# Global alignment

Ben Langmead

JOHNS HOPKINS
WHITING SCHOOL
*of* ENGINEERING

Department of Computer Science

# Generalizing edit distance

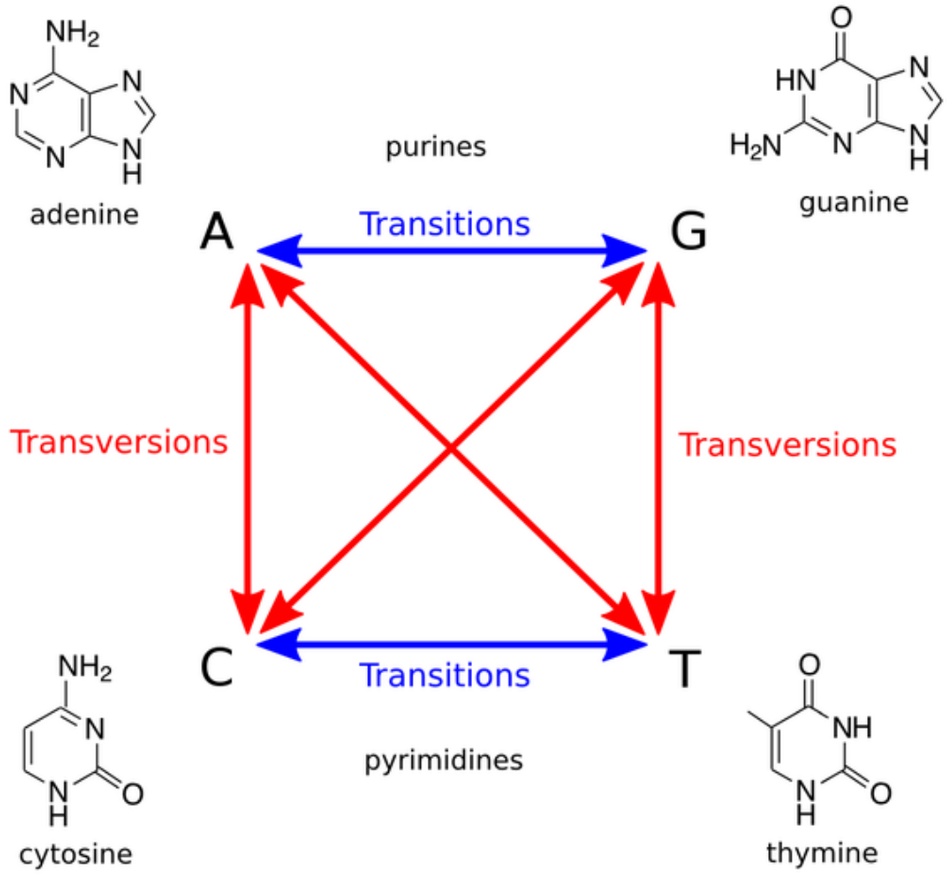
What if cost of edit could be ≠ 1?

E.g. sequencing errors tend to manifest as mismatches rather than gaps, so maybe gap penalty should be > mismatch penalty

It's also more likely for a genetic variant to be a mismatch rather than a gap

Also, some mismatches are more likely than others

Human *transition to transversion ratio* (AKA *ti/tv*) is ~2.1




http://en.wikipedia.org/wiki/Transversion

# Global alignment

$s(a, b)$ :

| | A | C | G | T | - |
|---|---|---|---|---|---|
| A | 0 | 4 | 2 | 4 | 8 |
| C | 4 | 0 | 4 | 2 | 8 |
| G | 2 | 4 | 0 | 4 | 8 |
| T | 4 | 2 | 4 | 0 | 8 |
| - | 8 | 8 | 8 | 8 | |

gaps in $b$

gaps in $a$

| | |
|---|---|
| 2 | *Transitions* (A↔G, C↔T) |
| 4 | *Transversions* (everything else) |
| 8 | *Gaps* |

Scoring function reflecting that transitions are more common than transversions and mismatches are more common than gaps

(Could have been even more specific, e.g. varying cost according to what character appears *opposite* a gap.)

