Sequence bioinformatics
http://bio.lundberg.gu.se/courses/ht07/bio2/

Perl programming (GK)
Hidden Markov Models (MO)
Methods and applications
- Algorithms of sequence alignment, BLAST, multiple alignments (GK)
- Patterns (TS) Home assignment
- Profiles (TS) Practical
- RNA bioinformatics (TS) Practical
- Molecular phylogeny (TS) Home assignment
- Databases (TS,MDL) Practical
- Genome bioinformatics (TS) Practical

+ group projects

Patterns / Regular expressions

Methods / Algorithms in biological sequence analysis

Pairwise and multiple alignments
Dynamic programming algorithm

Applications in:
- Sequence assembly
- Classification
- Prediction of function
- Comparative genomics
- Phylogeny / Evolutionary history

Pattern matching

Regular expression
"Regexp"
"Regex"
Sequence alignment using dynamic programming matrix

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>A</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>-5</td>
<td>-10</td>
</tr>
<tr>
<td>G</td>
<td>-5</td>
<td>2</td>
<td>-3</td>
</tr>
<tr>
<td>C</td>
<td>-10</td>
<td>-3</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>-15</td>
<td>-8</td>
<td></td>
</tr>
</tbody>
</table>

Patterns / Regular expression matching

GRTKLPKLMKKWREKNRLYKMKWRRAGGALKALK

Is there a match to \textbf{KWR} in this string?

Notes on the theory of regular expressions

* A \textit{regular expression} is an expression in a \textit{regular grammar}.

(Regular grammars cannot handle long range interactions like below - a \textit{context free grammar} is required for that)

\textbf{GAGG G T T C G A T C C C T C}

|   |   |   |   |   |   |   |   |

* Regular expressions are also related to theory of \textit{automata}:

  Regular expressions can be translated into:
  Non-deterministic Finite Automaton (NFA) &
  Deterministic Finite Automaton (DFA)

  (and this is what happens in Perl etc.)
Regular expressions and finite automata

Regexp :  a(bb)+a

<table>
<thead>
<tr>
<th>Start state</th>
<th>Match state</th>
</tr>
</thead>
<tbody>
<tr>
<td>S₀ → a → S₁ → b → S₂ → b → S₃ → a → S₄</td>
<td></td>
</tr>
</tbody>
</table>

Testing the string "abbbba"

<table>
<thead>
<tr>
<th>Step</th>
<th>States</th>
<th>Input</th>
<th>State Move</th>
</tr>
</thead>
<tbody>
<tr>
<td>STEP 0</td>
<td>S₃</td>
<td>b</td>
<td>S₂</td>
</tr>
<tr>
<td>STEP 1</td>
<td>S₃</td>
<td>b</td>
<td>S₂</td>
</tr>
<tr>
<td>STEP 2</td>
<td>S₃</td>
<td>b</td>
<td>S₂</td>
</tr>
</tbody>
</table>

Final match state: S₄

Regular expression: a(bb)+a
This machine (automaton) is deterministic because in any state each possible input character leads to at most one new state.

This machine is equivalent to the previous one, but is non-deterministic because in the state S2 it has multiple choices for b:
Patterns / Regular expressions

When are patterns useful in bioinformatics?

* Finding patterns in DNA and protein sequences
  - Restriction enzyme sites (DNA)
  - Transcription factor binding sites (DNA)
  - PROSITE, using patterns to infer biological function from protein sequences
* Finding patterns to extract information from text files of bioinformatics applications, such as GenBank reports and BLAST output.

One major advantage with patterns:
Search is extremely fast.

Recognition sites of restriction enzymes

EcoRI  -GAATTC-  -CTTAAG-

BamHI  -GGATCC-  -CCTAGG-

XhoII  -RGATCY-  \( R = A \text{ or } G \)
     -YCTAGR-  \( Y = C \text{ or } T \)

PpuMI  -RGGWCCY-  \( W = A \text{ or } T \)
     -YCCWGR-
Restriction enzyme sites

http://rna.lundberg.gu.se/cutter2/

Web-cutter output

tcaaggctctctctggaagctgactctacgattcaggctgactcacagtgttggtctcatgacaaattaatttatatttcacatgagg

agttccgaggagagaactaagtccgactgagtgtcacaaccagagtactgtttaattaaatataaagtgtactcc

RcaI      VspI
BsaI      AseI

Eco57I                             NspI
gacaggatgattttcttgaaaaaacatttcgaagaactcttgactctctctctctctgcctggcaagacatgca...
ctgtcctactaaagaacttttttggaagcttttatgtagtagtagacggataaagagtcgggtcgtctgtacg76 to 150

MslI
Transcription factor databases contain information about DNA sequences recognized by transcription factors.

