

# Perl Project Report

---

CODON ANALYZER | GC ANALYZER | H-PLOT

Abhinav Mishra | 131503 || Namit Bhagwanani | 131565  
JUIT | 6<sup>TH</sup> SEMESTER

# Perl Project Report

*CODON ANALYZER | GC ANALYZER | H-PLOT*

## Codon Analyzer

Reads in a DNA Sequence file (only DNA sequence no header) containing one or several open reading frames pasted as one piece of DNA without intergenic regions and generates a list with the occurrence (in %) of (user defined) rare codons. The script checks only for rarely used E. coli codons. In order to analyze for other rare codons, the script has to be modified.

### INPUT:

Escherichia coli O157:H7 str. **Sakai** DNA, complete genome

5,498,450 bp circular DNA

Accession: BA000007.2 | GI:47118301

**USGAGE:** CodonAnalyzer.pl INPUT.txt (or. fasta)

**OUTPUT:** Codon\_Percent\_Values.txt

**GC-content** (or **guanine-cytosine content**) is the percentage of nitrogenous bases on a DNA molecule that are either guanine or cytosine (from a possibility of four different ones, also including adenine and thymine). DNA with high GC-content is more stable than DNA with low GC-content; however, the hydrogen bonds do not stabilize the DNA significantly, and stabilization is due mainly to stacking interactions. However, it has been shown that there is a strong correlation between the prokaryotic optimal growth at higher temperatures and the GC content of structured RNAs (such as ribosomal RNA, transfer RNA, and many other non-coding RNAs). GC content is usually expressed as a percentage value, but sometimes as a ratio (called **GC-ratio**).

GC-content percentage (%) is calculated as:

$$\frac{G + C}{A + G + C + T}$$

Since, the sequence from NCBI is sequenced, We used simple arithmetic in my code to calculate **GC Content**. These variations in GC ratio within the genomes of more complex organism's result in a mosaic-like formation with islet regions called *isochores*. This results in the variations in staining intensity in the chromosomes. GC-rich isochores include in them many protein coding genes, and thus determination of ratio of these specific regions contributes in mapping gene-rich regions of the genome.

**Source Code:**

```
#!/usr/bin/perl

#Declare variables and Input Data
$/ = '\777';
# entire input to be read in one slurp
my $ORFs = <>;

# read input, assigning to single string
my $length = length($ORFs);
my $codoncount = $length / 3;
my $lORFs = lc($ORFs);

#Print Input Data
print "\n\n";
print "Input data: \n\n";
print "sequence length: $length \n";
print "number codons: $codoncount \n";
print "query sequence: $ORFs \n\n";

#Split Sequence in Codons
my @sequence = split( "", $lORFs );
my @codons = ( $lORFs =~ m/.../g );

#Initialize Rare Codon Counts
my $count_of_g = 0;
my $count_of_c = 0;
my $count_of_gga = 0;
my $count_of_ggt = 0;
my $count_of_cct = 0;
my $count_of_gca = 0;
my $count_of_gta = 0;
my $count_of_tta = 0;
my $count_of_cta = 0;
my $count_of_ata = 0;
my $count_of_att = 0;
my $count_of_tgt = 0;
my $count_of_tac = 0;
my $count_of_aga = 0;
my $count_of_agt = 0;
my $count_of_agc = 0;
my $count_of_tca = 0;
my $count_of_tct = 0;
my $count_of_acg = 0;
my $count_of_aca = 0;
```

```

#Count Codons
foreach my $nuc (@sequence) {
    if ( $nuc eq 'g' ) { ++$count_of_g; }
    if ( $nuc eq 'c' ) { ++$count_of_c; }
}

foreach my $triplet (@codons) {
    if ( $triplet eq 'gga' ) { ++$count_of_gga; }
    if ( $triplet eq 'ggg' ) { ++$count_of_ggt; }
    if ( $triplet eq 'cct' ) { ++$count_of_cct; }
    if ( $triplet eq 'gca' ) { ++$count_of_gca; }
    if ( $triplet eq 'gta' ) { ++$count_of_gta; }
    if ( $triplet eq 'tta' ) { ++$count_of_tta; }
    if ( $triplet eq 'cta' ) { ++$count_of_cta; }
    if ( $triplet eq 'ata' ) { ++$count_of_ata; }
    if ( $triplet eq 'att' ) { ++$count_of_att; }
    if ( $triplet eq 'tgt' ) { ++$count_of_tgt; }
    if ( $triplet eq 'tac' ) { ++$count_of_tac; }
    if ( $triplet eq 'aga' ) { ++$count_of_aga; }
    if ( $triplet eq 'agt' ) { ++$count_of_agt; }
    if ( $triplet eq 'agc' ) { ++$count_of_agc; }
    if ( $triplet eq 'tca' ) { ++$count_of_tca; }
    if ( $triplet eq 'tct' ) { ++$count_of_tct; }
    if ( $triplet eq 'acg' ) { ++$count_of_acg; }
    if ( $triplet eq 'aca' ) { ++$count_of_aca; }
    if ( $triplet eq 'act' ) { ++$count_of_act; }

    #else { print "\n"; }
}

#Compute percentage of rare codons
my $percent_gga = ( ( $count_of_gga / $codoncount ) * 100 );
my $percent_gga = sprintf( "%.2f", $percent_gga );
my $percent_ggt = ( ( $count_of_ggt / $codoncount ) * 100 );
my $percent_ggt = sprintf( "%.2f", $percent_ggt );
my $percent_cct = ( ( $count_of_cct / $codoncount ) * 100 );
my $percent_cct = sprintf( "%.2f", $percent_cct );
my $percent_gca = ( ( $count_of_gca / $codoncount ) * 100 );
my $percent_gca = sprintf( "%.2f", $percent_gca );
my $percent_gta = ( ( $count_of_gta / $codoncount ) * 100 );
my $percent_gta = sprintf( "%.2f", $percent_gta );
my $percent_tta = ( ( $count_of_tta / $codoncount ) * 100 );
my $percent_tta = sprintf( "%.2f", $percent_tta );
my $percent_cta = ( ( $count_of_cta / $codoncount ) * 100 );
my $percent_cta = sprintf( "%.2f", $percent_cta );
my $percent_ata = ( ( $count_of_ata / $codoncount ) * 100 );
my $percent_ata = sprintf( "%.2f", $percent_ata );

