# Sequence Alignment using *Align*

## Part III:  Scoring Schemes

Richard M. Salter
Oberlin College

Supported by NSF CCLI DUE 0618252

# The *Needleman-Wunsch* Algorithm

- The technique employed by *Align* in Part II is called the *Needleman-Wunsch* global alignment algorithm. Recall its highlights:
  - A scoring system based on match, mismatch and gap penalty scores
  - A matrix in which optimal subsequence alignment results are recorded
  - A growing “green path” representing the best alignment of the subsequences terminating at a particular cell within the matrix.
  - A final “green path” covering the entire sequence pair that shows the optimal alignment of the complete pair for the given scoring system
- The algorithm is **global** because the sequences are aligned over their entire lengths.
  - We will consider some alternatives in Part IV

## Scoring

- In our examples so far we have stuck to a simple scoring scheme:

  match: +1 mismatch: 0 gap: -1

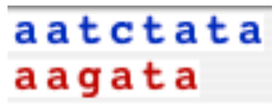
- Other schemes:
  - **Origination and Length Penalties**
  - **Scoring Matrices**

# Origination and Length Penalties

- Simple gap penalties often lead to many optimal alignments between two sequences.
- To further distinguish between these choices we differentiate between those alignments that have multiple short gaps and those that have fewer but longer gaps.
    - Gaps appear when an evolving sequence changes as a result of insertions or deletions (“indels”).
        - During alignment, each indel is represented by one or more consecutive gaps.
        - A long sequence of consecutive gaps probably represents a single long indel.
    - If two sequences are related by evolution, it is statistically more likely that any difference in length is due to a smaller number of longer indels rather than a larger number of shorter indels.
    - Consequently the alignment algorithm should favor alignments containing fewer but longer consecutive gap sequences over alignments containing many shorter, isolated gaps.
- In order to bias the scoring function to favor longer gap sequences we break the gap penalty into two parts
    - **origination penalty:** incurred for starting a new series of gaps
    - **length penalty:** based on the length of the gap sequence
- Note that if the origination penalty is large compared to the length penalty, then the algorithm will favor alignments that group gaps together rather than ones that spread them out.

# Origination and Length Penalties:  Example

- Fire up *Align* with our usual pair of sequences.

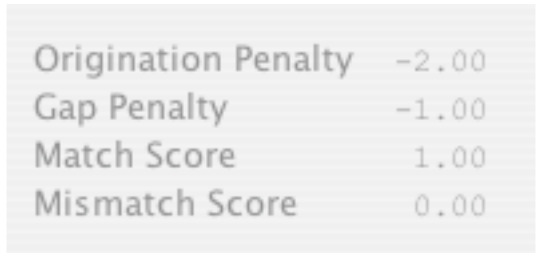


```
aatctata
aagata
```

- In **Manual** mode, using the old scoring system, verify that the three alignments displayed in the box give identical scores, as shown.

```
aatctata   3    aatctata   3    aatctata   3
aag--ata        aa-g-ata        aa--gata
```

- Now change the scoring used by *Align* so that it imposes a -2.0 origination penalty.



| | |
|---|---|
| Origination Penalty | -2.00 |
| Gap Penalty | -1.00 |
| Match Score | 1.00 |
| Mismatch Score | 0.00 |

  - To change the scoring used by the program, click **Edit | Configuration ...** and enter the origination penalty value into the *Edit Configuration* dialog window.
  - When completed, the new scoring should appear in the scoring window as shown.

- Verify in Manual mode that the three alignments now score as shown.

```
aatctata   1    aatctata   -1    aatctata   1
aag--ata        aa-g-ata         aa--gata
```

  - Be sure you can account for the change.

- Switch to **Auto** mode. Which alignment is selected as optimal?