### Regulation of transcription

![Diagram showing regulation of transcription]

### Transcription factor databases

<table>
<thead>
<tr>
<th>Name</th>
<th>Sequence</th>
<th>Trans. Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>UAS(G)-pMH100</td>
<td>0 CGGAGTACTGTCCTCCG</td>
<td>0 GAL4</td>
</tr>
<tr>
<td>TFIIC-46s-58.1</td>
<td>0 TGTAGGGAG</td>
<td>0 TFIIC</td>
</tr>
<tr>
<td>GSE_CS_inverted_repeat</td>
<td>0 CTGAAAGCTTTCAAG</td>
<td>0 HSTF</td>
</tr>
<tr>
<td>ZDNA_CS</td>
<td>0 GCTGGTGCA</td>
<td>0 unknown</td>
</tr>
<tr>
<td>GCN4-61s-189</td>
<td>0 ATGAGCTCTA</td>
<td>0 GCN4</td>
</tr>
<tr>
<td>AP1_EIA_element_I</td>
<td>0 AGGAGCTGAAA</td>
<td>0 unknown</td>
</tr>
<tr>
<td>AP1_EIA_element_II</td>
<td>0 GGGCGTAACCCGAGACATTGGGTCATTTTTC</td>
<td>0 unknown</td>
</tr>
<tr>
<td>BPV-E2_CS1</td>
<td>0 ACCNNNNCGGT</td>
<td>0 BPV-E2</td>
</tr>
<tr>
<td>Alb_DEII</td>
<td>0 GATTTTTATGTTG</td>
<td>0 C/EBP</td>
</tr>
<tr>
<td>Alb_DEIII</td>
<td>0 TTATTTGCAAAAAG</td>
<td>0 CTF/NF-1</td>
</tr>
<tr>
<td>Alb_PKI</td>
<td>0 GCAAAGAGCTTTAGT</td>
<td>0 unknown</td>
</tr>
<tr>
<td>BPV-E2_CS2</td>
<td>0 ACCNNNNCGGT</td>
<td>0 BPV-E2</td>
</tr>
<tr>
<td>CACA</td>
<td>0 CACACACACACA</td>
<td>0 unknown</td>
</tr>
<tr>
<td>dc_box</td>
<td>0 TTTATTTCCAT</td>
<td>0 unknown</td>
</tr>
<tr>
<td>GALU_E</td>
<td>0 AGAATAGTGTGCAACAG</td>
<td>0 unknown</td>
</tr>
<tr>
<td>GCNE</td>
<td>0 TGACCTC</td>
<td>0 GCN4</td>
</tr>
<tr>
<td>Pit-1 CS1</td>
<td>0 MATATTCAT</td>
<td>0 Pit-1</td>
</tr>
</tbody>
</table>

Transcription factor binding sites - example:

Estrogen response element

\[5'\text{AGGTCANNTGACCT}\]
.. but many useful transcription factor databases of today also make use of weight matrices

<table>
<thead>
<tr>
<th></th>
<th>10</th>
<th>0</th>
<th>0</th>
<th>10</th>
<th>5</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>T</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

(Possible pattern: A T T A N N)

Protein patterns and the PROSITE database
Proteins that bind the nucleotides ATP or GTP share a short sequence motif

Entry in PROSITE for the ATP/GTP binding site motif

ID   ATP_GTP_A; PATTERN.
AC   PS00017;
DT   APR-1990 (CREATED); APR-1990 (DATA UPDATE); NOV-1990 (INFO UPDATE).
DE   ATP/GTP-binding site motif A (P-loop).
PA   [AG]-x(4)-G-K-[ST].
CC   /TAXO-RANGE=ABEPV;
3D   1EFM; 1ETU; 1Q21; 2Q21; 4Q21; 5Q21; 6Q21;
DO   PDOC00017;

[AG]-x(4)-G-K-[ST]
Another example of protein family represented in PROSITE: a category of zinc finger proteins

PROSITE entry of C2H2 type of zinc finger protein

ID   ZINC_FINGER_C2H2; PATTERN.
AC   PS00028;
DT   APR-1990 (CREATED); JUN-1994 (DATA UPDATE); NOV-1997 (INFO UPDATE).
DE   Zinc finger, C2H2 type, domain.
PA   C-x(2,4)-C-x(3)-[LIVMFYWC]-x(8)-H-x(3,5)-H.
NR   /RELEASE=35,6913;
A PROSITE pattern typically corresponds to a protein structural element with a specific function

* Catalysis
  example: trypsin family (proteins responsible for digestion of protein)

* Regions involved in binding to other molecules
  example: DNA binding domain of a transcription factor

* Prosthetic group attachment sites

* Amino acids involved in binding to a metal ion.