```

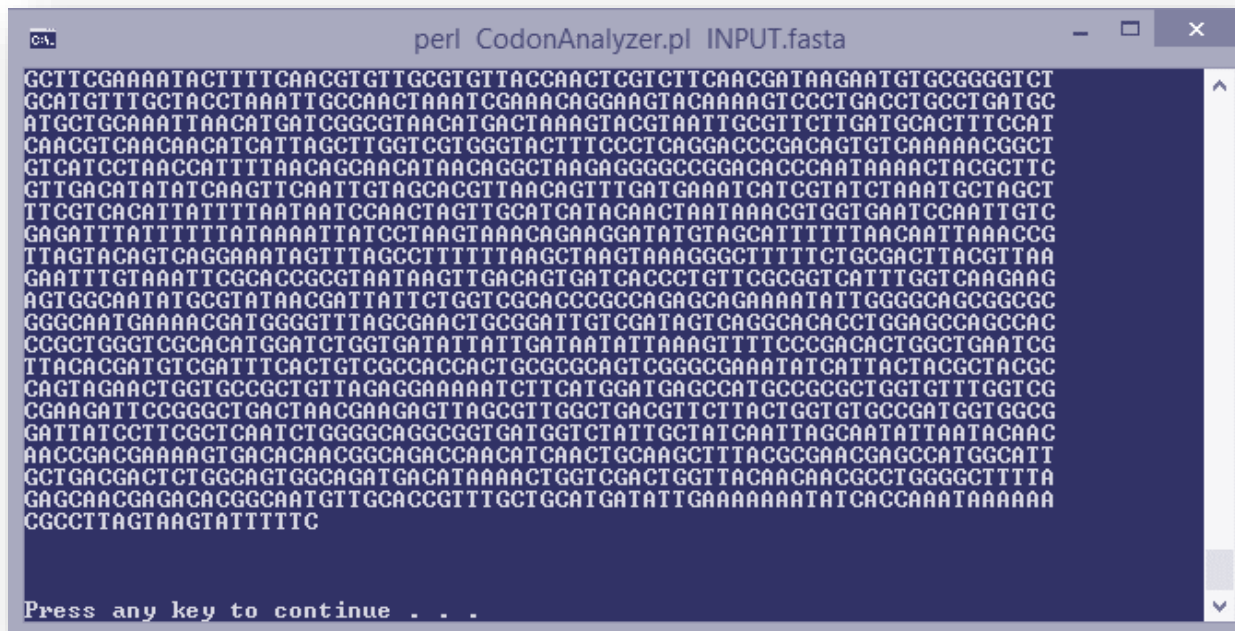
```
my $percent_att = (( $count_of_att / $codoncount ) * 100 );
my $percent_att = sprintf( "%.2f", $percent_att );
my $percent_tgt = (( $count_of_tgt / $codoncount ) * 100 );
my $percent_tgt = sprintf( "%.2f", $percent_tgt );
my $percent_tac = (( $count_of_tac / $codoncount ) * 100 );
my $percent_tac = sprintf( "%.2f", $percent_tac );
my $percent_aga = (( $count_of_aga / $codoncount ) * 100 );
my $percent_aga = sprintf( "%.2f", $percent_aga );
my $percent_agt = (( $count_of_agt / $codoncount ) * 100 );
my $percent_agt = sprintf( "%.2f", $percent_agt );
my $percent_agc = (( $count_of_agc / $codoncount ) * 100 );
my $percent_agc = sprintf( "%.2f", $percent_agc );
my $percent_tca = (( $count_of_tca / $codoncount ) * 100 );
my $percent_tca = sprintf( "%.2f", $percent_tca );
my $percent_tct = (( $count_of_tct / $codoncount ) * 100 );
my $percent_tct = sprintf( "%.2f", $percent_tct );
my $percent_acg = (( $count_of_acg / $codoncount ) * 100 );
my $percent_acg = sprintf( "%.2f", $percent_acg );
my $percent_aca = (( $count_of_aca / $codoncount ) * 100 );
my $percent_aca = sprintf( "%.2f", $percent_aca );
my $percent_act = (( $count_of_act / $codoncount ) * 100 );
my $percent_act = sprintf( "%.2f", $percent_act );
my $gc_content = ((( $count_of_g + $count_of_c ) / $length ) * 100 );
my $gc_content = sprintf( "%.2f", $gc_content );

#Print Results
open( MYFILE, '>>Codon_Percent_Values.txt' );
print MYFILE "\n\n";
print MYFILE "Input data: \n\n";
print MYFILE "sequence length: $length \n";
print MYFILE "number codons: $codoncount \n";
print MYFILE "query sequence: $ORFs \n\n";
print MYFILE "Results: \n\n";
print MYFILE "GC content: $gc_content \n";

print MYFILE
"GGA = $count_of_gga codon(s)... equals to $percent_gga % of all codons \n";
print MYFILE
"GGT = $count_of_ggt codon(s)... equals to $percent_ggt % of all codons \n";
print MYFILE
"CCT = $count_of_cct codon(s)... equals to $percent_cct % of all codons \n";
```

```
print MYFILE
"GCA = $count_of_gca codon(s)... equals to $percent_gca % of all codons \n";
print MYFILE
"GTA = $count_of_gta codon(s)... equals to $percent_gta % of all codons \n";
print MYFILE
"TTA = $count_of_tta codon(s)... equals to $percent_tta % of all codons \n";
print MYFILE
"CTA = $count_of_cta codon(s)... equals to $percent_cta % of all codons \n";
print MYFILE
"ATA = $count_of_ata codon(s)... equals to $percent_ata % of all codons \n";
print MYFILE
"ATT = $count_of_att codon(s)... equals to $percent_att % of all codons \n";
print MYFILE
"TGT = $count_of_tgt codon(s)... equals to $percent_tgt % of all codons \n";
print MYFILE
"TAC = $count_of_tac codon(s)... equals to $percent_tac % of all codons \n";
print MYFILE
"AGA = $count_of_aga codon(s)... equals to $percent_aga % of all codons \n";
print MYFILE
"AGT = $count_of_agt codon(s)... equals to $percent_agt % of all codons \n";
print MYFILE
"AGC = $count_of_agc codon(s)... equals to $percent_agc % of all codons \n";
print MYFILE
"TCA = $count_of_tca codon(s)... equals to $percent_tca % of all codons \n";
print MYFILE
"TCT = $count_of_tct codon(s)... equals to $percent_tct % of all codons \n";
print MYFILE
"ACG = $count_of_acg codon(s)... equals to $percent_acg % of all codons \n";
print MYFILE
"ACA = $count_of_aca codon(s)... equals to $percent_aca % of all codons \n";
print MYFILE
"ACT = $count_of_act codon(s)... equals to $percent_act % of all codons \n";
close MYFILE;
```

