CAA","Q","CGA","R","
CTG","L","CCG","P","CAG","Q","CGG","R","ATT","I","ACT","T","AAT","N","AGT","S","ATC","I","ACC"
,"T","AAC","N","AGC","S","ATA","I","ACA","T","AAA","K","AGA","R","ATG","M","ACG","T","AAG","K"
,"AGG","R","GTT","V","GCT","A","GAT","D","GGT","G","GTC","V","GCC","A","GAC","D","GGC","G","GT
A","V","GCA","A","GAA","E","GGA","G","GTG","V","GCG","A","GAG","E","GGG","G");
	}
	elsif ($tipusGenoma == 4) {
		#genetic code 4
		%codigenetic =
("TTT","F","TCT","S","TAT","Y","TGT","C","TTC","F","TCC","S","TAC","Y","TGC","C","TTA","L","TC
A","S","TAA",".","TGA","W","TTG","L","TCG","S","TAG",".","TGG","W","CTT","L","CCT","P","CAT","
H","CGT","R","CTC","L","CCC","P","CAC","H","CGC","R","CTA","L","CCA","P","CAA","Q","CGA","R","
CTG","L","CCG","P","CAG","Q","CGG","R","ATT","I","ACT","T","AAT","N","AGT","S","ATC","I","ACC"
,"T","AAC","N","AGC","S","ATA","I","ACA","T","AAA","K","AGA","R","ATG","M","ACG","T","AAG","K"
,"AGG","R","GTT","V","GCT","A","GAT","D","GGT","G","GTC","V","GCC","A","GAC","D","GGC","G","GT
A","V","GCA","A","GAA","E","GGA","G","GTG","V","GCG","A","GAG","E","GGG","G");
	}
	elsif ($tipusGenoma == 2) {
		%codigenetic =
("TTT"=>"F","TCT"=>"S","TAT"=>"Y","TGT"=>"C","TTC"=>"F","TCC"=>"S","TAC"=>"Y","TGC"=>"C","TTA"
=>"L","TCA"=>"S","TAA"=>".","TGA"=>"W","TTG"=>"L","TCG"=>"S","TAG"=>".","TGG"=>"W","CTT"=>"L",
"CCT"=>"P","CAT"=>"H","CGT"=>"R","CTC"=>"L","CCC"=>"P","CAC"=>"H","CGC"=>"R","CTA"=>"L","CCA"=
```

```
>"P","CAA"=>"Q","CGA"=>"R","CTG"=>"L","CCG"=>"P","CAG"=>"Q","CGG"=>"R","ATT"=>"I","ACT"=>"T","
AAT"=>"N","AGT"=>"S","ATC"=>"I","ACC"=>"T","AAC"=>"N","AGC"=>"S","ATA"=>"M","ACA"=>"T","AAA"=>
"K","AGA"=>".","ATG"=>"M","ACG"=>"T","AAG"=>"K","AGG"=>".","GTT"=>"V","GCT"=>"A","GAT"=>"D","G
GT"=>"G","GTC"=>"V","GCC"=>"A","GAC"=>"D","GGC"=>"G","GTA"=>"V","GCA"=>"A","GAA"=>"E","GGA"=>"
G","GTG"=>"V","GCG"=>"A","GAG"=>"E","GGG"=>"G");
        }
        elsif ($tipusGenoma == 3) {
                #genetic code 3
                %codigenetic =
("TTT"=>"F","TCT"=>"S","TAT"=>"Y","TGT"=>"C","TTC"=>"F","TCC"=>"S","TAC"=>"Y","TGC"=>"C","TTA"
=>"L","TCA"=>"S","TAA"=>".","TGA"=>"W","TTG"=>"L","TCG"=>"S","TAG"=>".","TGG"=>"W","CTT"=>"T",
"CCT"=>"P","CAT"=>"H","CGT"=>"R","CTC"=>"T","CCC"=>"P","CAC"=>"H","CGC"=>"R","CTA"=>"T","CCA"=
>"P","CAA"=>"Q","CGA"=>"R","CTG"=>"T","CCG"=>"P","CAG"=>"Q","CGG"=>"R","ATT"=>"I","ACT"=>"T","
AAT"=>"N","AGT"=>"S","ATC"=>"I","ACC"=>"T","AAC"=>"N","AGC"=>"S","ATA"=>"M","ACA"=>"T","AAA"=>
"K","AGA"=>"R","ATG"=>"M","ACG"=>"T","AAG"=>"K","AGG"=>"R","GTT"=>"V","GCT"=>"A","GAT"=>"D","G
GT"=>"G","GTC"=>"V","GCC"=>"A","GAC"=>"D","GGC"=>"G","GTA"=>"V","GCA"=>"A","GAA"=>"E","GGA"=>"
G","GTG"=>"V","GCG"=>"A","GAG"=>"E","GGG"=>"G");
        }
        elsif ($tipusGenoma == 5) {
                #genetic code 5
                %codigenetic =
("TTT"=>"F","TCT"=>"S","TAT"=>"Y","TGT"=>"C","TTC"=>"F","TCC"=>"S","TAC"=>"Y","TGC"=>"C","TTA"
=>"L","TCA"=>"S","TAA"=>".","TGA"=>"W","TTG"=>"L","TCG"=>"S","TAG"=>".","TGG"=>"W","CTT"=>"L",
"CCT"=>"P","CAT"=>"H","CGT"=>"R","CTC"=>"L","CCC"=>"P","CAC"=>"H","CGC"=>"R","CTA"=>"L","CCA"=
>"P","CAA"=>"Q","CGA"=>"R","CTG"=>"L","CCG"=>"P","CAG"=>"Q","CGG"=>"R","ATT"=>"I","ACT"=>"T","
AAT"=>"N","AGT"=>"S","ATC"=>"I","ACC"=>"T","AAC"=>"N","AGC"=>"S","ATA"=>"M","ACA"=>"T","AAA"=>
"K","AGA"=>"S","ATG"=>"M","ACG"=>"T","AAG"=>"K","AGG"=>"S","GTT"=>"V","GCT"=>"A","GAT"=>"D","G
GT"=>"G","GTC"=>"V","GCC"=>"A","GAC"=>"D","GGC"=>"G","GTA"=>"V","GCA"=>"A","GAA"=>"E","GGA"=>"
G","GTG"=>"V","GCG"=>"A","GAG"=>"E","GGG"=>"G");
        }
        elsif ($tipusGenoma == 6) {
                #genetic code 6
                %codigenetic =
("TTT"=>"F","TCT"=>"S","TAT"=>"Y","TGT"=>"C","TTC"=>"F","TCC"=>"S","TAC"=>"Y","TGC"=>"C","TTA"
=>"L","TCA"=>"S","TAA"=>"Q","TGA"=>".","TTG"=>"L","TCG"=>"S","TAG"=>"Q","TGG"=>"W","CTT"=>"L",
"CCT"=>"P","CAT"=>"H","CGT"=>"R","CTC"=>"L","CCC"=>"P","CAC"=>"H","CGC"=>"R","CTA"=>"L","CCA"=
>"P","CAA"=>"Q","CGA"=>"R","CTG"=>"L","CCG"=>"P","CAG"=>"Q","CGG"=>"R","ATT"=>"I","ACT"=>"T","
AAT"=>"N","AGT"=>"S","ATC"=>"I","ACC"=>"T","AAC"=>"N","AGC"=>"S","ATA"=>"I","ACA"=>"T","AAA"=>
"K","AGA"=>"R","ATG"=>"M","ACG"=>"T","AAG"=>"K","AGG"=>"R","GTT"=>"V","GCT"=>"A","GAT"=>"D","G
GT"=>"G","GTC"=>"V","GCC"=>"A","GAC"=>"D","GGC"=>"G","GTA"=>"V","GCA"=>"A","GAA"=>"E","GGA"=>"
G","GTG"=>"V","GCG"=>"A","GAG"=>"E","GGG"=>"G");
        }
        elsif ($tipusGenoma == 9) {
                #genetic code 9
                %codigenetic =
("TTT"=>"F","TCT"=>"S","TAT"=>"Y","TGT"=>"C","TTC"=>"F","TCC"=>"S","TAC"=>"Y","TGC"=>"C","TTA"
=>"L","TCA"=>"S","TAA"=>".","TGA"=>"W","TTG"=>"L","TCG"=>"S","TAG"=>".","TGG"=>"W","CTT"=>"L",
"CCT"=>"P","CAT"=>"H","CGT"=>"R","CTC"=>"L","CCC"=>"P","CAC"=>"H","CGC"=>"R","CTA"=>"L","CCA"=
>"P","CAA"=>"Q","CGA"=>"R","CTG"=>"L","CCG"=>"P","CAG"=>"Q","CGG"=>"R","ATT"=>"I","ACT"=>"T","
AAT"=>"N","AGT"=>"S","ATC"=>"I","ACC"=>"T","AAC"=>"N","AGC"=>"S","ATA"=>"I","ACA"=>"T","AAA"=>
"N","AGA"=>"S","ATG"=>"M","ACG"=>"T","AAG"=>"K","AGG"=>"S","GTT"=>"V","GCT"=>"A","GAT"=>"D","G
GT"=>"G","GTC"=>"V","GCC"=>"A","GAC"=>"D","GGC"=>"G","GTA"=>"V","GCA"=>"A","GAA"=>"E","GGA"=>"
G","GTG"=>"V","GCG"=>"A","GAG"=>"E","GGG"=>"G");
        }
```