# Exercise 3.1

- For each of the following sequence pairs:
  - Determine the optimal alignment in Auto mode using the original scoring system (i.e. without an origination penalty)*
  - Determine the optimal alignment a second time in Auto mode using an origination penalty of -2.00, as in the previous example. Did you get the same result? Explain.
  - Are you satisfied with the result?

a)
```
agactaatagctatgct
atacctggac
```

b)
```
acgtatcgcgtata
gatgctctcggaaa
```

c)
```
caattaacgtatcgcgtata
ctatcgtatgcgtatacgct
```

*Check your configuration to be certain that it is set to *Simple Scoring*, the alphabet is set to *DNA*, and that the scoring scheme is Origination Penalty: 0; Gap Penalty: -1; Match Score: 1; Mismatch Score: 0

Exercise 3.1 Solution

# Scoring Matrices

- We have refined the gap penalty to be more favorable toward evolutionarily similar sequences.
- Similarly, the match/mismatch procedure may also be refined to be more sensitive to sequences that are likely to be related.
- Our current procedure simply scores one value for a match and another for a mismatch.
- The most general approach is to use a matrix (i.e. table) that determines a score based on the letter pair under consideration.
- Below are two examples of such matrices

  a) **Identity Matrix** scores 1 for every match (`a-a, b-b, c-c, d-d`) and 0 for every mismatch. It is identical to our simple match/mismatch scoring scheme.

  b) **Transition Transversion Matrix** scores 1 for every match, but different scores for mismatches. Specifically, a mismatch between 2 purines (`a-g`) or 2 pyrimadines (`c-t`) is penalized less than a mismatch between a purine and a pyrimadine (`a-c`, `a-t`, `c-g` or `t-g`).

     This scheme recognizes the greater likelihood of an evolutionary transition within the purines and pyrimadines, than between a purine and a pyrimandine, or vice verse.

a)

| | a | t | c | g |
|---|---|---|---|---|
| a | 1 | 0 | 0 | 0 |
| t | 0 | 1 | 0 | 0 |
| c | 0 | 0 | 1 | 0 |
| g | 0 | 0 | 0 | 1 |

b)

| | a | t | c | g |
|---|---|---|---|---|
| a | 1 | -5 | -5 | -1 |
| t | -5 | 1 | -1 | -5 |
| c | -5 | -1 | 1 | -5 |
| g | -1 | -5 | -5 | 1 |

# Exercise 3.2

- Repeat Exercise 3.1 (sequences are shown below) using the transition transversion matrix. Compare results with the previous exercise.
  - To use this matrix in *Align*
    - Select **Edit | Configuration ...**
    - In the *Edit Configuration* dialog window, under *Preset Scoring*, select *DNA Trans. Transv.*
    - When the matrix appears you can change the *Origination Penalty* to -2.0 as needed.

a)
```
agactaatagctatgct
atacctggac
```

b)
```
acgtatcgcgtata
gatgctctcggaaa
```

c)
```
caattaacgtatcgcgtata
ctatcgtatgcgtatacgct
```

Exercise 3.2 Solution

# Aligning protein sequences

- Our examples so far have focused on DNA alignment.
- When aligning protein sequences you can use the same dynamic programming algorithm, only with an alphabet of at least 20 symbols representing the 20 amino acids.
- With protein sequence alignment scoring becomes more complex, and so matrix scoring is generally indicated.
- Two common approaches to deriving a scoring matrix for protein sequence alignment are
  - **Point Accepted Mutation (PAM) matrix**
  - **Block Substitution (BLOSUM) matrix**
- Both PAM and BLOSUM matrices are derived from actual substitution rates observed among similar protein sequences. Pointers to descriptions of how they are derived are given in the references.
- We will now use *Align* to look at some examples of protein sequences aligned with PAM and BLOSUM matrices.

# Protein Alphabet

- The standard 20 element amino acid list is displayed in the table on the right, along with their standard one and three letter codes.
- The next three lines specify letters usually used to designate two or more amino acids.
- Finally, some specifications use `*` to indicate the end of the sequence.
  - Sequence alignment should favor choices that terminate together.
- When a matrix is loaded into *Align*, the alphabet used by that matrix is displayed using the menu selection **Help | Alphabet Key**.