**Output Window:**



```
perl CodonAnalyzer.pl INPUT.fasta
1 GCTTCGAAATACTTTTCAACGCTGTTGCGTGTACCAACTCGTCTTCAACGATAAGAAATGTGCGGGGTCT
2 GCAIGTTTGCTACCTAAATTCGCAACTAAATCGAAACAGGAAGTACAAAAGTCCCTGACCTGCCTGATGC
3 ATGCTGCAAAATTAACATGATCGGCGTAACATGACTAAAGTACGTAATTGCGTTCTTGATGCACTTTCAT
4 CAACGTCAAACAACATCATTAGCTTGGTGGTGGTACTTTCCCTCAGGACCCGACAGTGTCAAAAACGGCT
5 GTCATCCCTAACCATTTTAAACAGCAACATAACAGGCTAAGAGGGGGCCGGACCCCAATAAACTACGCTTC
6 GTTGACATATATCAAGTTCATTGTAGCACGTTAACAGTTTGATGAAATCATCGTATCTAAATGCTAGCT
7 TTCGTCACATTATTTTAAATAATCCAAGTAGTTGCATCATACAATAATAAACGTTGGTGAATCCAATTGTC
8 GAGATTTATTTTTATAAATTTATCCTAAGTAAACAGAAGGATATGTAGCATTITTTTAACAATTAAACCG
9 TTAGTACAGTCAGGAATAGTTAGCCTTTTTTAAAGCTAAGTAAAGGGCTTTTTCTGCGACTTACGTTAA
10 GAATTTGTAATTCGCACCGCGTAATAAGTIGACAGTGATCACCCGTGTCGCGGTCATTGGTCAAGAAG
11 AGTGGCAATATGCGTATAACGATTATTCTGGTCGCACCCGCCAGAGCAGAAAATATTGGGGCAGCGGGCG
12 GGGCAATGAAAACGATGGGGTTTAGCGAAGTGGCGATTGTGATAGTCAGGCACACCTGGAGCCAGCCAC
13 CCGCTGGGTGCGACATGGATCTGGTGATATTATTGATAATATTAAAGTTTCCCGACACTGGCTGAATCG
14 TTACACGATGTCGATTTCAGTGTGCCACCACTGGCGCGAGTCGGCGGAATATCATTACTACGCTACGC
15 CAGTAGAACTGGTGCCGCTGTAGAGGAAAAATCTTCATGGATGAGCCATGCCGCGCTGGTGTGTTGGTCG
16 CGAAGATTCGGGGCTGACTAACGAAGAGTTAGCGTTGGCTGACGTTCTTACTGGTGTGCCGATGGTGGCG
17 GATTATCCTTCGCTCAATCTGGGGCAGGCGGTGATGGTCTATTGCTATCAATTAGCAATATTAATACAAC
18 AACCGACGAAAAGTGACACAACGGCAGACCAACATCAACTGCAAGCTTTACGCGAACGAGCCATGGCATT
19 GCTGACGACTCTGGCAGTGGCAGATGACATAAACTGGTCGACTGGTTACAACAACGCCTGGGGCTTTTA
20 GAGCAACGAGACACGGCAATGTTGCACCGTTTGTGATGATATTGAAAAAATATACCAAAATAAAAAA
21 CGCCTTAGTAAAGTATTTTC
Press any key to continue . . .
```

Fig 1. Reading the Query data | CodonAnalyzer.pl

**Output File:**

```

Input data:

sequence length: 5577001
number codons: 1859000.33333333
query sequence: agcttttcattctgactgcaacgggcaatatgtctctgtgtggattaaaaaaag:
ttctgaactgggttacctgcccgtgagtaaaattttattgacttaggtcactaaatactttaaccaa
tataggcatagcgacagacagataaaaaattacagagtacacaacatccatgaaacgcattagcaccacc
attaccaccaccatcaccaccaccatcaccattaccattaccacaggtaacgggtgcggtgacgcgtac
aggaaacacagaaaaaagcccgcacctgacagtgcggtctttttttcgaccaaaaggtaacgaggtaca
accatgagtggttgaagttcggtggtacatcagtggaatgcagaacgttttctgcggttgccgata
ttctggaagcaatgccaggcaggggaggtggccaccgtcctctctgccccgcgcaaaatcaccaacca
cctggtgagtgatgattgaaaaaacattagcggtcaggtgctttaccaaatatcagcgatgccgaacgt
atttttgccaacttctgacgggactcgccgcgcccagcgggattcccgctggcgcaattgaaaactt
tcgtgcgaccaggaatttgcgaataaaacatgtcctgcattggcattagttgttagggcagtgcccgga
tagcattaacgctgagctgatttgcgtggcgagaaaaatgtcgatcgccattatggcggcggtattagaa
gctgcgggtcacacgtttaccgttatcgatccggtcgaaaaactgctggcagtgggcgattacctcgaat
ctactgtcgatattgcagagtcaccccgcggtattgcggaagtgcgtattccggctgatcacatggtgct
gatggcaggtttcacgcgggtaataaaaaaggcgaactggtggtacttggaacgcaacggttcgactac
tcgcggcggtgctggtgcctgtttacgcgcggtattgttgcgagatttggaacggacgttgacgggtat
atacctgcgacccgctcaggtgcccgcgaggttgttgaatcgatgtcctaccaggaagcgatgga
gctttcctacttcggcgctaaagtcttccaccccgccaccattaccccatcgccaggttcagatccct
tgctgattaaaaataccggaatcctcaagctccaggtacgctcattggtgccagtcgtgatgaagacg