```
        elsif ($tipusGenoma == 10) {
                #genetic code 10
                %codigenetic =
("TTT"=>"F","TCT"=>"S","TAT"=>"Y","TGT"=>"C","TTC"=>"F","TCC"=>"S","TAC"=>"Y","TGC"=>"C","TTA"
=>"L","TCA"=>"S","TAA"=>".","TGA"=>"C","TTG"=>"L","TCG"=>"S","TAG"=>".","TGG"=>"W","CTT"=>"L",
"CCT"=>"P","CAT"=>"H","CGT"=>"R","CTC"=>"L","CCC"=>"P","CAC"=>"H","CGC"=>"R","CTA"=>"L","CCA"=
>"P","CAA"=>"Q","CGA"=>"R","CTG"=>"L","CCG"=>"P","CAG"=>"Q","CGG"=>"R","ATT"=>"I","ACT"=>"T","
AAT"=>"N","AGT"=>"S","ATC"=>"I","ACC"=>"T","AAC"=>"N","AGC"=>"S","ATA"=>"I","ACA"=>"T","AAA"=>
"K","AGA"=>"R","ATG"=>"M","ACG"=>"T","AAG"=>"K","AGG"=>"R","GTT"=>"V","GCT"=>"A","GAT"=>"D","G
GT"=>"G","GTC"=>"V","GCC"=>"A","GAC"=>"D","GGC"=>"G","GTA"=>"V","GCA"=>"A","GAA"=>"E","GGA"=>"
G","GTG"=>"V","GCG"=>"A","GAG"=>"E","GGG"=>"G");
        }
        elsif ($tipusGenoma == 11) {
                #genetic code 11
                %codigenetic =
("TTT"=>"F","TCT"=>"S","TAT"=>"Y","TGT"=>"C","TTC"=>"F","TCC"=>"S","TAC"=>"Y","TGC"=>"C","TTA"
=>"L","TCA"=>"S","TAA"=>".","TGA"=>".","TTG"=>"L","TCG"=>"S","TAG"=>".","TGG"=>"W","CTT"=>"L",
"CCT"=>"P","CAT"=>"H","CGT"=>"R","CTC"=>"L","CCC"=>"P","CAC"=>"H","CGC"=>"R","CTA"=>"L","CCA"=
>"P","CAA"=>"Q","CGA"=>"R","CTG"=>"L","CCG"=>"P","CAG"=>"Q","CGG"=>"R","ATT"=>"I","ACT"=>"T","
AAT"=>"N","AGT"=>"S","ATC"=>"I","ACC"=>"T","AAC"=>"N","AGC"=>"S","ATA"=>"I","ACA"=>"T","AAA"=>
"K","AGA"=>"R","ATG"=>"M","ACG"=>"T","AAG"=>"K","AGG"=>"R","GTT"=>"V","GCT"=>"A","GAT"=>"D","G
GT"=>"G","GTC"=>"V","GCC"=>"A","GAC"=>"D","GGC"=>"G","GTA"=>"V","GCA"=>"A","GAA"=>"E","GGA"=>"
G","GTG"=>"V","GCG"=>"A","GAG"=>"E","GGG"=>"G");
        }
        elsif ($tipusGenoma == 12) {
                #genetic code 12
                %codigenetic =
("TTT"=>"F","TCT"=>"S","TAT"=>"Y","TGT"=>"C","TTC"=>"F","TCC"=>"S","TAC"=>"Y","TGC"=>"C","TTA"
=>"L","TCA"=>"S","TAA"=>".","TGA"=>".","TTG"=>"L","TCG"=>"S","TAG"=>".","TGG"=>"W","CTT"=>"L",
"CCT"=>"P","CAT"=>"H","CGT"=>"R","CTC"=>"L","CCC"=>"P","CAC"=>"H","CGC"=>"R","CTA"=>"L","CCA"=
>"P","CAA"=>"Q","CGA"=>"R","CTG"=>"S","CCG"=>"P","CAG"=>"Q","CGG"=>"R","ATT"=>"I","ACT"=>"T","
AAT"=>"N","AGT"=>"S","ATC"=>"I","ACC"=>"T","AAC"=>"N","AGC"=>"S","ATA"=>"I","ACA"=>"T","AAA"=>
"K","AGA"=>"R","ATG"=>"M","ACG"=>"T","AAG"=>"K","AGG"=>"R","GTT"=>"V","GCT"=>"A","GAT"=>"D","G
GT"=>"G","GTC"=>"V","GCC"=>"A","GAC"=>"D","GGC"=>"G","GTA"=>"V","GCA"=>"A","GAA"=>"E","GGA"=>"
G","GTG"=>"V","GCG"=>"A","GAG"=>"E","GGG"=>"G");
        }
        elsif ($tipusGenoma == 13) {
                #genetic code 13
                %codigenetic =
("TTT"=>"F","TCT"=>"S","TAT"=>"Y","TGT"=>"C","TTC"=>"F","TCC"=>"S","TAC"=>"Y","TGC"=>"C","TTA"
=>"L","TCA"=>"S","TAA"=>".","TGA"=>"W","TTG"=>"L","TCG"=>"S","TAG"=>".","TGG"=>"W","CTT"=>"L",
"CCT"=>"P","CAT"=>"H","CGT"=>"R","CTC"=>"L","CCC"=>"P","CAC"=>"H","CGC"=>"R","CTA"=>"L","CCA"=
>"P","CAA"=>"Q","CGA"=>"R","CTG"=>"L","CCG"=>"P","CAG"=>"Q","CGG"=>"R","ATT"=>"I","ACT"=>"T","
AAT"=>"N","AGT"=>"S","ATC"=>"I","ACC"=>"T","AAC"=>"N","AGC"=>"S","ATA"=>"M","ACA"=>"T","AAA"=>
"K","AGA"=>"G","ATG"=>"M","ACG"=>"T","AAG"=>"K","AGG"=>"G","GTT"=>"V","GCT"=>"A","GAT"=>"D","G
GT"=>"G","GTC"=>"V","GCC"=>"A","GAC"=>"D","GGC"=>"G","GTA"=>"V","GCA"=>"A","GAA"=>"E","GGA"=>"
G","GTG"=>"V","GCG"=>"A","GAG"=>"E","GGG"=>"G");
        }
        elsif ($tipusGenoma == 14) {
                #genetic code 14
                %codigenetic  =
("TTT"=>"F","TCT"=>"S","TAT"=>"Y","TGT"=>"C","TTC"=>"F","TCC"=>"S","TAC"=>"Y","TGC"=>"C","TTA"
=>"L","TCA"=>"S","TAA"=>"Y","TGA"=>"W","TTG"=>"L","TCG"=>"S","TAG"=>".","TGG"=>"W","CTT"=>"L",
"CCT"=>"P","CAT"=>"H","CGT"=>"R","CTC"=>"L","CCC"=>"P","CAC"=>"H","CGC"=>"R","CTA"=>"L","CCA"=
```

