Algorithms

- Classical algorithm (focus)
- FastDTW
- DTW with bands

Step 1: create a matrix D

DTW(s[1…n], t[1…m]) //s is reference sequence, t is query sequence

```
//Initialization
D = array[0…n,0…m]
D[0,0] = 0
for i=1 to n D[i,0] = ∞
for j=1 to m D[0,j] = ∞

//Fill in the matrix
for i=1 to n
    for j=1 to m
        cost = d(s[i],t[j])
        D[i,j] = cost + min(D[i-1,j], D[I,j-1], D[i-1,j-1])
return D[n,m]
```

(i,j-1)	(i,j)
(i-1,j-1)	(i-1,j)

Example: Initialization

Query sequence
j: 0…m, m=7

[7]	1	∞						
[6]	2	∞						
[5]	3	∞						
[4]	2	∞						
[3]	1	∞						
[2]	1	∞						
[1]	0	∞						
[0]		0	∞	∞	∞	∞	∞	∞
t			1	1	2	3	2	0
	s	[0]	[1]	[2]	[3]	[4]	[5]	[6]

i: 0…n, n=6
Reference sequence: s

Initialization

D = array[0…n,0…m]
D[0,0] = 0
for i=1 to n D[i,0] = ∞
for j=1 to m D[0,j] = ∞

Column i = 1

Query sequence
j: 0…m, m=7

[7]	1	∞	5					
[6]	2	∞	5					
[5]	3	∞	4					
[4]	2	∞	2					
[3]	1	∞	1					
[2]	1	∞	1					
[1]	0	∞	1					
[0]		0	∞	∞	∞	∞	∞	∞
t			1	1	2	3	2	0
	s	[0]	[1]	[2]	[3]	[4]	[5]	[6]

i: 0…n, n=6
Reference sequence: s

Loop

```
for i=1 to n
    for j=1 to m
        cost = d(s[i],t[j]) //d(s[i],t[j]) = |s[i]-t[j]|
        D[i,j] = cost + min(D[i-1,j], D[I,j-1], D[i-1,j-1])
return D[n,m]
```

Column i = 2

Query sequence
j: 0…m, m=7

[7]	1	∞	5	5				
[6]	2	∞	5	5				
[5]	3	∞	4	4				
[4]	2	∞	2	2				
[3]	1	∞	1	1				
[2]	1	∞	1	1				
[1]	0	∞	1	2				
[0]		0	∞	∞	∞	∞	∞	∞
t			1	1	2	3	2	0
	s	[0]	[1]	[2]	[3]	[4]	[5]	[6]

i: 0…n, n=6
Reference sequence: s

Loop

```
for i=1 to n
    for j=1 to m
        cost = d(s[i],t[j]) //d(s[i],t[j]) = |s[i]-t[j]|
        D[i,j] = cost + min(D[i-1,j], D[I,j-1], D[i-1,j-1])
return D[n,m]
```

Column i = 3

Query sequence
j: 0…m, m=7

[7]	1	∞	5	5	3			
[6]	2	∞	5	5	2			
[5]	3	∞	4	4	2			
[4]	2	∞	2	2	1			
[3]	1	∞	1	1	2			
[2]	1	∞	1	1	2			
[1]	0	∞	1	2	4			
[0]		0	∞	∞	∞	∞	∞	∞
t			1	1	2	3	2	0
	s	[0]	[1]	[2]	[3]	[4]	[5]	[6]

i: 0…n, n=6
Reference sequence: s

Loop

```
for i=1 to n
    for j=1 to m
        cost = d(s[i],t[j]) //d(s[i],t[j]) = |s[i]-t[j]|
        D[i,j] = cost + min(D[i-1,j], D[I,j-1], D[i-1,j-1])
return D[n,m]
```