```

Fig 2. Beginning of Codon\_Percent\_Values.txt

```

Results:

GC content: 49.83
GGA = 23069 codon(s)... equals to 1.24 % of all codons
GGT = 28431 codon(s)... equals to 1.53 % of all codons
CCT = 20040 codon(s)... equals to 1.08 % of all codons
GCA = 36941 codon(s)... equals to 1.99 % of all codons
GTA = 20694 codon(s)... equals to 1.11 % of all codons
TTA = 27112 codon(s)... equals to 1.46 % of all codons
CTA = 10291 codon(s)... equals to 0.55 % of all codons
ATA = 25444 codon(s)... equals to 1.37 % of all codons
ATT = 32939 codon(s)... equals to 1.77 % of all codons
TGT = 23466 codon(s)... equals to 1.26 % of all codons
TAC = 20370 codon(s)... equals to 1.10 % of all codons
AGA = 22245 codon(s)... equals to 1.20 % of all codons
AGT = 19668 codon(s)... equals to 1.06 % of all codons
AGC = 31323 codon(s)... equals to 1.68 % of all codons
TCA = 33334 codon(s)... equals to 1.79 % of all codons
TCT = 21877 codon(s)... equals to 1.18 % of all codons
ACG = 28072 codon(s)... equals to 1.51 % of all codons
ACA = 23350 codon(s)... equals to 1.26 % of all codons
ACT = 19601 codon(s)... equals to 1.05 % of all codons

```

Fig 3. Bottom of Codon\_Percent\_Values.txt



## GC Analyzer

Reads in a DNA Sequence file (only DNA sequence, no header) and generates a list with **mean GC Content per user defined window**.

### **INPUT :**

Escherichia coli O157:H7 str. **Sakai** DNA, complete genome

5,498,450 bp circular DNA

Accession: BA000007.2 | GI:47118301

**USGAGE:** GC\_Analyzer.pl INPUT.txt (or. fasta)

**OUTPUT:** LocalGC.txt

GC- Content is often used as one of the markers to reveal the events of Horizontal Gene Transfer (HGT). We used traditional Fixed Length Window Analysis of [G+C] % that strongly depends on the arbitrarily on the selection of DNA Window length.

DNA sequences are generally not random sequences. To show this non- randomness visually, it is plotted as moving averages for certain length of window (in this case, it is '10 bp' long) (See Source Code) that slid along a sequence. We inferred from results (LocalGC.txt), GC- Content is NOT same for all sequence regions. Moreover, this contributes to Thermo-stability of DNA Double Helix using Base- stacking and Base- pairing. Differential contribution of base-stacking in AT and GC-containing contacts is responsible for 50% of the dependence of DNA stability on its GC- content.

**Source Code:**

```
#!/usr/bin/perl

#Declare variables and Input Data
$/ = '\777';

# entire input to be read in one slurp
my $ORFs = <>;

# read input, assigning to single string
my $window = 10;

# define sequence window, e.g averaging over 10 bp
my $length = length($ORFs);
my $codoncount = $length / $window;

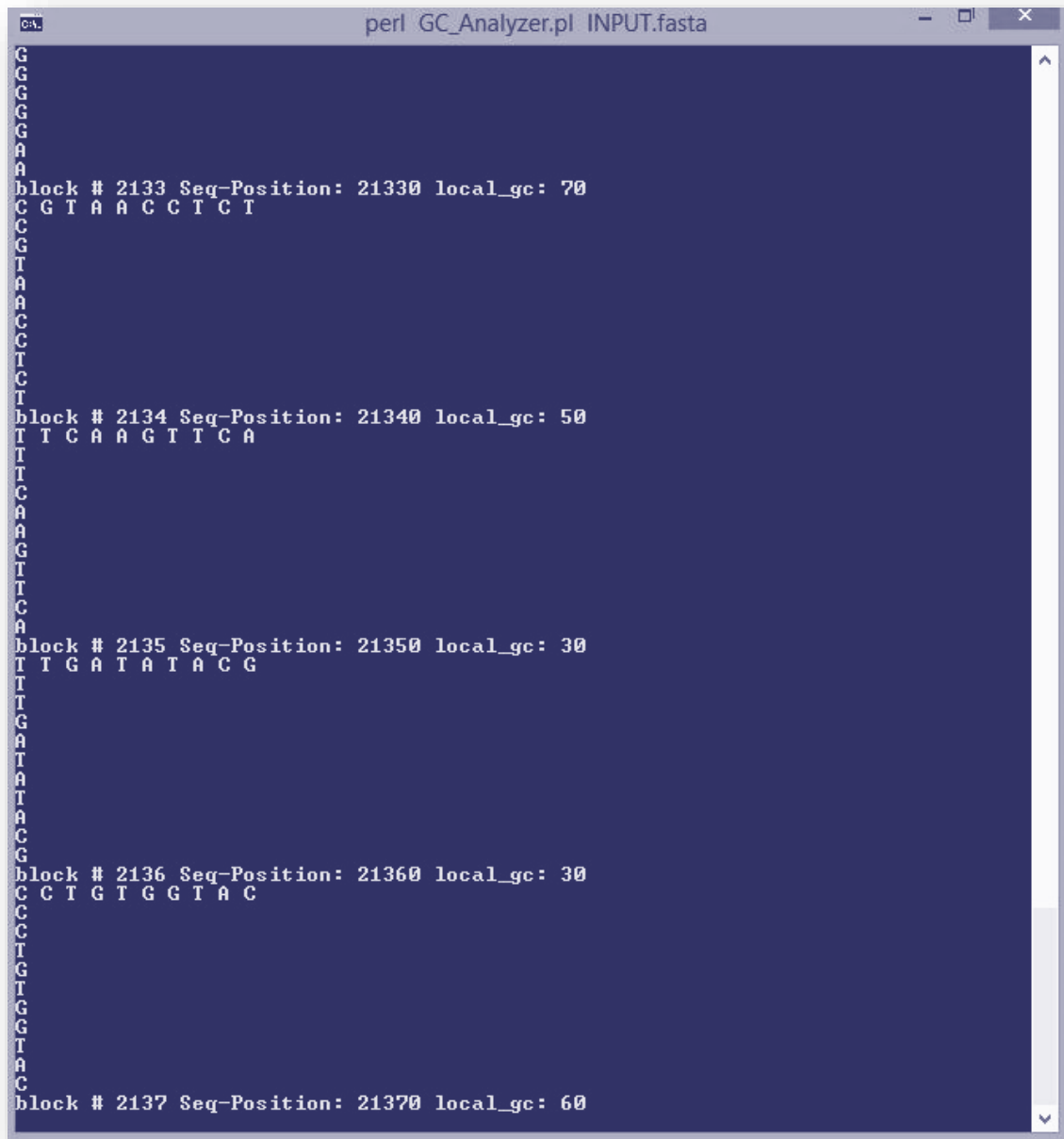
#Print Input Data Statistics
print "\n\n";
print "Input data: \n\n";
print "sequence length: $length \n";
print "number codons: $codoncount \n";

#Initialize Counts
my $count_of_g = 0;
my $count_of_c = 0;
my $count_g = 0;
my $count_c = 0;
my $i = 0;
my $pos = 0;

#Split Sequence in Codons
my @sequence = split(" ", $ORFs);
my @codons = ( $ORFs =~ m/...../g );
#Count Codons
foreach my $nuc (@sequence) {
    if ( $nuc eq 'G' or $nuc eq 'g' ) { ++$count_of_g; }
    if ( $nuc eq 'C' or $nuc eq 'c' ) { ++$count_of_c; }
}
```