```
>"P","CAA"=>"Q","CGA"=>"R","CTG"=>"L","CCG"=>"P","CAG"=>"Q","CGG"=>"R","ATT"=>"I","ACT"=>"T","
AAT"=>"N","AGT"=>"S","ATC"=>"I","ACC"=>"T","AAC"=>"N","AGC"=>"S","ATA"=>"I","ACA"=>"T","AAA"=>
"N","AGA"=>"S","ATG"=>"M","ACG"=>"T","AAG"=>"K","AGG"=>"S","GTT"=>"V","GCT"=>"A","GAT"=>"D","G
GT"=>"G","GTC"=>"V","GCC"=>"A","GAC"=>"D","GGC"=>"G","GTA"=>"V","GCA"=>"A","GAA"=>"E","GGA"=>"
G","GTG"=>"V","GCG"=>"A","GAG"=>"E","GGG"=>"G");
        }
        elsif ($tipusGenoma == 15) {
                #genetic code 15
                %codigenetic  =
("TTT"=>"F","TCT"=>"S","TAT"=>"Y","TGT"=>"C","TTC"=>"F","TCC"=>"S","TAC"=>"Y","TGC"=>"C","TTA"
=>"L","TCA"=>"S","TAA"=>".","TGA"=>".","TTG"=>"L","TCG"=>"S","TAG"=>"Q","TGG"=>"W","CTT"=>"L",
"CCT"=>"P","CAT"=>"H","CGT"=>"R","CTC"=>"L","CCC"=>"P","CAC"=>"H","CGC"=>"R","CTA"=>"L","CCA"=
>"P","CAA"=>"Q","CGA"=>"R","CTG"=>"L","CCG"=>"P","CAG"=>"Q","CGG"=>"R","ATT"=>"I","ACT"=>"T","
AAT"=>"N","AGT"=>"S","ATC"=>"I","ACC"=>"T","AAC"=>"N","AGC"=>"S","ATA"=>"I","ACA"=>"T","AAA"=>
"K","AGA"=>"R","ATG"=>"M","ACG"=>"T","AAG"=>"K","AGG"=>"R","GTT"=>"V","GCT"=>"A","GAT"=>"D","G
GT"=>"G","GTC"=>"V","GCC"=>"A","GAC"=>"D","GGC"=>"G","GTA"=>"V","GCA"=>"A","GAA"=>"E","GGA"=>"
G","GTG"=>"V","GCG"=>"A","GAG"=>"E","GGG"=>"G");
        }
        else {
                #genetic code 1
                %codigenetic =
("TTT","F","TCT","S","TAT","Y","TGT","C","TTC","F","TCC","S","TAC","Y","TGC","C","TTA","L","TC
A","S","TAA",".","TGA",".","TTG","L","TCG","S","TAG",".","TGG","W","CTT","L","CCT","P","CAT","
H","CGT","R","CTC","L","CCC","P","CAC","H","CGC","R","CTA","L","CCA","P","CAA","Q","CGA","R","
CTG","L","CCG","P","CAG","Q","CGG","R","ATT","I","ACT","T","AAT","N","AGT","S","ATC","I","ACC"
,"T","AAC","N","AGC","S","ATA","I","ACA","T","AAA","K","AGA","R","ATG","M","ACG","T","AAG","K"
,"AGG","R","GTT","V","GCT","A","GAT","D","GGT","G","GTC","V","GCC","A","GAC","D","GGC","G","GT
A","V","GCA","A","GAA","E","GGA","G","GTG","V","GCG","A","GAG","E","GGG","G");
        }

        foreach my $codon(keys(%codigenetic)) {
                push @codigenetic_aux, $codon;
                push @codigenetic_aux, $codigenetic{$codon};
        }

        return @codigenetic_aux;
}

##############################
# FUNCTION 3.c - Stop codons
##############################
sub stopCodons{
        my (@codigenetic_aux)= @_;
        my %codigenetic = @codigenetic_aux;
        my @CodonsSTOP = ();

        foreach my $codon(keys(%codigenetic)) {
                # Stop codons
                if ($codigenetic{$codon} eq ".") {
                        push @CodonsSTOP, $codon;
                }
        }

        return (@CodonsSTOP);
```

```
}

##############################
# FUNCTION 3.d - Codons per AA
##############################
sub codonsPerAA{
	my (@codigenetic_aux)= @_;
	my %codigenetic = @codigenetic_aux;
	my %CodonsPerAA = ();
	my @CodonsPerAA_aux = ();

	foreach my $codon(keys(%codigenetic)) {
		if ($codon ne ""){
			# Codons per AA
			$CodonsPerAA{$codigenetic{$codon}}++;
		}
	}
	foreach my $AA(keys(%CodonsPerAA)) {
		push @CodonsPerAA_aux, $AA;
		push @CodonsPerAA_aux, $CodonsPerAA{$AA};
	}
	return @CodonsPerAA_aux;
}

##############################
# FUNCTION 3.e - Variable AA
##############################
sub varAA {
	my (@CodonsPerAA_aux)= @_;
	my %CodonsPerAA = @CodonsPerAA_aux;
	my %varAA = ();
	my @varAA_aux = ();

	foreach my $v(sort keys(%CodonsPerAA)) {
		$varAA{$CodonsPerAA{$v}}=0;
	}

	foreach my $v(sort keys(%varAA)) {
		push @varAA_aux, $v;
		push @varAA_aux, $varAA{$v};
	}

	return @varAA_aux;
}

##############################
# FUNCTION 3.f - Codons without synonymous
##############################
sub CodonsUnics {
	my (@codigenetic_aux)= @_;
	my %codigenetic = @codigenetic_aux;
	my %CodonsPerAA = ();
	my %CodonsPerAA2 = ();
```

```
	foreach my $codon(keys(%codigenetic)) {
		if ($codon ne ""){
			# Codons per AA
			$CodonsPerAA{$codigenetic{$codon}}++;
			$CodonsPerAA2{$codigenetic{$codon}} .= "$codon ";
		}
	}

	my @CodonsUnics = ();
	foreach my $AA(keys(%CodonsPerAA)) {
		if ($CodonsPerAA{$AA} > 1) {
		}else {
			push @CodonsUnics, $CodonsPerAA2{$AA};
		}
	}
	return @CodonsUnics;
}

##############################
# FUNCTION 3.g - Codons with synonymous
##############################
sub CodonsVariables {
	my (@codigenetic_aux)= @_;
	my %codigenetic = @codigenetic_aux;
	my %CodonsPerAA = ();
	my %CodonsPerAA2 = ();

	foreach my $codon(keys(%codigenetic)) {
		if ($codon ne ""){
			# Codons per AA
			$CodonsPerAA{$codigenetic{$codon}}++;
			$CodonsPerAA2{$codigenetic{$codon}} .= "$codon ";
		}
	}

	my @CodonsVariables = ();
	foreach my $AA(keys(%CodonsPerAA)) {
		if ( ($CodonsPerAA{$AA} > 1) && ($AA ne ".") ) {
			my @aux = split /\s/, $CodonsPerAA2{$AA};
			foreach $a(@aux) {
				if ($a ne "") {
					push @CodonsVariables, $a;
				}
			}
		}
	}


	return @CodonsVariables;
}

##############################
# FUNCTION 3.h - Number of codons with synonyomous
##############################
```