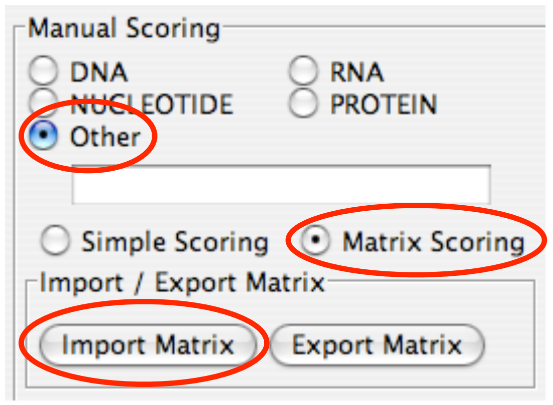
| **Amino acid** | **One letter symbol** | **Three letter symbol** |
|---|---|---|
| alanine | A | Ala |
| arginine | R | Arg |
| asparagine | N | Asn |
| aspartic acid | D | Asp |
| cysteine | C | Cys |
| glutamic acid | E | Glu |
| glutamine | Q | Gln |
| glycine | G | Gly |
| histidine | H | His |
| isoleucine | I | Ile |
| leucine | L | Leu |
| lysine | K | Lys |
| methionine | M | Met |
| phenylalanine | F | Phe |
| proline | P | Pro |
| serine | S | Ser |
| threonine | T | Thr |
| tryptophan | W | Trp |
| tyrosine | Y | Tyr |
| valine | V | Val |
| | B | Asn or Asp |
| | Z | Gln or Glu |
| | X | any |
| | * | terminal |

## Using Matrices in *Align*

- With *Align*, you can load a scoring matrix directly from text, and store that matrix, together with any gap penalties, as a *scoring configuration*.
  - Once you've created a scoring configuration, you can reload it directly.
- The *Align* distribution includes the BLOSUM 45, BLOSUM 62 and PAM 250 matrices in FASTA format.
  - FASTA format is a standard format used throughout bioinformatics for encoding sequences and data pertaining to sequence alignment.
  - *Align* uses FASTA format for both matrix and sequence files. More on this later.

# Creating an *Align* Scoring Configuration

- With *Align* it is easy to create and save scoring configurations.
- As mentioned above, *Align* is distributed with two BLOSUM matrices (BLOSUM45 and BLOSUM62) and one PAM matrix (PAM250). These are contained in files ending with ".mat" extensions.
- Let's select the PAM250 matrix for our configuration.
  - Start *Align* and select **Edit | Configuration**. Under **Manual Scoring** click **Other**.
  - Next click **Matrix Scoring**, followed by **Import Matrix**.
  - In the file chooser window open **pam250.mat**.
    - The matrix will appear in the score configuration panel.
  - Now complete the configuration by adding a gap penalty of -1.0.  Click **OK**.
    - Click **OK** when the Warning appears. Your current sequences, which use the DNA alphabet, will be deleted.


Manual Scoring
DNA  RNA  NUCLEOTIDE  PROTEIN  [✓ Other]
Simple Scoring  [✓ Matrix Scoring]
Import / Export Matrix

- To save the configuration, choose **File | Save Configuration ...** and save the configuration as **pam250-1**.
- To verify that the configuration has been saved, exit and restart *Align*, select **File | Load Configuration ...**, and choose **pam250-1.conf**. The PAM250 matrix with gap penalty of -1.0 should appear in the configuration panel on the lower right.
  - Once again, click **OK** to the Warning.

# Exercise 3.3

- Create a second configuration for the PAM250 matrix with a -10.0 origination penalty and a -1.0 gap penalty. Call it **pam250-5.conf**.
- Create similar configurations for the BLOSUM45 and BLOSUM62 matrices.
- Our origination and gap penalty choices were for demonstration purposes only.
  - Using the internet or other resources, try to find the suggested gap penalties to be used with these matrices.
  - Create alternate configurations using proper gap and/or origination penalties, as indicated by the designers of the matrices.