* Cysteines forming disulfide bonds
Disulfide bonds in EGF domain

- First identified in epidermal growth factor, but present in many other proteins
- Function unclear, but found in extracellular part of membrane proteins
- Includes six cysteins involved in disulfide bonds

\[
\begin{align*}
C-x(0,48)-C-x(3,12)-C-x(1,70)-C-x(1,6)-C-x2-G-x(0,21)-G-x2-C
\end{align*}
\]

Syntax of PROSITE patterns

- **D** matches D
- **[AS]** matches A or S
- **x** any symbol (amino acid)
- **x(3) or x3** any three symbols (amino acids)
- **{AT}** matches any symbol except A and T
- **[AS]2** Matches two positions with either A or S (i.e. AA, AS, SS, or SA)
- **x(3,7)** a sequence of symbols between 3 and 7 in length (for instance GAT, GCRE, PPLKM, GTTREC or PPPPPP)
Flexibility of patterns

The expression
x (i,j) is said to have flexibility j-i+1

The total flexibility of a PROSITE pattern may be defined as the product of the individual flexibilities.

Consider for instance the pattern:
A - x(4,7) - G -x(5,6) - D

Total flexibility = (7-4+1) x (6-5+1) = 8

High flexibility increases search time
PROSITE

Database of protein families and domains

http://www.expasy.org/prosite/ (Swiss Institute of Bioinformatics)

Release 19.29, of 19-Sep-2006 (contains 1446 documentation entries that describe 1331 patterns and 650 profiles/matrices).

A - x(4,7) - G -x (5,6) - D

"Non-probabilistic"

K  0 0   0 0  0  0
G  0 0   0 0  1  4
L  0 10  10 0  4  2

"Probabilistic"

Method to generate PROSITE patterns

1. Study literature on the protein family.
2. Study one or more alignments of sequences belonging to the family and look for motifs corresponding to, for example:
   - Catalytically active region
   - Cysteines forming disulphide bonds
   - Regions involved in binding to other molecules (including other proteins)
3. Identify a conserved sequence of 4-5 residues within the chosen motif.
4. Search SWISS-PROT for sequences matching this initial pattern.
5. Extend the pattern until its classification accuracy is "adequate".
Certain patterns have a high probability of occurrence - are more likely to generate false positives

*Classification accuracy* varies between families/patterns

* True positive - member of the protein family that is matched by the pattern
* False positive - a protein which is *not* a member of the protein family is matched by the pattern
* True negative - a protein which is *not* a member of the protein family is *not* matched by the pattern
* False negative - member of the protein family is *not* matched by pattern
We want to minimize FN + FP.

Classification accuracy

\[
\text{Specificity} : \frac{\text{TP}}{\text{TP} + \text{FP}}
\]

\[
\text{Sensitivity} : \frac{\text{TP}}{\text{TP} + \text{FN}}
\]

- Ex: The flavodoxin family (PS00201)
  - True positives: 51
  - False positives: 4
  - False negatives: 9

\[
\text{Specificity} = \frac{51}{(51+4)} \Rightarrow 0.93
\]

\[
\text{Sensitivity} = \frac{51}{(51+9)} \Rightarrow 0.85
\]
Classification accuracy varies

<table>
<thead>
<tr>
<th>Classification</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zinc finger C2H2 type domain signature (PATTERN)</td>
<td>PS00028 ZINC_FINGER_C2H2_1</td>
<td>C - x(2,4) - C - x(3) - [LIVMFYW] - x(8) - H - x(5,6) - H. The 2 C's and the 2 H's are zinc ligands.</td>
</tr>
<tr>
<td>ATP/GTP-binding site motif A (P-loop) (PATTERN with a high probability of occurrence)</td>
<td>ATP_GTP_A, PS00001</td>
<td>[AG] - x(4) - G - K - [ST]. A majority.</td>
</tr>
<tr>
<td>N-glycosylation site (PATTERN with a high probability of occurrence)</td>
<td>ASN_GLYCOSYLATED, PS00001</td>
<td>N - (P) - [ST] - (P). N is the glycosylation site.</td>
</tr>
</tbody>
</table>

Biological information may be extracted from a protein sequence using PROSITE, for example:

A ‘new’ protein sequence has been identified using bioinformatics methods (like in a genome project). A scan of PROSITE using this sequence (regular expression matching) can give important clues as to the biological function of the protein.
Common applications of PROSITE (ScanProsite/Motifscan)

- Search a sequence for all PROSITE patterns/profiles
- Search database with a pattern/profile

Regular expressions are used in ...