```
sub nCodonsVariables {
	my @CodonsVariables = @_;
	my $countVariables = 0;

	# Number of Variable codons
	$countVariables = @CodonsVariables;

	return $countVariables;
}


###########
# Credits #
###########
sub credits() {
	print "\n\n";
	print
"#################################################################################\n";
	print "#                                                                               #\n";
	print "#                          CAIcal/E-CAI PERL version 1.4                        #\n";
	print "#                               (July 2009)                                     #\n";
	print "#                                                                               #\n";
	print "#  This version of CAIcal/E-CAI calculates CAI or RCDI                          #\n";
	print "#  from DNA sequences or the expected value determined by randomly generating
#\n";
	print "#  sequences.                                                                   #\n";
	print "#                                                                               #\n";
	print "#                Created by Pere Puigbo (puigboap".'@'."ncbi.nlm.nih.gov)  #\n";
	print "#                                                                               #\n";
	print "#              Freely available at http://genomes.urv.es/CAIcal.
	#\n";
	print "#                                                                               #\n";
	print
"#################################################################################\n\n";
	sleep(1);
}


##########
# Readme #
##########
#
############################################################
# FUNCTION 1.d - Readme
############################################################
#
sub readme() {

	print
"#########################################################################################
######################\n";
	print "\n";
	print "Parameters to run CAIcal:\n";
	print "\n";
```

```
	print '$'." CAIcal -e [cai|expected|cai_and_expected] -f [file_name] -h [file_name] -g
[1|4|11] -c [90|95|99] -p [90|95|99] -o1 [file_name] -o2 [file_name] -o3 [file_name] -n
[number] -l [number] -m [markov/poisson]\n";
	print "\n";
	print "* PROGRAM EXECUTION:                    -e
cai|expected|cai_and_expected|rcdi|ercdi|rcdi_and_ercdi       <defaults: cai_and_expected>
#Option 'cai' calculates CAI from DNA sequences and option 'expected' calculates, expected CAI
and option 'cai_and_expected' calculates both.\n";
	print "\n";
	print "* INPUT1 DATA FILE:                     -f  file_name
<default: example.ffn>                 #DNA sequences in fasta format\n";
	print "\n";
	print "* INPUT2 HOST FILE:                     -h  file_name
<default: human>                       #Codon usage reference table to calculate CAI.\n";
	print "\n";
	print "* GENETIC CODE:                         -g  1|2|3|4|5|6|9|10|11|12|13|14|15
<default: 1>                           # code 1) Standard 11) Eubacteria 4) Mycoplasma
...\n";
	print "\n";
	print "* CONFIDENCE:                           -c  90|95|99
<default: 95>\n";
	print "\n";
	print "* POPULATION:                           -p  90|95|99
<default: 99>\n";
	print "\n";
	print "* \%G+C:                                      -gc  0-100
<default: ->                     # Optional parameter\n";
	print "\n";
	print "* OUTPUT1 [2]:                        -o1 file_name
<default: ./cai>\n";
	print "\n";
	print "* OUTPUT2 (CAI random sequences):    -o2 file_name
<default: ./random_sequences_and_cai>\n";
	print "\n";
	print "* OUTPUT3 (EXPECTED CAI):              -o3 file_name
<default: ./expected> \n";
	print "\n";
	print "* NUMBER OF SEQUENCES:                  -n  number
<default: 1000>                        #1000 Sequences\n";
	print "\n";
	print "* LENGTH OF SEQUENCES:                  -l  number
<default: 300>                         # 300 codons\n";
	print "\n";
	print "* METHODS FOR RANDOM SEQUENCES          -m markov/poisson
<default: markov>                      # Markov or Poisson method\n";
	print "* SPLIT SEQUENCES          -s y/n                    <default: n>
# y:yes -> To estimate eCAI and CAI/eCAI one by one sequence,  n:no\n";
	print "\n";
	print "\n";
	print "* HELP                                  -help\n";
	print "\n";
	print
"#################################################################################################
################\n";
```

```
}

1;
```

V.4.2 CAI statistics and box plotting

```
##IPNV CAI Analysis
library(ggplot2)
library(gridExtra)
library(reshape2)
setwd("~/Google Drive/IPNV_seqcopy/ipnvroteinsnucleotidesequence/")

#Load Data
info <- read.csv("CAI_Fish.csv")
info$nCAI <- as.numeric(info$nCAI)

#Group by Gene
RdRp <- subset(info, Gene == "RdRp")
VP2 <- subset(info, Gene == "VP2")
VP3 <- subset(info, Gene == "VP3")
VP4 <- subset(info, Gene == "VP4")
VP5 <- subset(info, Gene == "VP5")
#Comb <- subset(info, Gene == "Concatenated")

##Order by year
RdRp <- RdRp[order(RdRp$Year),]
VP2 <- VP2[order(VP2$Year),]
VP3 <- VP3[order(VP3$Year),]
VP4 <- VP4[order(VP4$Year),]
VP5 <- VP5[order(VP5$Year),]

#Boxplot by Host and Gene
VP2.box <- ggplot(VP2, aes(VP2$Table, VP2$nCAI, colour = VP2$Table, fill = VP2$Table)) +
        theme_bw() + geom_boxplot() +
        geom_hline(aes(yintercept=1), colour="black", linetype="dashed") +
                          theme(panel.grid.major = element_line(colour = "grey"))
VP2.box


VP3.box <- ggplot(VP3, aes(VP3$Table, VP3$nCAI, colour = VP3$Table, fill = VP3$Table)) +
        theme_bw() + geom_boxplot() +
        geom_hline(aes(yintercept=1), colour="black", linetype="dashed") +
        theme(panel.grid.major = element_line(colour = "grey"))
VP3.box

VP4.box <- ggplot(VP4, aes(VP4$Table, VP4$nCAI, colour = VP4$Table, fill = VP4$Table)) +
        theme_bw() + geom_boxplot() +
        geom_hline(aes(yintercept=1), colour="black", linetype="dashed") +
        theme(panel.grid.major = element_line(colour = "grey"))
VP4.box

VP5.box <- ggplot(VP5, aes(VP5$Table, VP5$nCAI, colour = VP5$Table, fill = VP5$Table)) +
        theme_bw() + geom_boxplot() +
        geom_hline(aes(yintercept=1), colour="black", linetype="dashed") +
        theme(panel.grid.major = element_line(colour = "grey"))
VP5.box

RdRP.box <- ggplot(RdRp, aes(RdRp$Table, RdRp$nCAI, colour = RdRp$Table, fill = RdRp$Table)) +
```

```
        theme_bw() + geom_boxplot() +
        geom_hline(aes(yintercept=1), colour="black", linetype="dashed") +
        theme(panel.grid.major = element_line(colour = "grey"))
RdRP.box

#Boxplot by Variant + Gene
VP2.box <- ggplot(VP2, aes(VP2$Variant, VP2$nCAI, colour = VP2$Table)) +
        theme_bw() + geom_boxplot() +
        geom_hline(aes(yintercept=1), colour="black", linetype="dashed") +
        theme(panel.grid.major = element_line(colour = "grey"))
VP2.box

