Week 3, Lecture 5

István Albert
Bioinformatics Consulting Center
Penn State

Finding matching strings

- Exact matches → it is straightforward task →
  grep ATGC
- Approximate matches → so called regular expressions →
  egrep AT.GC
  The dot above means "any character", reading the expression it means:
  find patterns where there is an "AT followed by any character followed by GC"
  it will find the following
  ATGGC, ATAGC, ATTGC, ATCGC but will not find ATGC

Regular Expressions (RE)

- Are a world unto their own → can be simple or extremely complex → you can think of them as a programming language
- But even simple patterns can get you pretty far.
- Warning: REs tricky, it is not easy to see when you match too much: example
  egrep “AAA..AAA”

Find two base long sequences bounded by triple As

How to solve it: discussion on homework

- Default sort can be deceiving – it looks right and happens to be right this time
- Look up documentation on methods. Understand what parameters to the the `awk substr` actually mean.
- Use simple inputs to test behaviors pass: 1, 2, ABC, Hello World,
Analyze the matches

To build them use a regular expression evaluator

Regular expression patterns

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[abc]</td>
<td>A single character of a, b, or c</td>
</tr>
<tr>
<td>[^abc]</td>
<td>Any single character except a, b, or c</td>
</tr>
<tr>
<td>[a-z]</td>
<td>Any single character in the range a - z</td>
</tr>
<tr>
<td>[a-zA-Z]</td>
<td>Any single character in the range a - Z</td>
</tr>
<tr>
<td>^</td>
<td>Start of line</td>
</tr>
<tr>
<td>$</td>
<td>End of line</td>
</tr>
<tr>
<td>\A</td>
<td>Start of string</td>
</tr>
<tr>
<td>\Z</td>
<td>End of string</td>
</tr>
<tr>
<td>.</td>
<td>Any single character</td>
</tr>
<tr>
<td>\s</td>
<td>Any whitespace character</td>
</tr>
<tr>
<td>\S</td>
<td>Any non-whitespace character</td>
</tr>
<tr>
<td>\d</td>
<td>Any digit</td>
</tr>
<tr>
<td>\D</td>
<td>Any non-digit</td>
</tr>
<tr>
<td>\w</td>
<td>Any word character (letter, number, underscore)</td>
</tr>
<tr>
<td>\W</td>
<td>Any non-word character</td>
</tr>
<tr>
<td>\b</td>
<td>Any word boundary</td>
</tr>
</tbody>
</table>

(C...) Capture everything enclosed
(a+) a or b
(a?) Zero or one of a
(a*) Zero or more of a
(a+) One or more of a
(a{3}) Exactly 3 of a
(a{3,}) 3 or more of a
(a{3,6}) Between 3 and 6 of a

Having fun with regular expressions

- For anything more complicated you need to protect separate parts of the sequence with ()

- an A followed by three Ts → AT[3]

- three ATs in a row → (AT){3}

- one or more ATG followed by C → (ATG)+C{2}

Keep building and testing them → very empowering
Edit distances

• The number of changes to transform from one string to another. Edit distance 1

HAT → BAT → BIT → LIT

• A simple way to describe the similarity between strings → overly simple

Alignment algorithms

• Can be **optimal** (mathematically precise) or **heuristic** (occasional false negatives)

• Optimal alignments are usually computationally very demanding requiring a matrix of length of the sequences (N x M)

• Optimal global alignment: Needleman-Wunsch, 1970
• Optimal local alignment: Smith-Waterman, 1981

Alignments

This is a topic we will regularly revisit on various topics → expanding the concepts.

**Global Alignments:** every residue will be aligned

**Local Alignments:** similar regions will be aligned

**Semi-Global Alignments:** no penalty for gaps at the end → applicable when looking to fully aligning short sequences against very long ones.
Scoring Alignments

Values are associated to:

- an exact match \(\rightarrow\) usually a positive score
- a mismatch \(\rightarrow\) usually a negative score but that depends on the kind of mismatch
- gap opening \(\rightarrow\) usually the most penalized action
- gap extensions \(\rightarrow\) making the gap longer (smaller than gap opening)

The values above are the so called **scoring matrix**

Important to remember

1. **Any two sequences can be aligned**, the alignment score represents the match/mismatch/insertion/deletions
2. Most aligners will **only report alignments that make some sense**
3. **Alignment is a measure of similarity but not homology**

Align some sequences

First sequence:

>1 THISLINE

Second sequence:

>2 ISALIGNED

Example from the book Understanding Bioinformatics (see website)

Global Alignment

Use the `needle` program.

Usually Gaps: - or spaces

Exact matches: |

Other format specific qualifiers: ...

Check the documentation for exact details
Local Alignments

<table>
<thead>
<tr>
<th>#</th>
<th>1</th>
<th>7 NE 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>7 NE 8</td>
<td></td>
</tr>
</tbody>
</table>

Scoring matrices

```
# Matrix made by matblas from blocks
# * column uses minimum score
# Blocks Database = /data/blocks_5
# Cluster Percentage: >= 62
# Entropy = 0.6979, Expected =
#           A   R   N   D   C   Q   E   G   H   I   L
#          -1 -2 -2  0  -1 -1  -2 -1 -1 -1 -1
#          R -5  0 -2 -3  1  -2  0 -3 -2
#          N -2  0  1 -3  0  0  0  1 -3 -3 -3
#          D -2 -2  1  6 -3  0  2 -1 -3 -4 -4
#          C  0  3  3 -3  9  -3 -4 -3 -1 -1 -1
```

Homework 5 – part 1

- How many of your sequences contain a “TATA box” defined as:

  1. a string that starts with TA followed by any number of A or T ending with TA.

  2. a string that starts with TA followed by at least five A or T ending with TA

Homework 5 – part 2

What parameters could lead to a local alignment that looks like this

Use the water tool on EMBL to produce this alignment