- Patterns as used in PROSITE
- EMBOSS program fuzzpro
- Operating system tools, for instance grep
- Text editors
- Programming and scripting languages, for instance perl
Sample session with fuzzpro

% fuzzpro
Protein pattern search
Input sequence(s): tsw:*
Search pattern: [FY][LIV]-G-[DE]-E-A-Q-x-[RKQ](2)-G
Number of mismatches [0]:
Output report [100k_rat.fuzzpro]:

The UNIX utility grep / egrep

grep "[AG]{4}GK[TS]" /dbs/swissprot

will print all lines containing matching the pattern corresponding to the ATP/GTP binding site motif in PROSITE

/dbs/swissprot is the Swissprot protein sequence database in FASTA format
Perl

Example code that uses a regular expression:

```perl
variable
$str = 'GACACAGGGATCGGGGATC';

binding operator
if ( $str =~ /[AG][CT]CG/ ) { 
    regexp delimiter
    print "Is there a match of [AG][CT]CG in the variable $str?"
    print "Found match!
    }
}

regexp will match here
GACACAGGGATCGGGGATC
```

The substitution and transliteration operators

Transcribing DNA into RNA.

```
$substitution operator
$dna = 'GCAATGG';
print "The DNA sequence is $dna\n";
$ma = $dna;
$ma =~ s/T/U/g;
print "and the RNA sequence is $ma\n";
```

```
transliteration operator
s/T/U/g;
```

```
PATTERN; regexp to be replaced by REPLACEMENT
```
The substitution and transliteration operators

Count the bases in a DNA sequence using `tr`

```perl
$dna = 'GCAATGNGATTACTTCG';
$basecount = ($dna =~ tr/ACGT//;)
$nonbase = length($dna)-$basecount;
print "There are $basecount As,Cs,Ts,and Gs \n";
print "There is/are $nonbase other symbol(s) \n";
```

`tr/ACGT//;`
operation does not change the string!

Metacharacters

.  match any character
^  match beginning of string
$  match end of string
?  optional match

Quantifiers

*  Any number of characters, including zero
+  One or more characters
{m,n} minimum m , maximum n characters
Reading a file in PERL and checking for regular expressions

Program will read the file with the name 'myfile' and that has a sequence in EMBL format. The program will print to the screen all lines starting with 'FT', i.e lines with feature table information.

```perl
open IN, 'myfile';
while (<IN>) {
  if (/^FT/) {print ;}
}
close IN;
```

Example of how regexps may be used to parse the output from a BLAST search

'bl2seq' is a program to blast two sequences against each other. This is an output using two sequences seq1 and seq2:

```
Query= seq1
   (31 letters)
>seq2  Length - 29
Score = 42.1 bits (21), Expect = 1e-10
Identities = 27/29 (93%)
Strand = Plus / Plus

Query: 3 acgacgtacagactagtcaggcggagct 31
       ||||||||| |||||||||||| ||||||||
Sbjct: 1 acgacgttcacgactagtcacgcggagct 29
       .... (and some more text ) ......
```
Example of how regexps may be used to parse the output from a BLAST search

Let's say you have many output files like this and you want to make a script to present the names of the two sequences as well as the Expect value listed. Here's an example of code: (we assume that the blast output is in a file called 'bl2seq.out')

```perl
open IN, 'bl2seq.out';
while (<IN>) {  # we read one line at a time, each
    # line is stored in the default variable $
    chomp;  # remove the end of line character in $
    if (/[Query= (.*)]/) {print "$1 ";}  # greedy
    if (/[^>(.*)]/) {print "$1 ";}  # greedy
    if (/[Expect = (.*)]/) {print "$1\n";}  # greedy
}
close IN;
```

* means any number (including zero) of any characters; the regexp algorithm is **greedy** so it will try to find the longest substring that matches.

---

**Regular expression match basics**

1. Quantifiers `* + ? {m,n}` are greedy

   Perl example:

   ```perl
   $str = "GGAAGG";
   $str =~ s/GG/XX/;
   ``

   $str is now XXAAGG

2. The match that begins earliest (leftmost) wins

   Perl example:

   ```perl
   $str = "GGAAGG";
   $str =~ s/GG/XX/;  # greedy
   print $str;
   ``

   $str is now XXAAGG