VP3.box <- ggplot(VP3, aes(VP3$Variant, VP3$nCAI, colour = VP3$Table)) +
        theme_bw() + geom_boxplot() + #geom_jitter()
        geom_hline(aes(yintercept=1), colour="black", linetype="dashed") +
        theme(panel.grid.major = element_line(colour = "grey"))
VP3.box

VP4.box <- ggplot(VP4, aes(VP4$Variant, VP4$nCAI, colour = VP4$Table)) +
        theme_bw() + geom_boxplot() + #geom_jitter()
        geom_hline(aes(yintercept=1), colour="black", linetype="dashed") +
        theme(panel.grid.major = element_line(colour = "grey"))
VP4.box

VP5.box <- ggplot(VP5, aes(VP5$Variant, VP5$nCAI, colour = VP5$Table)) +
        theme_bw() + geom_boxplot() + #geom_jitter()
        geom_hline(aes(yintercept=1), colour="black", linetype="dashed") +
        theme(panel.grid.major = element_line(colour = "grey"))
VP5.box

rdrp.box <- ggplot(RdRp, aes(RdRp$Variant, RdRp$nCAI, colour = RdRp$Table)) +
        theme_bw() + geom_boxplot() + #geom_jitter()
        geom_hline(aes(yintercept=1), colour="black", linetype="dashed") +
        theme(panel.grid.major = element_line(colour = "grey"))
rdrp.box

##Melt data + extract trendlines
min(VP2$Year) #1982
max(VP2$Year) #2014

#subset by table
VP2.orno <- subset(VP2, Table == "Oncor")
VP3.orno <- subset(VP3, Table == "Oncor")
VP4.orno <- subset(VP4, Table == "Oncor")
VP5.orno <- subset(VP5, Table == "Oncor")
RdRp.orno <- subset(RdRp, Table == "Oncor")
VP2.salm <- subset(VP2, Table == "Salmon")
VP3.salm <- subset(VP3, Table == "Salmon")
VP4.salm <- subset(VP4, Table == "Salmon")
VP5.salm <- subset(VP5, Table == "Salmon")
RdRp.salm <- subset(RdRp, Table == "Salmon")

##Subset by variant
VP2.orno.LV <- subset(VP2.orno, Variant == "LV")
```

```
VP2.orno.PS <- subset(VP2.orno, Variant == "Persistent")
VP2.orno.HV <- subset(VP2.orno, Variant == "Virulent")
VP3.orno.LV <- subset(VP3.orno, Variant == "LV")
VP3.orno.PS <- subset(VP3.orno, Variant == "Persistent")
VP3.orno.HV <- subset(VP3.orno, Variant == "Virulent")
VP4.orno.LV <- subset(VP4.orno, Variant == "LV")
VP4.orno.PS <- subset(VP4.orno, Variant == "Persistent")
VP4.orno.HV <- subset(VP4.orno, Variant == "Virulent")
VP5.orno.LV <- subset(VP5.orno, Variant == "LV")
VP5.orno.PS <- subset(VP5.orno, Variant == "Persistent")
VP5.orno.HV <- subset(VP5.orno, Variant == "Virulent")
RdRp.orno.LV <- subset(RdRp.orno, Variant == "LV")
RdRp.orno.PS <- subset(RdRp.orno, Variant == "Persistent")
RdRp.orno.HV <- subset(RdRp.orno, Variant == "Virulent")
VP2.salm.LV <- subset(VP2.salm, Variant == "LV")
VP2.salm.PS <- subset(VP2.salm, Variant == "Persistent")
VP2.salm.HV <- subset(VP2.salm, Variant == "Virulent")
VP3.salm.LV <- subset(VP3.salm, Variant == "LV")
VP3.salm.PS <- subset(VP3.salm, Variant == "Persistent")
VP3.salm.HV <- subset(VP3.salm, Variant == "Virulent")
VP4.salm.LV <- subset(VP4.salm, Variant == "LV")
VP4.salm.PS <- subset(VP4.salm, Variant == "Persistent")
VP4.salm.HV <- subset(VP4.salm, Variant == "Virulent")
VP5.salm.LV <- subset(VP5.salm, Variant == "LV")
VP5.salm.PS <- subset(VP5.salm, Variant == "Persistent")
VP5.salm.HV <- subset(VP5.salm, Variant == "Virulent")
RdRp.salm.LV <- subset(RdRp.salm, Variant == "LV")
RdRp.salm.PS <- subset(RdRp.salm, Variant == "Persistent")
RdRp.salm.HV <- subset(RdRp.salm, Variant == "Virulent")

#Stats for boxplot
wilcox.test(VP2.orno$nCAI, VP2.salm$nCAI) #p-value = 4.867e-05
wilcox.test(VP3.orno$nCAI, VP3.salm$nCAI) #p-value = 0.2617
wilcox.test(VP4.orno$nCAI, VP4.salm$nCAI) #p-value = 0.2812
wilcox.test(VP5.orno$nCAI, VP5.salm$nCAI) #p-value = 1.544e-08
wilcox.test(RdRp.orno$nCAI, RdRp.salm$nCAI) #p-value = 0.3529

####Stats for variant boxplots
#######Oncor
#VP2
wilcox.test(VP2.orno.HV$nCAI, VP2.orno.LV$nCAI) #p-value = 0.1163
wilcox.test(VP2.orno.HV$nCAI, VP2.orno.PS$nCAI) #p-value = 0.4887
wilcox.test(VP2.orno.PS$nCAI, VP2.orno.LV$nCAI) # p-value = 0.014
#VP3
wilcox.test(VP3.orno.HV$nCAI, VP3.orno.LV$nCAI) #p-value = 0.1158
wilcox.test(VP3.orno.HV$nCAI, VP3.orno.PS$nCAI) #p-value = 0.1185
wilcox.test(VP3.orno.PS$nCAI, VP3.orno.LV$nCAI) # p-value = 0.048
#VP4
wilcox.test(VP4.orno.HV$nCAI, VP4.orno.LV$nCAI) #p-value = 0.321
wilcox.test(VP4.orno.HV$nCAI, VP4.orno.PS$nCAI) #p-value = 1
wilcox.test(VP4.orno.PS$nCAI, VP4.orno.LV$nCAI) # p-value = 0.1915
#VP5
wilcox.test(VP5.orno.HV$nCAI, VP5.orno.LV$nCAI) #p-value = 0.1133
wilcox.test(VP5.orno.HV$nCAI, VP5.orno.PS$nCAI) #p-value = 0.124
```

```
wilcox.test(VP5.orno.PS$nCAI, VP5.orno.LV$nCAI) # p-value = 0.1556
#RDRP
wilcox.test(RdRp.orno.HV$nCAI, RdRp.orno.LV$nCAI) #p-value = 0.2257
wilcox.test(RdRp.orno.HV$nCAI, RdRp.orno.PS$nCAI) #p-value = 0.95
wilcox.test(RdRp.orno.PS$nCAI, RdRp.orno.LV$nCAI) # p-value = 0.00037

