# Processing Biological Sequences in MATLAB

## Introduction to Sequence Analysis

Prof. Gautam B. Singh

## MATLAB Bioinformatics Toolbox

The Bioinformatics Toolbox provides functions for

1. Downloading, Saving and Reading Biological Sequences
2. Creating Biological Sequence Objects
3. Performing a Variety of Sequence Analyses
4. Aligning Sequences using Needle-Wunsch and Smith-Waterman Algorithms
5. Running BLAST
6. Generating Sequence and Pattern Models
7. Storing and Processing Gene Ontologies
8. Performing Phylogenetic Analyses
9. Processing Microarrays and Visualization

## Sequence Acquisition

- MATLAB allows direct retrieval of sequences from sequence database.
- The commands below illustrate retrieval of DNA sequences from GenBank and EMBL, and protein sequences from GenPept.
- Sequence objects are created in each case allowing for a uniform mechanism for processing sequence objects.

```
dnaSeq = getgenbank('M10051');
seqObj = getembl ('X00558');
pepSeq = getgenpept('AAA59174');
```

## Sequence Analysis Functions

| function | Description | Example |
|---|---|---|
| nt2aa | Converts a nucleotide sequence to an amino acid sequence | aaSeq = nt2aa(dnaSeq) |
| aa2nt | Converts an amino acid sequence to a nucleotide sequence | ntSeq = aa2nt(aaSeq) |
| dna2rna | Converts a DNA to an RNA sequence | rnaSeq = dna2rna(dnaSeq) |
| rna2dna | Converts an RNA sequence to DNA sequences | dnaSeq = rna2dna(rnaSeq) |
| seqcomplement | Complementary sequence | seqC = seqcomplement(seq) |
| seqrcomplement | Reverse-complementary sequence | seqRC = seqrcomplement(seq) |
| seqreverse | Sequence orientation reversed | seqR = seqreverse(seq) |
| aacount | Counts frequency of amino acids | aaCnt = aacount(aaSeq) |
| basecount | Counts frequency of nucleotides | ntCnt = basecount(dnaSeq) |
| dimercount | Counts frequency of 2-mers | diCnt = dimercount(dnaSeq) |
| codoncount | Counts frequency of 3-mers | cdnCnt = codoncount(dnaSeq) |
| nmercount | Counts frequency of n-mers | nmerCnt = nmercount(dnaSeq, n) |
| ntdensity | Nucleotide density profile sequence | ntdensity(dnaSeq) |
| codonbias | Compute bias in the usage of codons | codonbias(dnaSeq) |
| cpgislands | Locate stretches of CG dimers or CpG islands | cpgisland(dnaSeq) |
| seqshowwords | Find specific words in sequence | seqshowwords(seq) |
| seqwordcount | Counts words in a sequence | wCnt = seqwordcount(seq) |
| seqshoworfs | Show location of Open Reading Frames (ORFs) | seqshoworfs(dnaSeq) |

## Sequence Representation – components of dnaSeq

```
>>dnaSeq
dnaSeq =
               LocusName: 'HUMINSR'
     LocusSequenceLength: '4723'
    LocusNumberofStrands: ''
           LocusTopology: 'linear'
       LocusMoleculeType: 'mRNA'
    LocusGenBankDivision: 'PRI'
   LocusModificationDate: '06-JAN-1995'
              Definition: 'Human insulin receptor mRNA, complete c
               Accession: 'M10051'
                 Version: 'M10051.1'
                      GI: '186439'
                Keywords: 'insulin receptor; tyrosine kinase.'
                  Source: 'Homo sapiens (human)'
          SourceOrganism: [4x65 char]
               Reference: {[1x1 struct]}
                 Comment: [14x67 char]
```