Column i = 4, 5, 6

Query sequence
j: 0…m, m=7

[7]	1	∞	5	5	3	4	2	2
[6]	2	∞	5	5	2	2	1	3
[5]	3	∞	4	4	2	1	2	5
[4]	2	∞	2	2	1	2	2	4
[3]	1	∞	1	1	2	4	5	6
[2]	1	∞	1	1	2	4	5	6
[1]	0	∞	1	2	4	7	9	9
[0]		0	∞	∞	∞	∞	∞	∞
t			1	1	2	3	2	0
	s	[0]	[1]	[2]	[3]	[4]	[5]	[6]

i: 0…n, n=6
Reference sequence: s

Loop

```
for i=1 to n
    for j=1 to m
        cost = d(s[i],t[j]) //d(s[i],t[j]) = |s[i]-t[j]|
        D[i,j] = cost + min(D[i-1,j], D[I,j-1], D[i-1,j-1])
return D[n,m]
```

Example in R

```
library("dtw")
s<-c(1,1,2,3,2,0)
t<-c(0,1,1,2,3,2,1)
d<-dtw(t,s,dist.method="Manhattan", keep.internals=TRUE,step.pattern=symmetric1)
//query is t, s is reference
```

```
> d$costMatrix
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    2    4    7    9    9
[2,]    1    1    2    4    5    6
[3,]    1    1    2    4    5    6
[4,]    2    2    1    2    2    4
[5,]    4    4    2    1    2    5
[6,]    5    5    2    2    1    3
[7,]    5    5    3    4    2    2
```

```
> d$stepPattern
Step pattern recursion:
g[i,j] = min(
    g[i-1,j-1] +     d[i  ,j  ] ,
    g[i  ,j-1] +     d[i  ,j  ] ,
    g[i-1,j  ] +     d[i  ,j  ] ,
 )
 Normalization hint: NA
```

Other parameters (dtw)

> print(symmetric2) //this is the default step patterns

```
Step pattern recursion:
g[i,j] = min(
    g[i-1,j-1] + 2 * d[i  ,j  ] ,
    g[i  ,j-1] +     d[i  ,j  ] ,
    g[i-1,j  ] +     d[i  ,j  ] ,
 )
 Normalization hint: N+M
```

> d$dist //the alignment distance

```
[1] 2
```

Step 2: backtrack to get the warping path

OptimalWarpingPath(D) [1]

- Let $p_L=(n,m)$
- Suppose $p_i= (n_i,m_i)$ has been computed, compute $p_{i-1.}$
  - If $(n_i,m_i) = (1,1)$, terminate.
  - $p_{i-1}= \begin{cases} (1, m_i-1), & \text{if } n_i=1 \\ (n_i-1,1), & \text{if } m_i=1 \\ \operatorname{argmin}\{D(n_i-1, m_i-1), D(n_i-1, m_i), D(n_i, m_i-1)\}, & \text{otherwise} \end{cases}$
  - We take the lexicographically smallest pair in case "argmin" is not unique.
- Return the optimal path $p^*=(p_1,\ldots,p_L)$

Query sequence
j: 0…m, m=7

[7]	1	∞	5	5	3	4	2	2
[6]	2	∞	5	5	2	2	1	3
[5]	3	∞	4	4	2	1	2	5
[4]	2	∞	2	2	1	2	2	4
[3]	1	∞	1	1	2	4	5	6
[2]	1	∞	1	1	2	4	5	6
[1]	0	∞	1	2	4	7	9	9
[0]		0	∞	∞	∞	∞	∞	∞
t			1	1	2	3	2	0
	s	[0]	[1]	[2]	[3]	[4]	[5]	[6]

i: 0…n, n=6
Reference sequence: s

Backtracking to get warping path

$$p_{i-1} = \operatorname{argmin}\{D(n_i-1, m_i-1), D(n_i-1, m_i), D(n_i, m_i-1)\}$$

