Week 3, Lecture 5

István Albert

Bioinformatics Consulting Center

Penn State
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,
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
Analyze the matches

To build them use a regular expression evaluator

```bash
bialbert@porthos ~:/web/pyblue-sites/ialbert.web.web/courses/files/2013/code
cat lec4.fa | egrep "AAA..AAA" | head -1
ACAGTATATAGGAGGCAGCAGTAAGGAAATCTTTTCGTCAATGGGCGCAAGCCTGAAC
GAGCGATGCGCGCGTGGTGAGGAGGAGTTTTCGCTGTAAGCCCTCGTTGTGTTG
AGTGAAAAGGAAAAGACCGTGTCAGGAAAATAGGTCGTGCTTTTTTGATGGAAAGGTCACC
GAGGAAATGCAACGGCTAACTACGTGCCA
```
There are many regular expression testers → slight differences between languages this is only a problem for more advanced patterns
Regular expression patterns

- `[abc]` A single character of: a, b, or c
- `[^abc]` Any single character except: a, b, or c
- `[a-z]` Any single character in the range a-z
- `[a-zA-Z]` Any single character in the range a-z or A-Z
- `^` Start of line
- `$` End of line
- `\A` Start of string
- `\z` End of string
- `( . . . )` Capture everything enclosed
- `(a|b)` 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

- Any single character
- `\s` Any whitespace character
- `\S` Any non-whitespace character
- `\d` Any digit
- `\D` Any non-digit
- `\w` Any word character (letter, number, underscore)
- `\W` Any non-word character
- `\b` Any word boundary
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 $\rightarrow$ AT{3}

- three ATs in a row $\rightarrow$ (AT){3}

- one ore more ATG followed by C $\rightarrow$ (ATG)+C{2}

Keep building and testing them $\rightarrow$ very empowering
Edit distances

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

HAT → BAT → BIT → LIT

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

- **Input Sequences**
  - Sequence A
  - Sequence B

- **Global Alignment**
  - Sequence A
  - Sequence B
  - GAPS
  - Sequence B
  - GAPS

- **Two Local Alignments**
  - Sequence A
  - Sequence B
  - Sequence B

- **Semi Global Alignment**
  - Sequence A
  - Sequence B
  - GAPS
  - Sequence B
  - GAPS
  - Sequence B
Scoring Alignments

Values are associated to:

- an exact match → usually a positive score
- a mismatch → usually a negative score but that depends on the kind of mismatch
- gap opening → usually the most penalized action
- gap extensions → 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
Pairwise Sequence Alignment is used to identify regions of similarity that may indicate functional, structural and/or evolutionary relationships between two biological sequences (protein or nucleic acid).

By contrast, Multiple Sequence Alignment (MSA) is the alignment of three or more biological sequences of similar length. From the output of MSA applications, homology can be inferred and the evolutionary relationship between the sequences studied.

Global Alignment

Global alignment tools create an end-to-end alignment of the sequences to be aligned. There are separate forms for protein or nucleotide sequences.

Needle (EMBOSS)

EMBOSS Needle creates an optimal global alignment of two sequences using the Needleman-Wunsch algorithm.

Genomic Alignment

Genomic alignment tools concentrate on DNA (or to DNA) alignments while accounting for characteristics present in genomic data.

Wise2DBA

Wise2DBA (DNA Block Aligner) aligns two sequences under the assumption that the sequences share a number of colinear blocks of conservation separated by potentially large and varied lengths of DNA in the two sequences.

GeneWise

GeneWise compares a protein sequence to a genomic DNA sequence, allowing for introns and frameshifting errors.

PromoterWise

PromoterWise compares two DNA sequences allowing for inversions and translocations, ideal for promoters.

EMBL software tools
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

Use the **water** program.

Usually Gaps: - or spaces
Exact matches: | 
Other format specific qualifiers: . : 

Check the documentation for exact details
**Scoring matrices**


```
# Matrix made by matblas from blast
# * column uses minimum score
# BLOSUM Clustered Scoring Matrix
# Blocks Database = /data/blocks_5
# Cluster Percentage: >= 62
# Entropy = 0.6979, Expected =

A  4  -1  -2  -2  0  -1  -1  0  -2  -1  -1
R -1   5  0  -2  -3  1  0  -2  0  -3  -2
N -2   0  6  1  -3  0  0  0  1  -3  -3
D -2  -2   1  6  -3  0  2  -1  -1  -3  -4
C  0  -3  -3  -3  9  -3  -4  -3  -3  -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