## Sequence Features

Features are biological annotations on a Sequence object. (*seqObj.Features*). Features have a location and type that provides its functional significance.

```
>> seqObj.Feature

ans =
Key             Location/Qualifiers
source          1..877
                /organism="Rattus norvegicus"
                /mol_type="mRNA"
                /db_xref="taxon:10116"
sig_peptide     33..86
CDS             33..812
                /product="preproapolipoprotein A-I"
                /db_xref="GOA:P04639"
                /db_xref="HSSP:P02647"
                /db_xref="InterPro:IPR000074"
                /db_xref="InterPro:IPR013326"
                /db_xref="UniProtKB/Swiss-Prot:P04639"
                /protein_id="CAA25224.1"
                /translation="MKAAVLAVALVFLTGCQAWEFWQQDEPQSQWDRVKDFATVYVDAV
                KDSGRDYVSQFESSTLGKQLNLNLLDNWDTLGSTVGRLQEQLGPVTQEFWANLEKETDW
                LRNEMNKDLENVKQKMQPHLDEFQEKWNEEVEAYRQKLEPLGTELHKNAKEMQRHLKVV
                AEEFRDRMRVNADALRAKFGLYSDQMRENLAQRLTEIRNHPTLIEYHTKAGDHLRTLGE
                KAKPALDDLGQGLMPVLEAWKAKIMSMIDEAKKKLNA"
mat_peptide     105..812
                /product="apolipoprotein A-I"
polyA_signal    858..863
polyA_site      877..877
```

## Sequence Display
### Example

- The sequence data retrieved from the sequence object, *seqObj.Sequence*
- And displayed *seqdist (. . . )*
- Alternatively, MATLAB provides a GUI of the sequence object with its *seqviewer*

```
>> seqdisp (seqObj.Sequence)

ans =
  1  AGCTCCGGGG GAGGTCGCCC ACATCCTTCG GGATGAAAGC TGCAGTGTTG
 61  TGGTCTTCCT GACAGGTTGC CAAGCTTGGG AGTTCTGGCA GCAAGATGAG
              .....
781  TCGATGAGGC CAAAAAGAAG CTGAACGCTT AGTGAGGCGC CCGTCACCAC
841  TGAATTGGCT TTCTTACAAT AAACGTTTCC AAAGTGG
>>
```

## Operations on Nucleotide Sequences

### Length, and Base Composition of Sequence

- Begin with a string variable initialized to a nucleotide sequence
- *basecount* counts the frequency of bases in sequence

```
ntSeq = 'ACAGTGCCCCCTATATGGCCACCAGGTAG'
ntSeq =
ACAGTGCCCCCTATATGGCCACCAGGTAG
>> length (ntSeq)
ans =
    30
%Find the base frequencies
>> basecount (ntSeq)
ans =
    A: 6
    C: 6
    G: 5
    T: 4
```

## Operations on Nucleotide Sequences - Continued
### Sequence Complementation and Reverse Complementation

- *seqdisp* function makes it visually easier to view sequence
- Complemented (line 6) and Reverse complemented (line 10)

```
%Display the original sequence
>> seqdisp (ntSeq)
ans =
 1   ACAGTGCCCC CCTATATGGC CACCAGGTAG

%Display the complement of sequence
>> seqdisp (seqcomplement (ntSeq))
ans =
 1   TGTCACGGGG GGATATACCG GTGGTCCATC
%Display the reverse complement of sequence
>> seqdisp (seqrcomplement (ntSeq))
ans =
 1   CTACCTGGTG GCCATATAGG GGGGCACTGT
```

## Operations on Nucleotide Sequences - Continued
### Translation to Amino Acid Sequence

```
%Transform to a amino-acid (protein) sequence
>> aaSeq = nt2aa (ntSeq)
aaSeq =
TVPPYMATR*
%Obtain amino acid counts
>> aacount (aaSeq)
ans =
          A: 1
          R: 1
          N: 0
          D: 0
          C: 0
          Q: 0
          ...
          ...
          V: 1
     Others: 1
```

## High Level Sequence Analysis

### Calculating and Plotting Sequence Properties

- High level functions anable computation of GC content
- And, location of CpG island – often focus of interest in epigenomics

```
% Download the sequence from GenBank
>> seqObj = getgenbank('M10051');

% Plot the nucleotide density profile
>> ntdensity(seqObj.Sequence)

% Plot the CpG profile
>> cpgisland(seqObj.Sequence, 'PLOT', true)
```