# Global alignment

Let $D[0, j] = \sum_{k=0}^{j-1} s(-, y[k])$, and let $D[i, 0] = \sum_{k=0}^{i-1} s(x[k], -)$

Otherwise, let $D[i, j] = \min \begin{cases} D[i-1, j] + s(x[i-1], -) \\ D[i, j-1] + s(-, y[j-1]) \\ D[i-1, j-1] + s(x[i-1], y[j-1]) \end{cases}$

$s(a, b)$ assigns a cost to a particular gap or substitution

$s(a, b)$ :

| | A | C | G | T | - |
|---|---|---|---|---|---|
| A | 0 | 4 | 2 | 4 | 8 |
| C | 4 | 0 | 4 | 2 | 8 |
| G | 2 | 4 | 0 | 4 | 8 |
| T | 4 | 2 | 4 | 0 | 8 |
| - | 8 | 8 | 8 | 8 | |

2 — *Transitions* (A↔G, C↔T)

4 — *Transversions* (everything else)

8 — *Gaps*

# Global alignment: implementation

```
from numpy import zeros

def exampleCost(xc, yc):
    """ Cost function assigning 0 to match, 2 to transition, 4 to
        transversion, and 8 to a gap """
    if xc == yc: return 0 # match
    if xc == '-' or yc == '-': return 8 # gap
    minc, maxc = min(xc, yc), max(xc, yc)
    if minc == 'A' and maxc == 'G': return 2 # transition
    elif minc == 'C' and maxc == 'T': return 2 # transition
    return 4 # transversion

def globalAlignment(x, y, s):
    """ Calculate global alignment value of sequences x and y using
    dynamic programming.  Return global alignment value. """
D = zeros((len(x)+1, len(y)+1), dtype=int)
for j in xrange(1, len(y)+1):
    D[0, j] = D[0, j-1] + s('-', y[j-1])
for i in xrange(1, len(x)+1):
    D[i, 0] = D[i-1, 0] + s(x[i-1], '-')
for i in xrange(1, len(x)+1):
    for j in xrange(1, len(y)+1):
        D[i, j] = min(D[i-1, j-1] + s(x[i-1], y[j-1]), # diagonal
                      D[i-1, j  ] + s(x[i-1], '-'),    # vertical
                      D[i  , j-1] + s('-',    y[j-1])) # horizontal
return D, D[len(x), len(y)]
```

Use of new cost function

| | A | C | G | T | - |
|---|---|---|---|---|---|
| A | 0 | 4 | 2 | 4 | 8 |
| C | 4 | 0 | 4 | 2 | 8 |
| G | 2 | 4 | 0 | 4 | 8 |
| T | 4 | 2 | 4 | 0 | 8 |
| - | 8 | 8 | 8 | 8 | |

Extremely similar to edit distance algorithm

Python example: http://bit.ly/CG_DP_Global

# Global alignment: dynamic programming

**globalAlignment**
initialization:

```
D = zeros((len(x)+1, len(y)+1), dtype=int)
for j in xrange(1, len(y)+1):
    D[0, j] = D[0, j-1] + s('-', y[j-1])
for i in xrange(1, len(x)+1):
    D[i, 0] = D[i-1, 0] + s(x[i-1], '-')
```

| | ϵ | T | A | T | G | T | C | A | T | G | C |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ϵ | **0** | **8** | **16** | **24** | **32** | **40** | **48** | **56** | **64** | **72** | **80** |
| T | **8** | | | | | | | | | | |
| A | **16** | | | | | | | | | | |
| C | **24** | | | | | | | | | | |
| G | **32** | | | | | | | | | | |
| T | **40** | | | | | | | | | | |
| C | **48** | | | | | | | | | | |
| A | **56** | | | | | | | | | | |
| G | **64** | | | | | | | | | | |
| C | **72** | | | | | | | | | | |

$s(a, b)$

| | A | C | G | T | - |
|---|---|---|---|---|---|
| A | 0 | 4 | 2 | 4 | 8 |
| C | 4 | 0 | 4 | 2 | 8 |
| G | 2 | 4 | 0 | 4 | 8 |
| T | 4 | 2 | 4 | 0 | 8 |
| - | 8 | 8 | 8 | 8 | |

JOHNS HOPKINS
WHITING SCHOOL
*of* ENGINEERING

# Global alignment: dynamic programming

**globalAlignment**
loop:

```
for i in xrange(1, len(x)+1):
    for j in xrange(1, len(y)+1):
        D[i, j] = min(D[i-1, j-1] + s(x[i-1], y[j-1]), # diagonal
                      D[i-1, j  ] + s(x[i-1], '-'),    # vertical
                      D[i  , j-1] + s('-',    y[j-1])) # horizontal
```

| | ϵ | T | A | T | G | T | C | A | T | G | C |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ϵ | 0 | 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 | 72 | 80 |
| T | 8 | 0 | 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 | 72 |
| A | 16 | 8 | 0 | 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 |
| C | 24 | 16 | 8 | ? | | | | | | | |
| G | 32 | | | | | | | | | | |
| T | 40 | | | | | | | | | | |
| C | 48 | | | | | | | | | | |
| A | 56 | | | | | | | | | | |
| G | 64 | | | | | | | | | | |
| C | 72 | | | | | | | | | | |