####Salmon
#VP2
wilcox.test(VP2.salm.HV$nCAI, VP2.salm.LV$nCAI) #p-value = 0.1164
wilcox.test(VP2.salm.HV$nCAI, VP2.salm.PS$nCAI) #p-value = 0.5989
wilcox.test(VP2.salm.PS$nCAI, VP2.salm.LV$nCAI) # p-value = 0.01627
#VP3
wilcox.test(VP3.salm.HV$nCAI, VP3.salm.LV$nCAI) #p-value = 0.1159
wilcox.test(VP3.salm.HV$nCAI, VP3.salm.PS$nCAI) #p-value = 0.1874
wilcox.test(VP3.salm.PS$nCAI, VP3.salm.LV$nCAI) # p-value = 0.3926
#VP4
wilcox.test(VP4.salm.HV$nCAI, VP4.salm.LV$nCAI) #p-value = 0.1659
wilcox.test(VP4.salm.HV$nCAI, VP4.salm.PS$nCAI) #p-value = 1
wilcox.test(VP4.salm.PS$nCAI, VP4.salm.LV$nCAI) # p-value = 0.1952
#VP5
wilcox.test(VP5.salm.HV$nCAI, VP5.salm.LV$nCAI) #p-value = 0.1133
wilcox.test(VP5.salm.HV$nCAI, VP5.salm.PS$nCAI) #p-value = 0.1261
wilcox.test(VP5.salm.PS$nCAI, VP5.salm.LV$nCAI) # p-value = 0.05453
#RDRP
wilcox.test(RdRp.salm.HV$nCAI, RdRp.salm.LV$nCAI) #p-value = 0.2278
wilcox.test(RdRp.salm.HV$nCAI, RdRp.salm.PS$nCAI) #p-value = 0.89
wilcox.test(RdRp.salm.PS$nCAI, RdRp.salm.LV$nCAI) # p-value = 0.0004383

##Compare against species
##VP2
wilcox.test(VP2.orno.HV$nCAI, VP2.salm.HV$nCAI) #p-value = 1
wilcox.test(VP2.orno.LV$nCAI, VP2.salm.LV$nCAI) #p-value = 8.178e-06
wilcox.test(VP2.orno.PS$nCAI, VP2.salm.PS$nCAI) #p-value = 0.01589
##VP3
wilcox.test(VP3.orno.HV$nCAI, VP3.salm.HV$nCAI)#p-value = 1
wilcox.test(VP3.orno.LV$nCAI, VP3.salm.LV$nCAI)#p-value = 0.1159
wilcox.test(VP3.orno.PS$nCAI, VP3.salm.PS$nCAI)#p-value = 0.7418
#VP4
wilcox.test(VP4.orno.HV$nCAI, VP4.salm.HV$nCAI)#p-value =1
wilcox.test(VP4.orno.LV$nCAI, VP4.salm.LV$nCAI)#p-value = 0.06287
wilcox.test(VP4.orno.PS$nCAI, VP4.salm.PS$nCAI)#p-value =0.6142
#VP5
wilcox.test(VP5.orno.HV$nCAI, VP5.salm.HV$nCAI)#p-value =1
wilcox.test(VP5.orno.LV$nCAI, VP5.salm.LV$nCAI)#p-value = 0.0008824
wilcox.test(VP5.orno.PS$nCAI, VP5.salm.PS$nCAI)#p-value = 7.267e-06
#RdRp
wilcox.test(RdRp.orno.HV$nCAI, RdRp.salm.HV$nCAI)#p-value = NA
wilcox.test(RdRp.orno.LV$nCAI, RdRp.salm.LV$nCAI)#p-value = 0.2724
wilcox.test(RdRp.orno.PS$nCAI, RdRp.salm.PS$nCAI)#p-value = 0.8345

#Calculate loess
onco.lo.vp2 <- loess(nCAI ~ Year, VP2.onco)
onco.lo.vp3 <- loess(nCAI ~ Year, VP3.onco)
onco.lo.vp4 <- loess(nCAI ~ Year, VP4.onco)
```

```
onco.lo.vp5 <- loess(nCAI ~ Year, VP5.onco)
onco.lo.rdrp <- loess(nCAI ~ Year, RdRp.onco)
salm.lo.vp2 <- loess(nCAI ~ Year, VP2.salm)
salm.lo.vp3 <- loess(nCAI ~ Year, VP3.salm)
salm.lo.vp4 <- loess(nCAI ~ Year, VP4.salm)
salm.lo.vp5 <- loess(nCAI ~ Year, VP5.salm)
salm.lo.rdrp <- loess(nCAI ~ Year, RdRp.salm)
#####Oncor
#VP2
VP2.pred <- predict(onco.lo.vp2, data.frame(Year = seq(1982, 2014, 1)))
VP2.pred <- as.data.frame(VP2.pred)
#VP3
VP3.pred <- predict(onco.lo.vp3, data.frame(Year = seq(1982, 2014, 1)))
VP3.pred <- as.data.frame(VP3.pred)
#VP4
VP4.pred <- predict(onco.lo.vp4, data.frame(Year = seq(1982, 2014, 1)))
VP4.pred <- as.data.frame(VP4.pred)
#VP5
VP5.pred <- predict(onco.lo.vp5, data.frame(Year = seq(1982, 2014, 1)))
VP5.pred <- as.data.frame(VP5.pred)
#RdRP
rdrp.pred <- predict(onco.lo.rdrp, data.frame(Year = seq(1982, 2014, 1)))
rdrp.pred <- as.data.frame(rdrp.pred)

#combine + melt
o1 <- as.data.frame(c(VP2.pred, VP3.pred, VP4.pred, VP5.pred, rdrp.pred))
o1$Year <- c(1982:2014)
oncor_long <- melt(o1, id="Year")

##By Time
Oncor.time <- ggplot(oncor_long, aes(Year, value, colour = variable, fill = variable)) +
        theme_bw()  + geom_line() +
        geom_hline(aes(yintercept=1), colour="black", linetype="dashed") +
        theme(panel.grid.major = element_line(colour = "grey"))
Oncor.time

##Salmon
#VP2
VP2.pred.s <- predict(salm.lo.vp2, data.frame(Year = seq(1982, 2014, 1)))
VP2.pred.s <- as.data.frame(VP2.pred.s)
#VP3
VP3.pred.s <- predict(salm.lo.vp3, data.frame(Year = seq(1982, 2014, 1)))
VP3.pred.s <- as.data.frame(VP3.pred.s)
#VP4
VP4.pred.s <- predict(salm.lo.vp4, data.frame(Year = seq(1982, 2014, 1)))
VP4.pred.s <- as.data.frame(VP4.pred.s)
#VP5
VP5.pred.s <- predict(salm.lo.vp5, data.frame(Year = seq(1982, 2014, 1)))
VP5.pred.s <- as.data.frame(VP5.pred.s)
#RdRP
rdrp.pred.s <- predict(salm.lo.rdrp, data.frame(Year = seq(1982, 2014, 1)))
rdrp.pred.s <- as.data.frame(rdrp.pred.s)
#combine + melt
s1 <- as.data.frame(c(VP2.pred.s, VP3.pred.s, VP4.pred.s, VP5.pred.s, rdrp.pred.s))
```

```
s1$Year <- c(1982:2014)
s1 <- as.data.frame(s1)
salm_long <- melt(s1, id="Year")

##By Time
salmon.time <- ggplot(salm_long, aes(Year, value, colour = variable, fill = variable)) +
        theme_bw()  + geom_line() +
geom_hline(aes(yintercept=1), colour="black", linetype="dashed") +
        theme(panel.grid.major = element_line(colour = "grey"))
salmon.time

###Split Figures
VP2.SALM <- subset(salm_long, variable=="VP2.pred.s")
VP3.SALM <- subset(salm_long, variable=="VP3.pred.s")
VP4.SALM <- subset(salm_long, variable=="VP4.pred.s")
VP5.SALM <- subset(salm_long, variable=="VP5.pred.s")
RDRP.SALM <- subset(salm_long, variable=="rdrp.pred.s")
VP2.ONCO <- subset(oncor_long, variable=="VP2.pred")
VP3.ONCO <- subset(oncor_long, variable=="VP3.pred")
VP4.ONCO <- subset(oncor_long, variable=="VP4.pred")
VP5.ONCO <- subset(oncor_long, variable=="VP5.pred")
RDRP.ONCO <- subset(oncor_long, variable=="rdrp.pred")

#combine and graph Graph
#VP2
VP2.SALM$Table <- rep("Salmon",nrow(VP2.SALM))
VP2.ONCO$Table <- rep("Oncor",nrow(VP2.ONCO))
VP2.c <- rbind(VP2.SALM, VP2.ONCO)