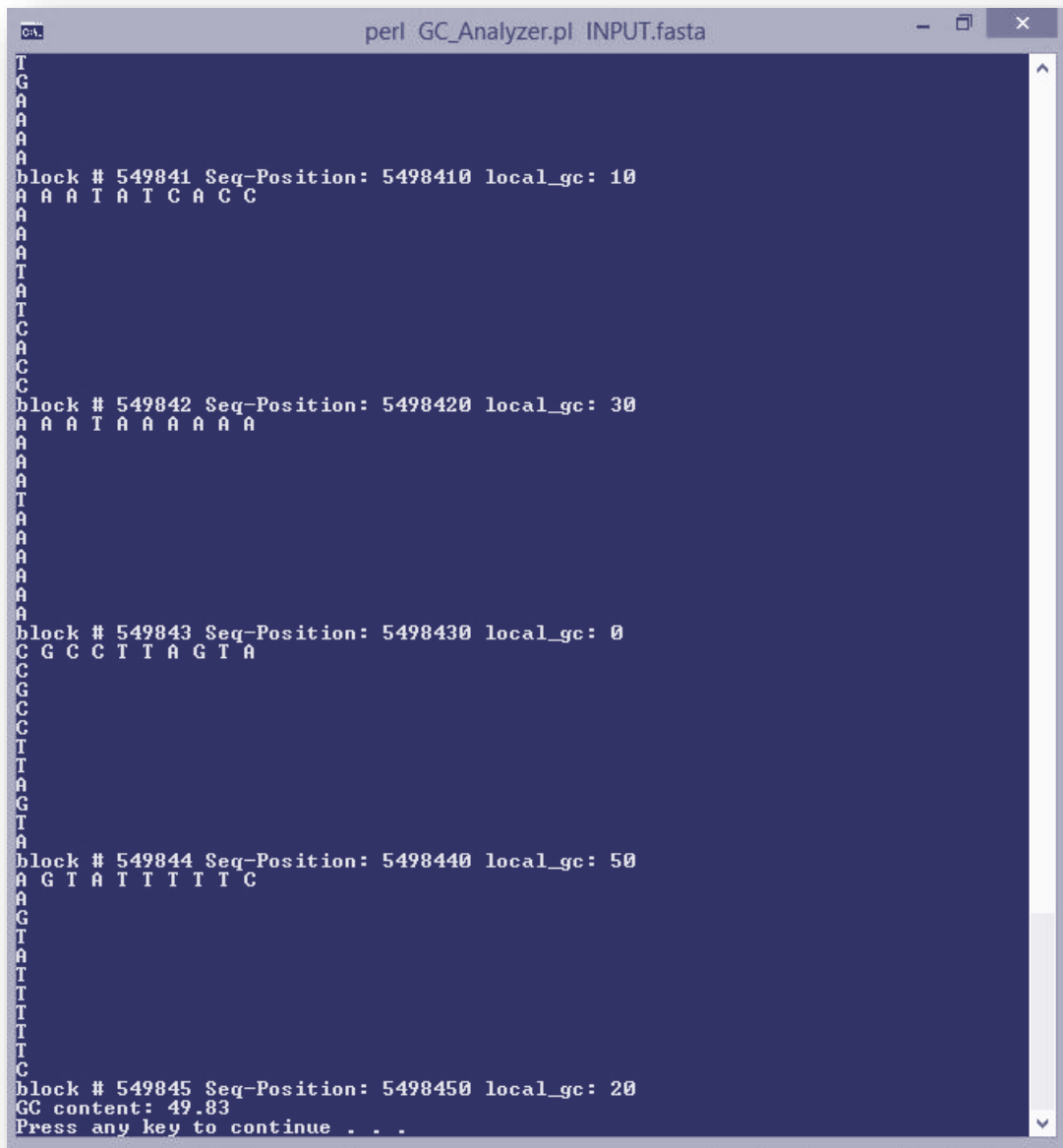
```
# derive GC content
my $gc_content = ((( $count_of_g + $count_of_c ) / $length ) * 100 );
$gc_content = sprintf( "%.2f", $gc_content );

#Count over Window and analyse Local GC
foreach my $block1 (@codons){
    $i = $i + 1;
    $pos = $i * $window;
    my @seqBlock = split( ", $block1 );
    print "@seqBlock\n";
    foreach my $base (@seqBlock){
        if ( $base eq 'G' ) { ++$count_g }
        if ( $base eq 'C' ) { ++$count_c }
        print "$base\n";
    }
    my $local_gc = ((( $count_g + $count_c ) / $window ) * 100 );
    print "block # $i Seq-Position: $pos local_gc: $local_gc\n";
    open( MYFILE, '>>LocalGC.txt' );
    print MYFILE "$i $pos $local_gc\n";
    close(MYFILE);
    $count_g = 0;
    $count_c = 0;
}
print "GC content: $gc_content\n";
exit;
```