# Exercise 3.4

- For this exercise we will import a matrix from the Internet and build a configuration with that matrix.
  - Open a browser (e.g. Internet Explorer, Firefox or Netscape) and visit the Website http://www.bioinformatics.nl/tools/pam.html
    This site is a calculator for PAM matrices
  - Calculate the PAM 30 matrix (you will have to change the PAM value suggested by the PAM calculator). Then transfer the data to a file called **pam30.mat** as follows:
    - Launch a text editor such as TextEdit (Mac) or NotePad (PC). Cut and paste the matrix into a new, empty file. Save this file as **pam30.mat**.
    - Edit **pam30.mat** in your text editor so that the top of the file looks like this:

```
# Pam 30
# This matrix was produced by "pam" Version 1.0.7 [13-Aug-03]
#
# PAM 30 substitution matrix, scale = ln(2)/2 = 0.346574
#
# Expected score = -5.06, Entropy = 2.57 bits
#
# Lowest score = -17, Highest score = 13
#
```

    - The last line of the file should be an empty line following the matrix. Remove any extraneous text at the bottom of the file. Compare your PAM 30 matrix with the PAM 250 matrix contained in the file **pam250.mat**.
- Use the PAM 30 matrix as in Exercise 3.3 to create an appropriate configuration (i.e. using proper origination and gap penalties).

# Exercise 3.5

- Find all global matches between the protein sequences

```
MVAILMWGVVAMWILVMVGTAVVIML
AILMVMGWILGTAVITVIML
```

using the following scoring schemes:

a) (-12, -1, Blosum 45)*
(-12, -5, Blosum 45)
(-15, -10, Blosum 45)

b) (-12, -1, Blosum 62)
(-12, -5, Blosum 62)
(-15, -10, Blosum 62)

c) (-10, -1, Pam 30)
(-10, -5, Pam 30)
(-5, -2, Pam 30)

d) (-10, -1, Pam 250)
(-10, -5, Pam 250)
(-5, -2, Pam 250)

How do you explain the different results obtained by (a) and (b), and by (c) and (d)?

* e.g., origination penalty: -12; gap penalty: -1; match/mismatch: Blosum 45 matrix.

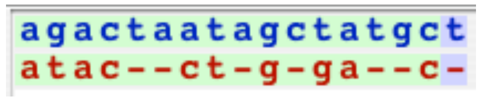
## End of Part III

Exercise 3.5 Solution

# Exercise 3.1 Solution

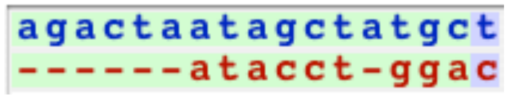
a) `agactaatagctatgct`
`atacctggac`

Origination penalty: 0



```
agactaatagctatgct
atac--ct-g-ga--c-
```

-2

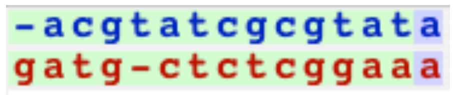


```
agactaatagctatgct
------atacct-ggac
```

Here we see expected results. Even though we have grouped the gaps in the second case, we still see a significant amount of matching.

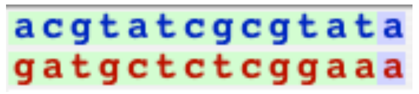
b) `acgtatcgcgtata`
`gatgctctcggaaa`

Origination penalty: 0



```
-acgtatcgcgtata
gatg-ctctcggaaa
```

-2

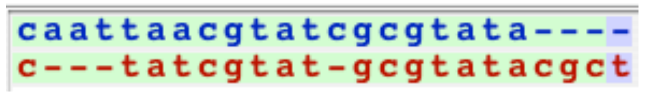


```
acgtatcgcgtata
gatgctctcggaaa
```

These are fairly dissimilar sequences. While the cost of two gaps in the first case was worth the additional matches they created, the cost of the gaps plus the origination penalty in the second case could not be recovered by having those additional matches.

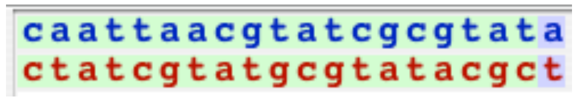
c) `caattaacgtatcgcgtata`
`ctatcgtatgcgtatacgct`

Origination penalty: 0



```
caattaacgtatcgcgtata----
c---tatcgtat-gcgtatacgct
```

-2



```
caattaacgtatcgcgtata
ctatcgtatgcgtatacgct
```

Here is a situation where the global algorithm is deficient. There are two regions where local subsequences correspond well in the first case, but once again these are erased by the cost of the gap origination penalty. We will reconsider this when we discuss semi-global and local alignment.