# High Level Sequence Analysis - Results

## Sequence Analysis Plots - G+C Regions

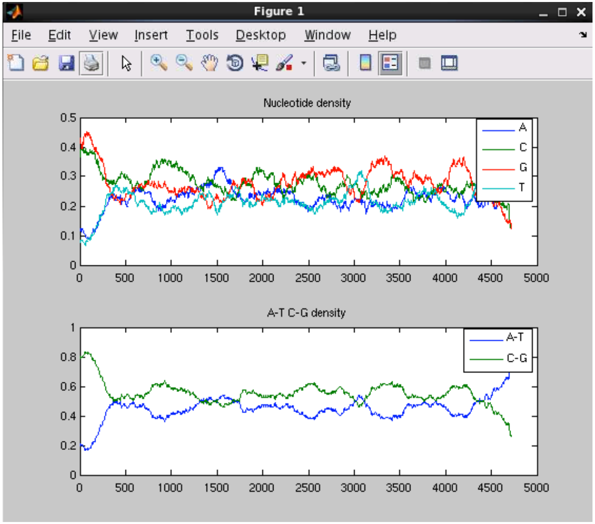

Figure: The display of nucleotide density computed by sliding a window across the

# High Level Sequence Analysis - Results

## Sequence Analysis Plots - CpG Islands

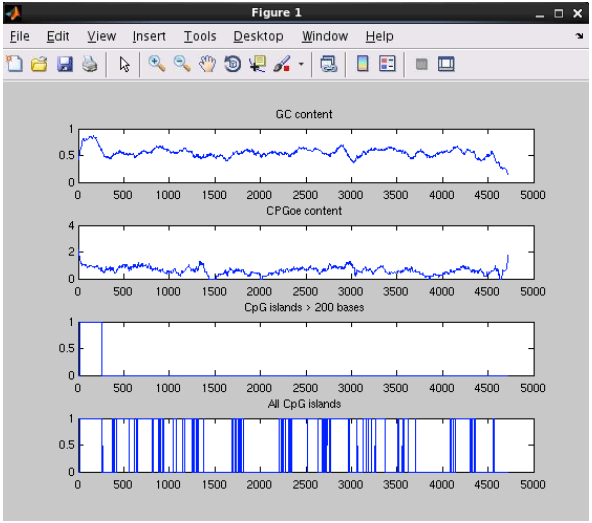

Figure: The result of applying CpG island analysis to sequence `M10051`.

## Download Sequence

- Assume we are working with a synthesized gene `Bmpr2`
- The accession number of this gene is `EU919427`
- The command to download this sequence and save it as a variable `seqObj` is

## Download Sequence

- Assume we are working with a synthesized gene `Bmpr2`
- The accession number of this gene is `EU919427`
- The command to download this sequence and save it as a variable `seqObj` is

-
  ```
      seqObj = getgenbank ('EU919427');
  ```
- And to write it into a FASTA file, `gmpr2.fa` is:

## Download Sequence

- Assume we are working with a synthesized gene Bmpr2
- The accession number of this gene is EU919427
- The command to download this sequence and save it as a variable seqObj is

-
  ```
  seqObj = getgenbank ('EU919427');
  ```
- And to write it into a FASTA file, gmpr2.fa is:

```
fastawrite('bmpr2.fa', seqObj);
```

## Obtain Sequence Data, Feature Data, Two Strands

- The Sequence Object `seqObj` is a structure with many members, including the actual sequence and its features
- Command to retrieve the actual (nucleotide, or DNA) sequence information from this structure into a character variable `seq`:

## Obtain Sequence Data, Feature Data, Two Strands

- The Sequence Object seqObj is a structure with many members, including the actual sequence and its features
- Command to retrieve the actual (nucleotide, or DNA) sequence information from this structure into a character variable seq:

-
  ```
      seq = seqObj.Sequence;
  ```
- Command to retrieve features into character variable feat:

## Obtain Sequence Data, Feature Data, Two Strands

- The Sequence Object seqObj is a structure with many members, including the actual sequence and its features
- Command to retrieve the actual (nucleotide, or DNA) sequence information from this structure into a character variable seq:
-

```
    seq = seqObj.Sequence;
```

- Command to retrieve features into character variable feat:

```
    feat = seqObj.Features;
```

- Command to parse the character variable feat into a feature structure fs:

## Obtain Sequence Data, Feature Data, Two Strands

- The Sequence Object `seqObj` is a structure with many members, including the actual sequence and its features
- Command to retrieve the actual (nucleotide, or DNA) sequence information from this structure into a character variable `seq`:
-

```
        seq = seqObj.Sequence;
```

- Command to retrieve features into character variable `feat`:

```
        feat = seqObj.Features;
```

- Command to parse the character variable `feat` into a feature structure `fs`:

```
    fs = featureparse(feat);
```

## Six Frames

- Commands to create a reverse complement strand `rev` for `seq`

## Six Frames

- Commands to create a reverse complement strand `rev` for `seq`

-
  ```
        rev = seqrcomplement (seq);
  ```
- Command to create the three forward strands:

## Six Frames

- Commands to create a reverse complement strand `rev` for `seq`
- 
```
rev = seqrcomplement (seq);
```

- Command to create the three forward strands:

```
fwd1 = seq;
fwd2 = seq(2:length(seq));
fwd3 = seq(3:length(seq));
```

- Command to create the three reverse strands::

## Six Frames

- Commands to create a reverse complement strand `rev` for `seq`

-
  ```
        rev = seqrcomplement (seq);
  ```
- Command to create the three forward strands:

```
        fwd1 = seq;
        fwd2 = seq(2:length(seq));
        fwd3 = seq(3:length(seq));
```

- Command to create the three reverse strands::

```
        rev1 = rev;
        rev2 = rev(2:length(rev));
        rev3 = rev(3:length(rev));
```

## Six Frame Translations

- Commands to generate protein sequence translations for the three forward frames:

## Six Frame Translations

- Commands to generate protein sequence translations for the three forward frames:

```
        aa1 = nt2aa(fwd1);
        aa2 = nt2aa(fwd2);
        aa3 = nt2aa(fwd3);
```

- Commands to generate protein sequence translations for the three reverse frames:

## Six Frame Translations

- Commands to generate protein sequence translations for the three forward frames:

```
aa1 = nt2aa(fwd1);
aa2 = nt2aa(fwd2);
aa3 = nt2aa(fwd3);
```

- Commands to generate protein sequence translations for the three reverse frames:

```
aa4 = nt2aa(rev1);
aa5 = nt2aa(rev2);
aa6 = nt2aa(rev3);
```

## Stop Codon Locations

- Commands to find locations of STOP codons in all six translations:

## Stop Codon Locations

- Commands to find locations of STOP codons in all six translations:

```
    stp1 = find (aa1 == '*');
    stp2 = find (aa2 == '*');
    stp3 = find (aa3 == '*');
    stp4 = find (aa4 == '*');
    stp5 = find (aa5 == '*');
    stp6 = find (aa6 == '*');
```

## Longest Open Reading Frame (ORF)

- Commands to find the longest open reading frame in each strand:

## Longest Open Reading Frame (ORF)

- Commands to find the longest open reading frame in each strand:

```
      orf1 = max (diff(stp1));
      orf2 = max (diff(stp2));
      orf3 = max (diff(stp3));
      orf4 = max (diff(stp4));
      orf5 = max (diff(stp5));
      orf6 = max (diff(stp6));
```

- **Note:** The function *diff* finds the differential of an *n-dimensional* matrix **X** to produce an *(n-1) dimensional* vector **Y**, where the element $\mathbf{Y}(\mathbf{i}) := \mathbf{X}(\mathbf{i}+\mathbf{1}) - \mathbf{X}(\mathbf{i})$.
- The **max** operator finds the longest difference.

## Closing the Loop

Review the annotations on the sequence downloaded from GenBank and compare your results.