**Output Window:**

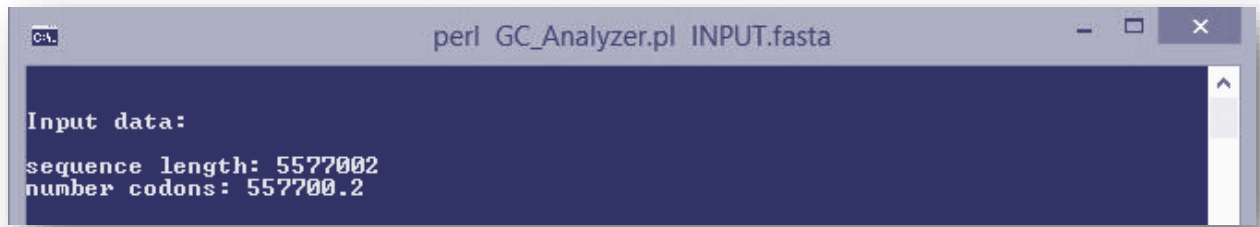
```
perl GC_Analyzer.pl INPUT.fasta
G
G
G
G
G
A
A
block # 2133 Seq-Position: 21330 local_gc: 70
C G T A A C C T C T
C
C
G
T
A
A
C
C
T
C
T
block # 2134 Seq-Position: 21340 local_gc: 50
T T C A A G T T C A
T
T
C
A
A
G
T
T
C
A
block # 2135 Seq-Position: 21350 local_gc: 30
T T G A T A T A C G
T
T
G
A
T
A
T
A
C
G
block # 2136 Seq-Position: 21360 local_gc: 30
C C T G T G G T A C
C
C
C
T
G
T
G
T
A
C
block # 2137 Seq-Position: 21370 local_gc: 60
```

**Fig 4.** Reading blocks of window size '10' |GC\_Analyzer.pl



```
perl GC_Analyzer.pl INPUT.fasta
T
G
A
A
A
A
A
block # 549841 Seq-Position: 5498410 local_gc: 10
A A A T A T C A C C
A
A
A
A
T
A
T
C
A
C
C
block # 549842 Seq-Position: 5498420 local_gc: 30
A A A T A A A A A
A
A
A
A
T
A
A
A
A
A
A
block # 549843 Seq-Position: 5498430 local_gc: 0
C G C C T T A G T A
C
C
G
C
C
T
T
A
G
T
A
block # 549844 Seq-Position: 5498440 local_gc: 50
A G T A T T T T T C
A
G
T
A
T
T
T
T
T
C
block # 549845 Seq-Position: 5498450 local_gc: 20
GC content: 49.83
Press any key to continue . . .
```

Fig 5. Displayed GC- Content, same as of Fig 3. |Running Time was approx. 30 minutes |High Complexity but More Accuracy.



```
perl GC_Analyzer.pl INPUT.fasta

Input data:
sequence length: 5577002
number codons: 557700.2
```

**Fig 6.** Length and Codon Numbers | GC\_Analyzer.pl

**Output File:**

```
1 10 30
2 20 50
3 30 50
4 40 50
5 50 20
....
....
....
....
549840 5498400 50
549841 5498410 10
549842 5498420 30
549843 5498430 0
549844 5498440 50
549845 5498450 20
```

**Fig 7.** Block number,  
Position and Local  
GC Content (Left to  
Right) columns |

Local GC.txt

## H- PLOT

Reads in txt-file in fasta format containing unaligned amino acid sequences. It Assigns a hydropathy "value" according to the **Kyte-Doolittle scale** to each amino acid and writes a file that contains an ordered space separated list of hydropathy values for each sequence in the fasta file. The outfile can easily be imported into any data analysis software such as MS excel, Origin, GraphPadprism.

**INPUT:** Translated AA Sequences of two files using Lab 6 (Translation)

1. Escherichia coli O157:H7 str. **Sakai** DNA, complete genome

5,498,450 bp circular DNA

Accession: BA000007.2 | GI:47118301

2. Escherichia coli CFT073, complete genome

5,231,428 bp circular DNA

Accession: AE014075.1 | GI: 26111730

Note: We used Codon Table for Bacterial types. Appended the sequences in one file “**Trans\_Prot.txt**” as:

*>Sequence 1*

.....

.....

*>Sequence 2*

.....

.....

**USAGE:** H\_Plot.pl Trans\_Prot.txt (or. fasta)

**OUTPUT:** H\_PLOT.txt

**A C program for evaluating the hydropathic character of sequence segments (used by SOAP)[See Table 1]:**

```

main()
{
int i, j, k;;
float total;
char residue;
extern char code  [23];
extern float factor [23];
char sequence  [1099];
float value    [1099];
j = 0;
while (getchar != '\n');
while (j <1099)
{
for (i = 0; i <11; i++)
getchar();
while (j <1099)
{
sequence[j++] = getchar();
if (getchar == '\n')
break;
}
if (sequence[j - 1] == '\n')
break;
}
j = (j - 1);
for (i = 0; i <j; i++)
{
residue = sequence[i];
for (k = 0; k <23; k++)
if(residue == code[k])
value[i] = factor[k];
}
for (i = 0; i <(j - 6); i++)
{
total = 0;
for (k = 0; k <7; k++)
total = total + value[i + k];
printf("84d $c $6.lf", i + 4, sequence[i+31, total);
for (k = 0; k <total; k++) {if(k == 29)
printf(". ");
else
printf ( " ");}
printf ( "X\n" )
};
printf("\n"); }
char code[] "RKDENSEHZQTGXAPVYCMILWF" ;
float factor[] [0.0,0.6,1.0,1.0,1.0,3.6,1.0,1.3,1.0,1.0,3.8,4.1,4.
1,6.3,2.9,8.7,3.2,7.0,6.4,9.0,5.2,3.6,7.2.1);
}