Return to Exercise 3.1

# Exercise 3.2 Solution

a) agactaatagctatgct
atacctggac

Origination penalty: 0

3.1
```
agactaatagctatgct
atac--ct-g-ga--c-
```
3.2
```
agactaatagctat-g-ct
------atacc--tggac-
```

-2

3.1
```
agactaatagctatgct
------atacct-ggac
```
3.2
```
agactaatagctatgct-
------atacc--tggac
```

Using this matrix produces two localized alignments even without an origination penalty.

b) acgtatcgcgtata
gatgctctcggaaa

Origination penalty: 0

3.1
```
-acgtatcgcgtata
gatg-ctctcggaaa
```
3.2
```
-ac-g-ta-tcgcgtata
ga-tgct-ctcg-g-aaa
```

-2

3.1
```
acgtatcgcgtata
gatgctctcggaaa
```
3.2
```
--acgta-tcgcgtata
gatgct-ctcg-g-aaa
```

Similarly, using this matrix leads to more focused localized scoring.

c) caattaacgtatcgcgtata
ctatcgtatgcgtatacgct

Origination penalty: 0

3.1
```
caattaacgtatcgcgtata----
c---tatcgtat-gcgtatacgct
```
3.2
```
caattaacgtatcgcgtata----
c---tatcgtat-gcgtatacgct
```

-2

3.1
```
caattaacgtatcgcgtata
ctatcgtatgcgtatacgct
```
3.2
```
caattaacgtatcgcgtata----
c---tatcgtat-gcgtatacgct
```

With this particular matrix the strong local alignments dominate.

Return to Exercise 3.2

# Exercise 3.5 Solution

a) (-12, -1, Blosum 45)

```
MVAILMWGVVAMWILVMVGTAV--VIML
--AILM---VMGWIL---GTAVITVIML
```

(-12, -5, Blosum 45)

```
MVAILMWGVVAMWILVMVGTAVVIML
--AILM---VMGWIL-GTAVITVIML
```

(-15, -10, Blosum 45)

```
MVAILMWGVVAMWILVMVGTAVVIML
--AILM---VMGWIL-GTAVITVIML
```

b) (-12, -1, Blosum 62)

```
MVAILMWGVVAMWILVMVGTAV--VIML
--AILM---VMGWIL---GTAVITVIML
```

(-12, -5, Blosum 62)

```
MVAILMWGVVAMWILVMVGTAVVIML
--AILM---VMGWILGT-AVITVIML
```

(-15, -10, Blosum 62)

```
MVAILMWGVVAMWILVMVGTAVVIML
AILMVMGWILGTAVITVI------ML
```

c) (-10, -1, Pam 30)

```
MVAILMWGVVAM-WILVMVGTAV--VIML
--AILM----VMGWIL---GTAVITVIML
```

(-10, -5, Pam 30)

```
MVAILMWGVVAMWILVMVGTAV--VIML
--AILM---VMGWIL---GTAVITVIML
```

(-5, -2, Pam 30)

```
MVAILMWGVVAM-WILVMVGTAV--VIML
--AILM---V-MGWIL---GTAVITVIML
```

d) (-10, -1, Pam 250)

```
MVAILMWGVVAMWILVMVGTAVV--IML
--AILM---VMGWIL---GTAVITVIML
```

(-10, -5, Pam 250)

```
MVAILMWGVVAMWILVMVGTAVVIML
--AILM---VMGWILG-TAVITVIML
```

(-5, -2, Pam 250)

```
MVAILMWGVVAMWILVMVGTAVV--IML
--AILM--VMG-WIL---GTAVITVIML
```

- **Note:** Proper choice of origination and gap penalties is crucial for the algorithm to succeed in finding the best matches. You can find extensive discussions of this issue online.

- The different outcomes seen in the two BLOSUM matrices and the two PAM matrices come about because the matrices, though related, favor different mismatches.

Return to Exercise 3.5