- What is the length of the CDS annotated on the sequence?
- What location does the CDS start?
- What frame is coding?
- How do these annotations compare with your computational findings?
- When working with an anonymous DNA segment, biologists use the six frame translations to determine putative gene products of that DNA. Of course, its all been automated now...

## Operations on Nucleotide Sequences

- seqObject.Feature
- seqdisp(seqObj.Sequence) VS seqObj.Sequence
- length(xxx)
- ntSeq = 'ACAGTGCCCCCTATATGGCCACCAGGTAG'

## some MATLAB Functions and their usage

| Function & Usage |
|---|
| basecount (ntSeq) |
| seqdisp (ntSeq) |
| seqdisp (seqcomplement (ntSeq)) |
| seqdisp (seqrcomplement (ntSeq)) |
| nt2aa (ntSeq) |
| aacount (aaSeq) |
| seqObj = getgenbank('M10051'); |
| ntdensity(seqObj.Sequence) |
| cpgisland(seqObj.Sequence, 'PLOT', true) |

## Joining Exons

```
join ={{'AF018429',[282:561]}, {'AF018429',[1034:1172]} ...
      {'AF018430',[560:651]}, ...
     {'AF018431',[1:45]}, ...
     {'AF018432',[658:732]}, {'AF018432',[884:954]},
   {'AF018432',[1391:1447]}}
exons = num2cell(zeros(1,length(join)))
for i=1:length(orfs)
    seqObj = getgenbank(joini1);
    orfs(i) =  seqObj.Sequence(joini2)
end
CDS = cell2mat(orfs)
```

See section 3.4 for an example.

## Restriction Site Detection

**Restriction Enzymes:** an enzyme with the property of cutting DNA molecules at or near a specific sequence of bases.

1. Eco-RI $\underrightarrow{palindrome}$ $GAATTC \rightsquigarrow G|A$
2. Hind-III $\underrightarrow{palindrome}$ $AAGCTT \rightsquigarrow A|A$

Restriction maps are computed to plan out biological experiments

## Restriction Enzymes and Motifs

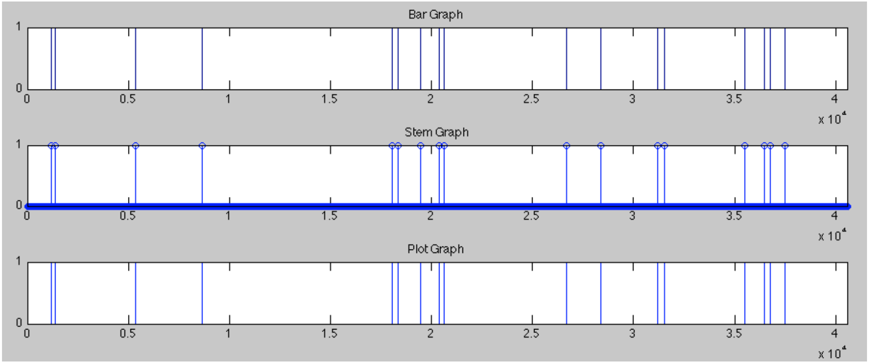

Figure: The motifs recognized by restriction enzymes (a) EcoRI and (b) HindIII. Both of these enzymes, like most restriction enzymes, cut a DNA sequence at a biological palindromic site.

## Example on Restriction Enzymes and Motifs

- fastawrite('U15422-Subseq.fasta', 'Subsequence U15422 (1:1000)-Fasta format', ... seqEmbl.Sequence(1:1000))
- fastawrite('U15422.fasta', 'U15422 - Fasta format', seqEmbl.Sequence)
- seq = fastaread('U15422.fasta')
- cutPattern = '(GAATTC|AAGCTT)';
  cutLocations = regexpi (seq.Sequence, cutPattern);
  cutLocations = cutLocations + 1;

## Restriction Enzyme example continues

-
  ```
  xvals = 1:length(seq.Sequence);
  yvals = zeros(1, length(xvals));
  for i = 1:length(cutLocations)
     yvals(cutLocations(i)) = 1;
  end;
  ```
- `stem (xvals, yvals), ... title('Stem Graph'), set(gca, 'YTick', 0:1)`