CAI.VP2.l <- ggplot(VP2.c, aes(Year, value, colour = Table, fill = Table)) +
        theme_bw()  + geom_line() +
        geom_hline(aes(yintercept=1), colour="black", linetype="dashed") +
        theme(panel.grid.major = element_line(colour = "grey"))
CAI.VP2.l
#VP3
VP3.SALM$Table <- rep("Salmon",nrow(VP3.SALM))
VP3.ONCO$Table <- rep("Oncor",nrow(VP3.ONCO))
VP3.c <- rbind(VP3.SALM, VP3.ONCO)

CAI.VP3.l <- ggplot(VP3.c, aes(Year, value, colour = Table, fill = Table)) +
        theme_bw()  + geom_line() +
        geom_hline(aes(yintercept=1), colour="black", linetype="dashed") +
        theme(panel.grid.major = element_line(colour = "grey"))
CAI.VP3.l
##Vp4
VP4.SALM$Table <- rep("Salmon",nrow(VP4.SALM))
VP4.ONCO$Table <- rep("Oncor",nrow(VP4.ONCO))
VP4.c <- rbind(VP4.SALM, VP4.ONCO)

CAI.VP4.l <- ggplot(VP4.c, aes(Year, value, colour = Table, fill = Table)) +
        theme_bw()  + geom_line() +
        geom_hline(aes(yintercept=1), colour="black", linetype="dashed") +
        theme(panel.grid.major = element_line(colour = "grey"))
CAI.VP4.l
```

```
##Vp5
VP5.SALM$Table <- rep("Salmon",nrow(VP5.SALM))
VP5.ONCO$Table <- rep("Oncor",nrow(VP5.ONCO))
VP5.c <- rbind(VP5.SALM, VP5.ONCO)

CAI.VP5.l <- ggplot(VP5.c, aes(Year, value, colour = Table, fill = Table)) +
        theme_bw()  + geom_line() +
        geom_hline(aes(yintercept=1), colour="black", linetype="dashed") +
        theme(panel.grid.major = element_line(colour = "grey"))
CAI.VP5.l
##RDRP
RDRP.SALM$Table <- rep("Salmon",nrow(RDRP.SALM))
RDRP.ONCO$Table <- rep("Oncor",nrow(RDRP.ONCO))
RDRP.c <- rbind(RDRP.SALM, RDRP.ONCO)

CAI.RDRP.l <- ggplot(RDRP.c, aes(Year, value, colour = Table, fill = Table)) +
        theme_bw()  + geom_line() +
        geom_hline(aes(yintercept=1), colour="black", linetype="dashed") +
        theme(panel.grid.major = element_line(colour = "grey"))
CAI.RDRP.l

#Correlation stats for trends
cor.test(VP2.ONCO$value, VP2.SALM$value)
cor.test(VP3.ONCO$value, VP3.SALM$value)
cor.test(VP4.ONCO$value, VP4.SALM$value)
cor.test(VP5.ONCO$value, VP5.SALM$value)
cor.test(RDRP.ONCO$value, RDRP.SALM$value)

##Stats for trends by decades 1982-1990, 1991-2000, 2001-2010, 2010-2014.
#####Oncor
#VP2
VP2.pred.80s <- predict(onco.lo.vp2, data.frame(Year = seq(1982, 1990, 1)))
VP2.pred.80s <- as.data.frame(VP2.pred.80s)
VP2.pred.90s <- predict(onco.lo.vp2, data.frame(Year = seq(1991, 2000, 1)))
VP2.pred.90s <- as.data.frame(VP2.pred.90s)
VP2.pred.00s <- predict(onco.lo.vp2, data.frame(Year = seq(2001, 2010, 1)))
VP2.pred.00s <- as.data.frame(VP2.pred.00s)
VP2.pred.10s <- predict(onco.lo.vp2, data.frame(Year = seq(2010, 2014, 1)))
VP2.pred.10s <- as.data.frame(VP2.pred.10s)
wilcox.test(VP2.pred.80s$VP2.pred.80s, VP2.pred.90s$VP2.pred.90s) #p-value = 2.165e-05
wilcox.test(VP2.pred.90s$VP2.pred.90s, VP2.pred.00s$VP2.pred.00s) # p-value = 0.3527
wilcox.test(VP2.pred.00s$VP2.pred.00s, VP2.pred.10s$VP2.pred.10s) #p-value = 0.462

#VP3
VP3.pred.80s <- predict(onco.lo.vp3, data.frame(Year = seq(1982, 1990, 1)))
VP3.pred.80s <- as.data.frame(VP3.pred.80s)
VP3.pred.90s <- predict(onco.lo.vp3, data.frame(Year = seq(1991, 2000, 1)))
VP3.pred.90s <- as.data.frame(VP3.pred.90s)
VP3.pred.00s <- predict(onco.lo.vp3, data.frame(Year = seq(2001, 2010, 1)))
VP3.pred.00s <- as.data.frame(VP3.pred.00s)
VP3.pred.10s <- predict(onco.lo.vp3, data.frame(Year = seq(2010, 2014, 1)))
VP3.pred.10s <- as.data.frame(VP3.pred.10s)
wilcox.test(VP3.pred.80s$VP3.pred.80s, VP3.pred.90s$VP3.pred.90s) #p-value = 2.165e-05
wilcox.test(VP3.pred.90s$VP3.pred.90s, VP3.pred.00s$VP3.pred.00s) # p-value = 1.083e-05
```

```
wilcox.test(VP3.pred.00s$VP3.pred.00s, VP3.pred.10s$VP3.pred.10s) #p-value = 0.03717

#VP4
VP4.pred.80s <- predict(onco.lo.vp4, data.frame(Year = seq(1982, 1990, 1)))
VP4.pred.80s <- as.data.frame(VP4.pred.80s)
VP4.pred.90s <- predict(onco.lo.vp4, data.frame(Year = seq(1991, 2000, 1)))
VP4.pred.90s <- as.data.frame(VP4.pred.90s)
VP4.pred.00s <- predict(onco.lo.vp4, data.frame(Year = seq(2001, 2010, 1)))
VP4.pred.00s <- as.data.frame(VP4.pred.00s)
VP4.pred.10s <- predict(onco.lo.vp4, data.frame(Year = seq(2010, 2014, 1)))
VP4.pred.10s <- as.data.frame(VP4.pred.10s)
wilcox.test(VP4.pred.80s$VP4.pred.80s, VP4.pred.90s$VP4.pred.90s) #p-value = 2.165e-05
wilcox.test(VP4.pred.90s$VP4.pred.90s, VP4.pred.00s$VP4.pred.00s) # p-value = 0.123
wilcox.test(VP4.pred.00s$VP4.pred.00s, VP4.pred.10s$VP4.pred.10s) #p-value = 0.003261