```



Amino Acid	One Letter Code	Hydropathy Score
Isoleucine	I	4.5
Valine	V	4.2
Leucine	L	3.8
Phenylalanine	F	2.8
Cysteine	C	2.5
Methionine	M	1.9
Alanine	A	1.8
Glycine	G	-0.4
Threonine	T	-0.7
Tryptophan	W	-0.9
Serine	S	-0.8
Tyrosine	Y	-1.3
Proline	P	-1.6
Histidine	H	-3.2
Glutamic acid	E	-3.5
Glutamine	Q	-3.5
Aspartic acid	D	-3.5
Asparagine	N	-3.5
Lysine	K	-3.9
Arginine	R	-4.5

**Table 1.** Kyte- Dolittle Scale ( Amino Acid Hydropathy Values)

The scale is based on an amalgam of experimental observations derived from the literature. The program uses a moving-segment approach that continuously determines the average hydropathy within a segment of predetermined length as it advances through the sequence. The consecutive scores are plotted from the amino to the carboxy terminus. At the same time, a midpoint line is printed that corresponds to the grand average of the hydropathy of the amino acid compositions found in most of the sequenced proteins. They used a computer program called SOAP as experimental procedure and NEWAT as the library.

**Source Code:**

```
#!/usr/bin/perl

$/ = '\777';

# entire input to be read in one slurp
$seqs = <>;

# read input, assigning to single string
while ( $seqs =~ m/^(>[^\>]+)/mg ) {

    # match indiv. sequences by '>'s = Fasta Format
    push( @seqs, $1 );

    # and store in array
}
for (@seqs) { # only allow characters A-Z,a-z,o-g,'_','-,' and '.' in names;
    /^> *([^\>-\.,_]+) && ( $seq_name = $1 );
    if ($seq_name) {
        my $seq1 = $_;
        $seq1 =~ s/^>*.+\n//;

        # remove FASTA header
        $seq1 =~ s/\n//g;

        # remove endlines
        my $value = "";
        my $codon = "";
        for ( my $i = 0 ; $i < ( length($seq1) ) ; $i += 1 ) {
            $codon = substr( $seq1, $i, 1 );
            $value .= seq2value($codon);
        }
    }
}
```

```
# write out-file
open( MYFILE, '>>H_PLOT.txt' );
printf MYFILE ( '%1$6s %2$6s %3$6s', $seq_name, $seqI, $value );
print MYFILE "\n";
close(MYFILE);

# on-screen output
print " \n\n$seq_name $value\n\n";
}
else { warn "couldn't recognise the sequence name in \n$_"; }
}
exit;

#sub hydropathy-scale - kyte-doolittle
sub seq2value {
my ($codon) = @_ ;
$codon = uc $codon;
my (%scale) = (
'A' => '1.80 ',
'R' => '-4.50 ',
'N' => '-3.50 ',
'C' => '2.50 ',
'D' => '-3.50 ',
'E' => '-3.50 ',
'F' => '2.80 ',
'G' => '-0.40 ',
'H' => '-3.29 ',
'I' => '4.50 ',
'K' => '-3.90 ',
'L' => '3.80 ',
'M' => '1.90 ',
'P' => '-1.60 ',
'Q' => '-3.50 ',
'S' => '-0.80 ',
'T' => '-0.70 ',
'V' => '4.20 ',
'W' => '-0.90 ',
'Y' => '-1.30 ',
);
if ( exists $scale{$codon} ) {
return $scale{$codon};
}

#else {print STDERR "Bad Amino Acid !!\n";exit;}
}
exit;
```