$s(a, b)$

| | A | C | G | T | - |
|---|---|---|---|---|---|
| A | 0 | 4 | 2 | 4 | 8 |
| C | 4 | 0 | 4 | 2 | 8 |
| G | 2 | 4 | 0 | 4 | 8 |
| T | 4 | 2 | 4 | 0 | 8 |
| - | 8 | 8 | 8 | 8 | |

# Global alignment: dynamic programming

**globalAlignment**
loop:

```
for i in xrange(1, len(x)+1):
    for j in xrange(1, len(y)+1):
        D[i, j] = min(D[i-1, j-1] + s(x[i-1], y[j-1]), # diagonal
                      D[i-1, j  ] + s(x[i-1], '-'),    # vertical
                      D[i  , j-1] + s('-',    y[j-1])) # horizontal
```

| | ϵ | T | A | T | G | T | C | A | T | G | C |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ϵ | 0 | 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 | 72 | 80 |
| T | 8 | 0 | 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 | 72 |
| A | 16 | 8 | 0 | 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 |
| C | 24 | 16 | 8 | 2 | 10 | 18 | 24 | 32 | 40 | 48 | 56 |
| G | 32 | 24 | 16 | 10 | 2 | 10 | 18 | 26 | 34 | 40 | 48 |
| T | 40 | 32 | 24 | 16 | 10 | 2 | 10 | 18 | 26 | 34 | 42 |
| C | 48 | 40 | 32 | 24 | 18 | 10 | 2 | 10 | 18 | 26 | 34 |
| A | 56 | 48 | 40 | 32 | 26 | 18 | 10 | 2 | 10 | 18 | 26 |
| G | 64 | 56 | 48 | 40 | 32 | 26 | 18 | 10 | 6 | 10 | 18 |
| C | 72 | 64 | 56 | 48 | 40 | 34 | 26 | 18 | 12 | 10 | 10 |

$s(a, b)$

| | A | C | G | T | - |
|---|---|---|---|---|---|
| A | 0 | 4 | 2 | 4 | 8 |
| C | 4 | 0 | 4 | 2 | 8 |
| G | 2 | 4 | 0 | 4 | 8 |
| T | 4 | 2 | 4 | 0 | 8 |
| - | 8 | 8 | 8 | 8 | |

Optimal global alignment value

# Global alignment: getting the alignment

Backtrace procedure works exactly as it did for edit distance

| | ϵ | T | A | T | G | T | C | A | T | G | C |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ϵ | **0** | **8** | **16** | **24** | **32** | **40** | **48** | **56** | **64** | **72** | **80** |
| T | **8** | **0** | **8** | **16** | **24** | **32** | **40** | **48** | **56** | **64** | **72** |
| A | **16** | **8** | **0** | **8** | **16** | **24** | **32** | **40** | **48** | **56** | **64** |
| C | **24** | **16** | **8** | **2** | **10** | **18** | **24** | **32** | **40** | **48** | **56** |
| G | **32** | **24** | **16** | **10** | **2** | **10** | **18** | **26** | **34** | **40** | **48** |
| T | **40** | **32** | **24** | **16** | **10** | **2** | **10** | **18** | **26** | **34** | **42** |
| C | **48** | **40** | **32** | **24** | **18** | **10** | **2** | **10** | **18** | **26** | **34** |
| A | **56** | **48** | **40** | **32** | **26** | **18** | **10** | **2** | **10** | **18** | **26** |
| G | **64** | **56** | **48** | **40** | **32** | **26** | **18** | **10** | **6** | **10** | **18** |
| C | **72** | **64** | **56** | **48** | **40** | **34** | **26** | **18** | **12** | **10** | **10** |

Backtrace is O($m + n$) time

Optimal edit transcript containing only **D**s and **I**s is possible, (unlike edit distance)

```
T A C G T C A - G C
|  |   | | | |   | |
T A T G T C A T G C
```

+2 (transition)  +8 (gap)

# Global alignment

$s(a, b)$

| | A | C | G | T | - |
|---|---|---|---|---|---|
| A | 0 | 4 | 2 | 4 | 8 |
| C | 4 | 0 | 4 | 2 | 8 |
| G | 2 | 4 | 0 | 4 | 8 |
| T | 4 | 2 | 4 | 0 | 8 |
| - | 8 | 8 | 8 | 8 | |

Could also use *larger* scores for similarities and *smaller* scores for dissimilarities...