#VP5
VP5.pred.80s <- predict(onco.lo.vp5, data.frame(Year = seq(1982, 1990, 1)))
VP5.pred.80s <- as.data.frame(VP5.pred.80s)
VP5.pred.90s <- predict(onco.lo.vp5, data.frame(Year = seq(1991, 2000, 1)))
VP5.pred.90s <- as.data.frame(VP5.pred.90s)
VP5.pred.00s <- predict(onco.lo.vp5, data.frame(Year = seq(2001, 2010, 1)))
VP5.pred.00s <- as.data.frame(VP5.pred.00s)
VP5.pred.10s <- predict(onco.lo.vp5, data.frame(Year = seq(2010, 2014, 1)))
VP5.pred.10s <- as.data.frame(VP5.pred.10s)
wilcox.test(VP5.pred.80s$VP5.pred.80s, VP5.pred.90s$VP5.pred.90s) #p-value = 2.165e-05
wilcox.test(VP5.pred.90s$VP5.pred.90s, VP5.pred.00s$VP5.pred.00s) # p-value = 0.003886
wilcox.test(VP5.pred.00s$VP5.pred.00s, VP5.pred.10s$VP5.pred.10s) #p-value = 0.003261
#RdRP
RDRP.pred.80s <- predict(onco.lo.rdrp, data.frame(Year = seq(1982, 1990, 1)))
RDRP.pred.80s <- as.data.frame(RDRP.pred.80s)
RDRP.pred.90s <- predict(onco.lo.rdrp, data.frame(Year = seq(1991, 2000, 1)))
RDRP.pred.90s <- as.data.frame(RDRP.pred.90s)
RDRP.pred.00s <- predict(onco.lo.rdrp, data.frame(Year = seq(2001, 2010, 1)))
RDRP.pred.00s <- as.data.frame(RDRP.pred.00s)
RDRP.pred.10s <- predict(onco.lo.rdrp, data.frame(Year = seq(2010, 2014, 1)))
RDRP.pred.10s <- as.data.frame(RDRP.pred.10s)
wilcox.test(RDRP.pred.80s$RDRP.pred.80s, RDRP.pred.90s$RDRP.pred.90s) #p-value = 2.165e-05
wilcox.test(RDRP.pred.90s$RDRP.pred.90s, RDRP.pred.00s$RDRP.pred.00s) # p-value = 0.2475
wilcox.test(RDRP.pred.00s$RDRP.pred.00s, RDRP.pred.10s$RDRP.pred.10s) #p-value = 0.003261


##Salmon
#VP2
VP2.pred.80s <- predict(salm.lo.vp2, data.frame(Year = seq(1982, 1990, 1)))
VP2.pred.80s <- as.data.frame(VP2.pred.80s)
VP2.pred.90s <- predict(salm.lo.vp2, data.frame(Year = seq(1991, 2000, 1)))
VP2.pred.90s <- as.data.frame(VP2.pred.90s)
VP2.pred.00s <- predict(salm.lo.vp2, data.frame(Year = seq(2001, 2010, 1)))
VP2.pred.00s <- as.data.frame(VP2.pred.00s)
VP2.pred.10s <- predict(salm.lo.vp2, data.frame(Year = seq(2010, 2014, 1)))
VP2.pred.10s <- as.data.frame(VP2.pred.10s)
wilcox.test(VP2.pred.80s$VP2.pred.80s, VP2.pred.90s$VP2.pred.90s) #p-value = 2.165e-05
wilcox.test(VP2.pred.90s$VP2.pred.90s, VP2.pred.00s$VP2.pred.00s) # p-value = 0.315
wilcox.test(VP2.pred.00s$VP2.pred.00s, VP2.pred.10s$VP2.pred.10s) #p-value = 0.2203
```

```
#VP3
VP3.pred.80s <- predict(salm.lo.vp3, data.frame(Year = seq(1982, 1990, 1)))
VP3.pred.80s <- as.data.frame(VP3.pred.80s)
VP3.pred.90s <- predict(salm.lo.vp3, data.frame(Year = seq(1991, 2000, 1)))
VP3.pred.90s <- as.data.frame(VP3.pred.90s)
VP3.pred.00s <- predict(salm.lo.vp3, data.frame(Year = seq(2001, 2010, 1)))
VP3.pred.00s <- as.data.frame(VP3.pred.00s)
VP3.pred.10s <- predict(salm.lo.vp3, data.frame(Year = seq(2010, 2014, 1)))
VP3.pred.10s <- as.data.frame(VP3.pred.10s)
wilcox.test(VP3.pred.80s$VP3.pred.80s, VP3.pred.90s$VP3.pred.90s) #p-value = 0.0006495
wilcox.test(VP3.pred.90s$VP3.pred.90s, VP3.pred.00s$VP3.pred.00s) # p-value = 1.083e-05
wilcox.test(VP3.pred.00s$VP3.pred.00s, VP3.pred.10s$VP3.pred.10s) #p-value = 0.01985

#VP4
VP4.pred.80s <- predict(salm.lo.vp4, data.frame(Year = seq(1982, 1990, 1)))
VP4.pred.80s <- as.data.frame(VP4.pred.80s)
VP4.pred.90s <- predict(salm.lo.vp4, data.frame(Year = seq(1991, 2000, 1)))
VP4.pred.90s <- as.data.frame(VP4.pred.90s)
VP4.pred.00s <- predict(salm.lo.vp4, data.frame(Year = seq(2001, 2010, 1)))
VP4.pred.00s <- as.data.frame(VP4.pred.00s)
VP4.pred.10s <- predict(salm.lo.vp4, data.frame(Year = seq(2010, 2014, 1)))
VP4.pred.10s <- as.data.frame(VP4.pred.10s)
wilcox.test(VP4.pred.80s$VP4.pred.80s, VP4.pred.90s$VP4.pred.90s) #p-value = 2.165e-05
wilcox.test(VP4.pred.90s$VP4.pred.90s, VP4.pred.00s$VP4.pred.00s) # p-value = 0.1051
wilcox.test(VP4.pred.00s$VP4.pred.00s, VP4.pred.10s$VP4.pred.10s) #p-value = 0.003261

#VP5
VP5.pred.80s <- predict(salm.lo.vp5, data.frame(Year = seq(1982, 1990, 1)))
VP5.pred.80s <- as.data.frame(VP5.pred.80s)
VP5.pred.90s <- predict(salm.lo.vp5, data.frame(Year = seq(1991, 2000, 1)))
VP5.pred.90s <- as.data.frame(VP5.pred.90s)
VP5.pred.00s <- predict(salm.lo.vp5, data.frame(Year = seq(2001, 2010, 1)))
VP5.pred.00s <- as.data.frame(VP5.pred.00s)
VP5.pred.10s <- predict(salm.lo.vp5, data.frame(Year = seq(2010, 2014, 1)))
VP5.pred.10s <- as.data.frame(VP5.pred.10s)
wilcox.test(VP5.pred.80s$VP5.pred.80s, VP5.pred.90s$VP5.pred.90s) #p-value = 2.165e-05
wilcox.test(VP5.pred.90s$VP5.pred.90s, VP5.pred.00s$VP5.pred.00s) # p-value = 0.002879
wilcox.test(VP5.pred.00s$VP5.pred.00s, VP5.pred.10s$VP5.pred.10s) #p-value = 0.003261
#RdRP
RDRP.pred.80s <- predict(salm.lo.rdrp, data.frame(Year = seq(1982, 1990, 1)))
RDRP.pred.80s <- as.data.frame(RDRP.pred.80s)
RDRP.pred.90s <- predict(salm.lo.rdrp, data.frame(Year = seq(1991, 2000, 1)))
RDRP.pred.90s <- as.data.frame(RDRP.pred.90s)
RDRP.pred.00s <- predict(salm.lo.rdrp, data.frame(Year = seq(2001, 2010, 1)))
RDRP.pred.00s <- as.data.frame(RDRP.pred.00s)
RDRP.pred.10s <- predict(salm.lo.rdrp, data.frame(Year = seq(2010, 2014, 1)))
RDRP.pred.10s <- as.data.frame(RDRP.pred.10s)
wilcox.test(RDRP.pred.80s$RDRP.pred.80s, RDRP.pred.90s$RDRP.pred.90s) #p-value = 2.165e-05
wilcox.test(RDRP.pred.90s$RDRP.pred.90s, RDRP.pred.00s$RDRP.pred.00s) # p-value = 0.4359
wilcox.test(RDRP.pred.00s$RDRP.pred.00s, RDRP.pred.10s$RDRP.pred.10s) #p-value = 0.003261
```