**Output Window:**

```

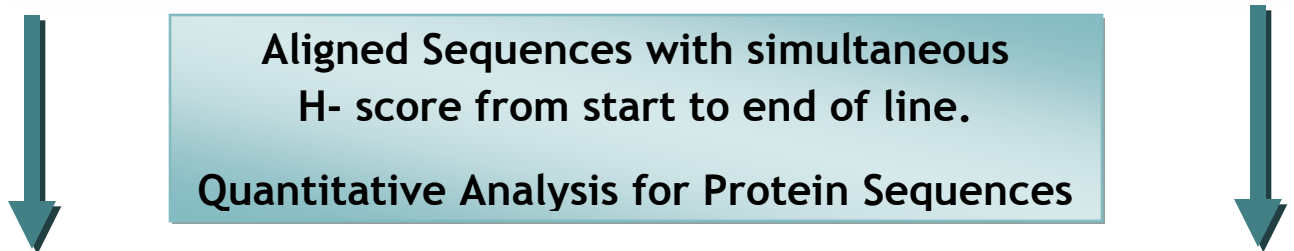
-3.50 2.50 -0.80 -0.70 3.80 -0.70 4.20 -3.50 -3.50 -4.50 4.50 1.90 3.80 1.80 2.8
0 4.20 -0.70 3.80 2.80 -0.80 -3.50 3.80 -3.29 -3.29 -0.70 -0.70 -3.50 -3.90 -1.6
0 -0.40 -3.50 -0.80 -3.50 -1.30 -4.50 -3.50 3.80 2.80 2.80 4.50 -3.90 3.80 -0.80
4.20 -3.50 -4.50 -4.50 4.50 2.50 -0.80 4.50 2.80 -3.50 2.80 -3.50 -4.50 -1.30 -0.
80 -3.50 -3.50 4.50 4.20 -1.60 2.80 3.80 -0.80 1.80 -3.90 -0.40 2.80 -0.80 2.50
-3.50 3.80 -4.50 -3.50 2.80 4.20 -3.50 -0.80 -3.50 -4.50 4.20 4.50 -0.80 -3.50 -
0.80 -1.60 2.50 -0.80 -4.50 3.80 2.80 -0.40 -3.50 -3.50 -3.50 -0.90 -3.50 -1.30 1
.80 -1.30 -3.50 -3.50 -1.30 -0.80 -0.40 2.50 -0.70 -4.50 -3.50 -0.80 -4.50 -3.90 -
1.30 -0.90 -0.40 -0.80 -0.40 1.80 -0.40 -3.50 -3.50 -3.50 -3.50 -0.40 2.80 -4.50
-0.70 1.80 -3.50 2.50 -0.80 -0.40 -0.70 -1.60 -0.40 1.80 -0.80 -4.50 -1.60 3.80
-0.40 -4.50 -0.70 -0.90 4.50 -0.90 -1.30 -1.30 -1.30 -0.80 2.80 -3.50 -3.50 -0.70 -
0.80 4.50 4.20 -0.70 -4.50 2.50 -4.50 2.80 -3.29 2.50 2.50 -3.29 -3.29 2.50 1.80
-3.50 -0.80 -0.40 -3.50 4.50 -0.80 3.80 3.80 -4.50 -1.30 1.80 -0.80 -4.50 -0.70
-0.40 1.80 1.80 4.20 -4.50 -0.40 -3.90 4.50 2.80 1.90 -3.50 -3.50 -1.60 2.50 -4.
50 -0.80 -0.40 4.20 -0.90 -0.80 -4.50 -4.50 2.80 -4.50 1.80 -3.50 -4.50 -4.50 4.
20 -0.80 4.20 -0.40 -4.50 -0.80 -1.30 -0.90 2.50 1.80 -3.50 -0.40 -0.40 -0.40 3.8
0 -0.80 2.80 1.80 -3.50 -0.80 -0.40 1.80 -0.40 -0.80 -3.50 -0.40 3.80 3.80 3.80
-0.80 4.50 -0.80 -3.50 4.50 -3.50 -0.70 -0.70 -0.70 -0.40 -3.50 -3.90 -3.29 -3.5
0 -0.40 -4.50 -1.60 -0.70 -0.80 -0.70 1.80 -0.80 2.80 -0.70 -4.50 -0.70 -0.80 -3
.29 -3.50 4.50 1.80 -3.50 -3.50 -0.80 -0.40 -0.80 -0.40 -4.50 -3.29 -3.90 -0.70
-0.40 -4.50 3.80 4.20 -0.70 -0.70 -0.70 -1.60 -0.40 1.80 2.80 -4.50 1.80 -0.70 -
4.50 -3.29 -0.40 -3.50 4.20 1.80 -1.60 2.80 1.80 1.80 -1.30 -3.90 -3.90 -1.30 -3.2
9 -3.50 4.50 -3.90 -3.50 1.80 3.80 4.20 -0.80 -3.50 2.80
Press any key to continue . . .
  
```

**Fig 8.** H\_PLOT |Hydropathy Mapping to whole sequence | H\_Plot.pl.

**Output File:**

```

3.80 -0.70 1.80 -1.30 -4.50 -4.50 -0.40 3.80 -3.90 1.80 -1.60 3.80 4.20 2.80 -1.30 -0.40 -3.90
1.80 1.80 -0.70 4.20 -3.50 -3.29 4.50 -3.50 -3.50 -4.50 3.80 -3.50 -3.50 4.50 4.20 -0.70 1.90
  
```



```

Sequence SFSFLQRAICLCVDKSLQQLLNWLPVAVSKLKFYLRSLNTLTNIGIAHRQIKITEYTTSMKRISTTITTTITTTITTTITTTGNGAC
Sequence SFSFLQRAICLCVDKSVQLLNWLPVAVSKLKFYLRSLNTLTNIGIAHRQIKITEYTTSMKRISTTITTTITTTITTTGNGAGRVQETQ
  
```

## References:

1. Galtier, N. and J. R. Lobry (1997). "Relationships between genomic G+C content, RNA secondary structures, and optimal growth temperature in prokaryotes." J Mol Evol **44**(6): 632-636.
2. Hurst, L. D. and A. R. Merchant (2001). "High guanine–cytosine content is not an adaptation to high temperature: a comparative analysis amongst prokaryotes." Proceedings of the Royal Society of London B: Biological Sciences **268**(1466): 493-497.
3. Ilatovskiy, A. and M. Petukhov "Genome-wide search for local DNA segments with anomalous GC-content." (1557-8666 (Electronic)).
4. Kyte, J. and R. F. Doolittle (1982). "A simple method for displaying the hydropathic character of a protein." J Mol Biol **157**(1): 105-132.
5. Makino, K., et al. (1999). "Complete nucleotide sequence of the prophage VT2-Sakai carrying the verotoxin 2 genes of the enterohemorrhagic *Escherichia coli* O157:H7 derived from the Sakai outbreak." Genes Genet Syst **74**(5): 227-239.
6. Welch RA, Burland V, Plunkett G, et al. Extensive mosaic structure revealed by the complete genome sequence of uropathogenic *Escherichia coli*. *Proceedings of the National Academy of Sciences of the United States of America*. 2002;99(26):17020-17024. doi:10.1073/pnas.252529799.
7. Šmarda, P., et al. (2014). "Ecological and evolutionary significance of genomic GC content diversity in monocots." Proceedings of the National Academy of Sciences **111**(39): E4096-E4102.
8. Yakovchuk, P., et al. (2006). "Base-stacking and base-pairing contributions into thermal stability of the DNA double helix." Nucleic Acids Research **34**(2): 564-574.
9. Dr. Curtis Jamison, Centre for Biomedical Genomics and Informatics, George Mason University, Virginia, USA." Perl Programming for Bioinformatics & Biologists (2004)". Wiley DreamTech India Pvt. Limited :163-183.

10. Michael Moorhouse, et al., Erasmus MC, The Netherlands. “Bioinformatics, Biocomputing and PERL (2004)”. John Wiley & Sons Ltd.
  11. Rex A. Dwyer, Genomic Perl Consultancy, Inc. “Genomic Perl: From Bioinformatics Basics to Working Code (2003)”. Cambridge University Press.
- 

### Notes:

1. To access codes for these programs, please visit this [link](#). Download all files, put them in one place and then Run them.
2. To access the results, please visit this [link](#).
3. To access the supplementary script for translating Genome (H Plot), please visit this [link](#).