Example in R

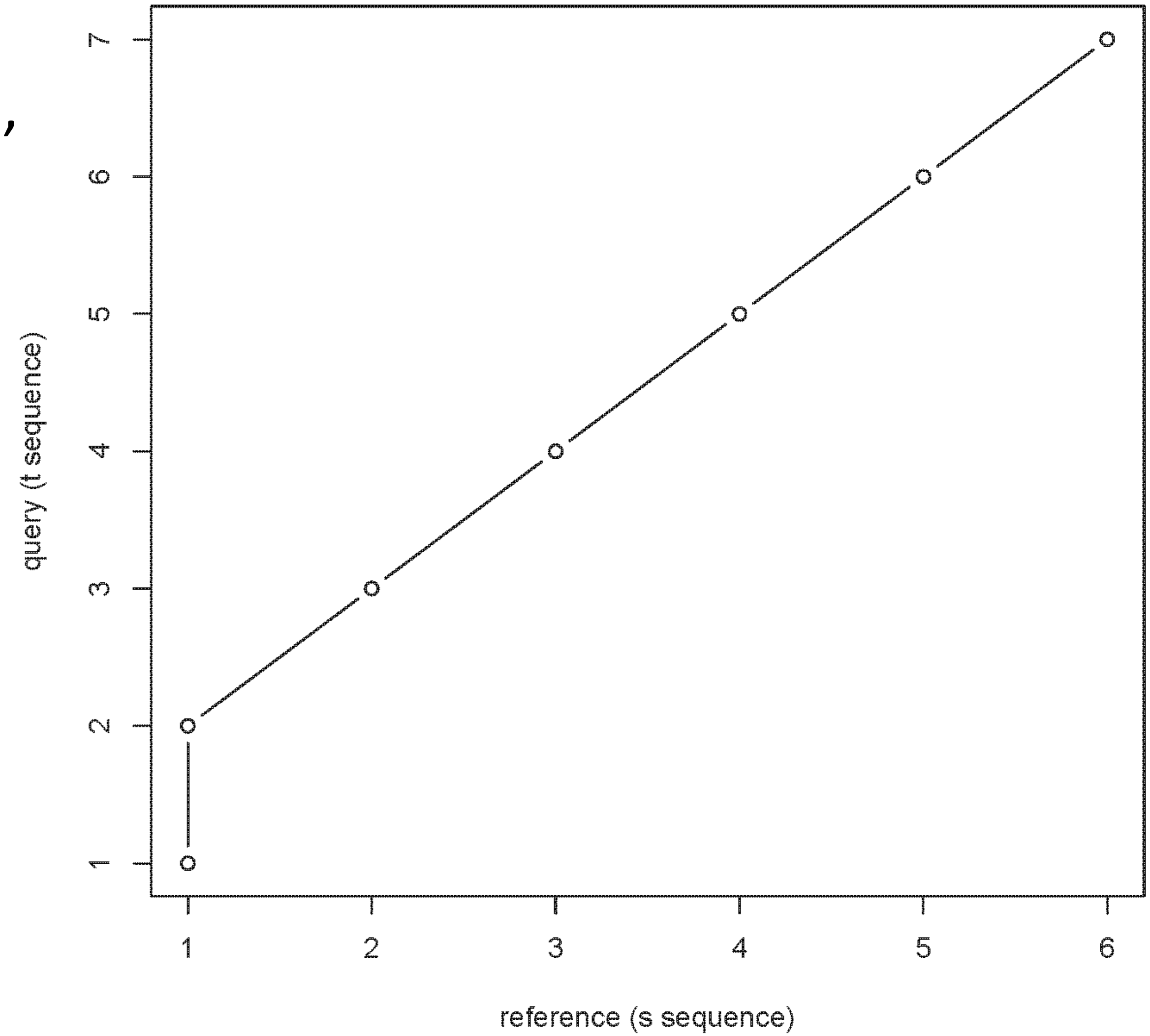
p*=((1,1),(1,2),(2,3),(3,4),
(4,5),(5,6), (6,7))

> d$index2

[1] 1 1 2 3 4 5 6

> d$index1

[1] 1 2 3 4 5 6 7



plot(d$index2,d$index1,xlab="reference (s sequence)",ylab="query (t sequence)",type="b")

Operations (Edit distance?)

- Insertion //**D[i-1,j] + d(s[i],t[j])**
  - $s_1 \ldots s_{i-1} s_i$
  - $t_1 \ldots t_j$
- Deletion //**D[i,j-1] + d(s[i],t[j])**
  - $s_1 \ldots s_i$
  - $t_1 \ldots t_{j-1} t_j$
- Match //**D[i-1,j-1] + d(s[i],t[j])**
  - $s_1 \ldots s_{i-1} s_i$
  - $t_1 \ldots t_{j-1} t_j$