E.g. subtract one then change sign

| | A | C | G | T | - |
|---|---|---|---|---|---|
| A | 1 | -3 | -1 | -3 | -7 |
| C | -3 | 1 | -3 | -1 | -7 |
| G | -1 | -3 | 1 | -3 | -7 |
| T | -3 | -1 | -3 | 1 | -7 |
| - | -7 | -7 | -7 | -7 | |

...as long as we switch min to max:

```
for i in xrange(1, len(x)+1):
    for j in xrange(1, len(y)+1):
        D[i, j] = max(D[i-1, j-1] + s(x[i-1], y[j-1]), # diagonal
                      D[i-1, j  ] + s(x[i-1], '-'),    # vertical
                      D[i  , j-1] + s('-',    y[j-1])) # horizontal
```

# Global alignment

| X \ Y | ϵ | T | A | T | G | T | C | A | T | G | C |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ϵ | 0 | -7 | -14 | -21 | -28 | -35 | -42 | -49 | -56 | -63 | -70 |
| T | -7 | 1 | -6 | -13 | -20 | -27 | -34 | -41 | -48 | -55 | -62 |
| A | -14 | -6 | 2 | -5 | -12 | -19 | -26 | -33 | -40 | -47 | -54 |
| C | -21 | -13 | -5 | 1 | -6 | -13 | -18 | -25 | -32 | -39 | -46 |
| G | -28 | -20 | -12 | -6 | 2 | -5 | -12 | -19 | -26 | -31 | -38 |
| T | -35 | -27 | -19 | -11 | -5 | 3 | -4 | -11 | -18 | -25 | -32 |
| C | -42 | -34 | -26 | -18 | -12 | -4 | 4 | -3 | -10 | -17 | -24 |
| A | -49 | -41 | -33 | -25 | -19 | -11 | -3 | 5 | -2 | -9 | -16 |
| G | -56 | -48 | -40 | -32 | -24 | -18 | -10 | -2 | 2 | -1 | -8 |
| C | -63 | -55 | -47 | -39 | -31 | -25 | -17 | -9 | -3 | -1 | 0 |

$s(a, b)$

| | A | C | G | T | - |
|---|---|---|---|---|---|
| A | 1 | -3 | -1 | -3 | -7 |
| C | -3 | 1 | -3 | -1 | -7 |
| G | -1 | -3 | 1 | -3 | -7 |
| T | -3 | -1 | -3 | 1 | -7 |
| - | -7 | -7 | -7 | -7 | |

JOHNS HOPKINS
WHITING SCHOOL
*of* ENGINEERING

# Global alignment

| X \ Y | ϵ | T | A | T | G | T | C | A | T | G | C |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ϵ | 0 | 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 | 72 | 80 |
| T | 8 | 0 | 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 | 72 |
| A | 16 | 8 | 0 | 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 |
| C | 24 | 16 | 8 | 2 | 10 | 18 | 24 | 32 | 40 | 48 | 56 |
| G | 32 | 24 | 16 | 10 | 2 | 10 | 18 | 26 | 34 | 40 | 48 |
| T | 40 | 32 | 24 | 16 | 10 | 2 | 10 | 18 | 26 | 34 | 42 |
| C | 48 | 40 | 32 | 24 | 18 | 10 | 2 | 10 | 18 | 26 | 34 |
| A | 56 | 48 | 40 | 32 | 26 | 18 | 10 | 2 | 10 | 18 | 26 |
| G | 64 | 56 | 48 | 40 | 32 | 26 | 18 | 10 | 6 | 10 | 18 |
| C | 72 | 64 | 56 | 48 | 40 | 34 | 26 | 18 | 12 | 10 | 10 |

$s(a, b)$

| | A | C | G | T | - |
|---|---|---|---|---|---|
| A | 0 | 4 | 2 | 4 | 8 |
| C | 4 | 0 | 4 | 2 | 8 |
| G | 2 | 4 | 0 | 4 | 8 |
| T | 4 | 2 | 4 | 0 | 8 |
| - | 8 | 8 | 8 | 8 | |

Same backtrace

# Global alignment: summary

Matrix-filling dynamic programming algorithm is $O(mn)$ time and space

FillIng matrix is $O(mn)$ space and time, and yields global alignment value

Traceback is $O(m + n)$ steps, yields optimal alignment / edit transcript

Can set scores how we like.  Can have mix of positive and negative scores. In the dynamic programming we can maximize a similarity *score* or minimize a dissimilarity *penalty*.