1785

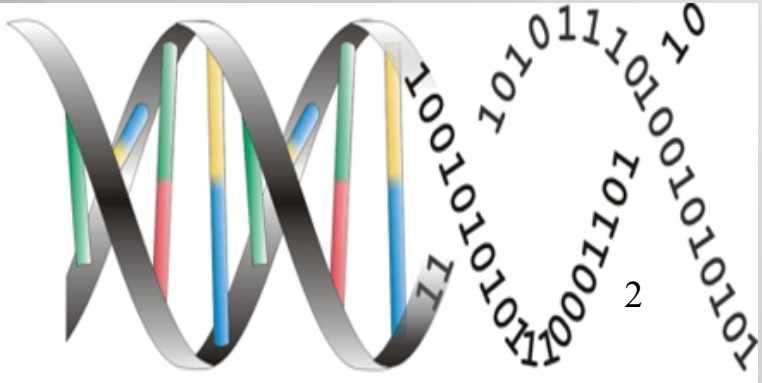
The University of Georgia

The University of Georgia

Programming for Computational & Systems Biology

Instructor: Paul Xie

Tue. & Thr. 9:30~10:45



Overview of Course

The University of Georgia

- Introduction to programming skills in computational and systems biology. Topics include real world examples, such as processing genome or proteome data, and analyzing large-scale data. The idea of “big data” will be emphasized to help students with their coding skills to discovering new knowledge in biomedical sciences and solving biomedical problems.

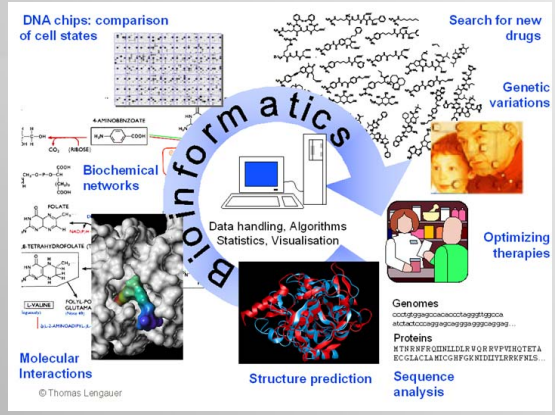
Course Format

The University of Georgia

- 1 hour lecture + 2 hour computer lab per week (attendance & participation 10%)
  - Please sign on the sheet
- 2 topics/week; 1 computational biology + 1 coding concept
- 1 assignment/week, send out by email on Thursday, due by Monday midnight (50%)
  - Please leave your email address
- 1 term paper (40%)

Topics

- -Omics data, e.g. Genomics, Proteomics…
- Basic Programming, e.g. I/O, variables, string, loop, regular express, array…etc.
- Data retrieval & processing
- Data analysis
- Model & building model
- Clustering and Classification
- Prediction

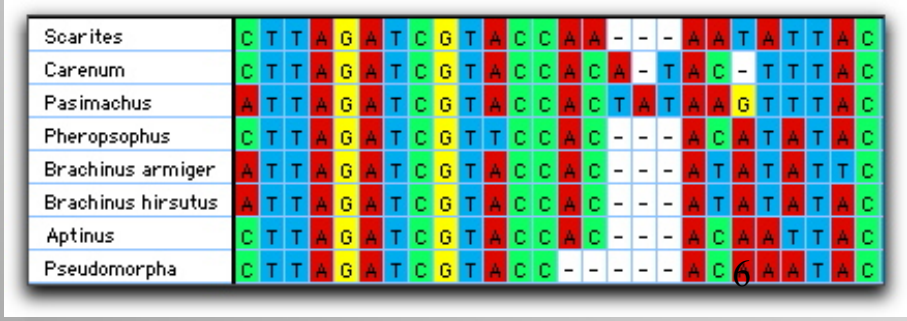


The University of Georgia

Loop, List & Dictionary Sequence Alignment

January 22-Feburary 21, 2020

Instructor: Paul Xie



This Week

- Sequence alignment
  - Global (N-W) and Local (S-W)
- Statistics model & Blast
- List
- Random number

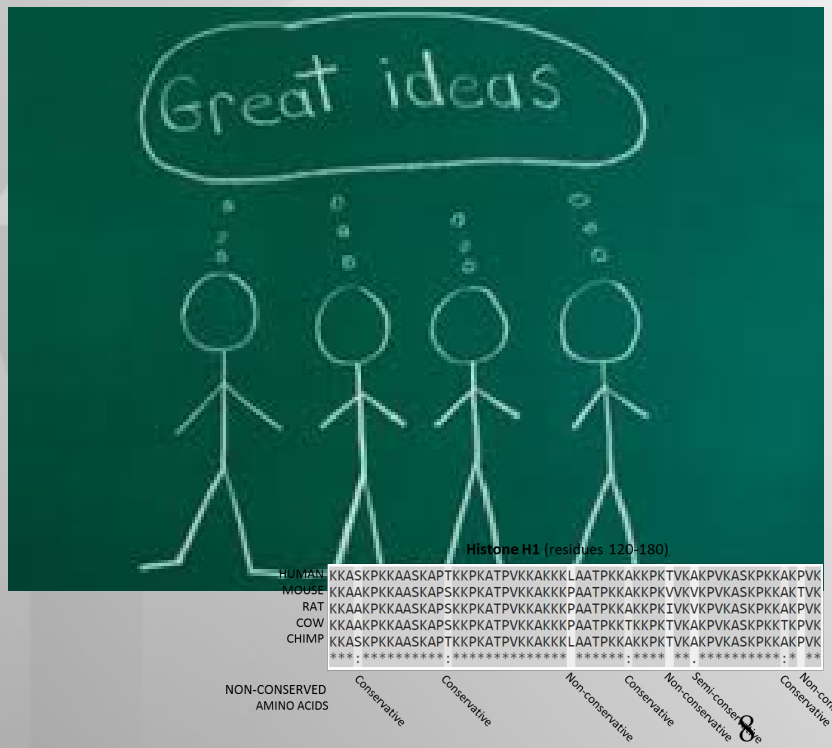
Histone H1 (residues 120-180)

```
HUMAN   KKASKPKKAASKAPTKKPKATPVKKAKKKLAATPKKAKKPKTVKAKPVKASKPKKAKPVK
MOUSE   KKAAKPKKAASKAPSKKPKATPVKKAKKKPAATPKKAKKPKVVKVKPVKASKPKKAKTVK
RAT     KKAAKPKKAASKAPSKKPKATPVKKAKKKPAATPKKAKKPKIVKVKPVKASKPKKAKPVK
COW     KKAAKPKKAASKAPSKKPKATPVKKAKKKPAATPKKTKKPKTVKAKPVKASKPKKTKPVK
CHIMP   KKASKPKKAASKAPTKKPKATPVKKAKKKLAATPKKAKKPKTVKAKPVKASKPKKAKPVK
        ***:**********:************** ******:**** **.**********:* **
```

NON-CONSERVED AMINO ACIDS

Conservative | Conservative | Non-conservative | Conservative | Non-conservative | Semi-conservative | Conservative | Non-conservative

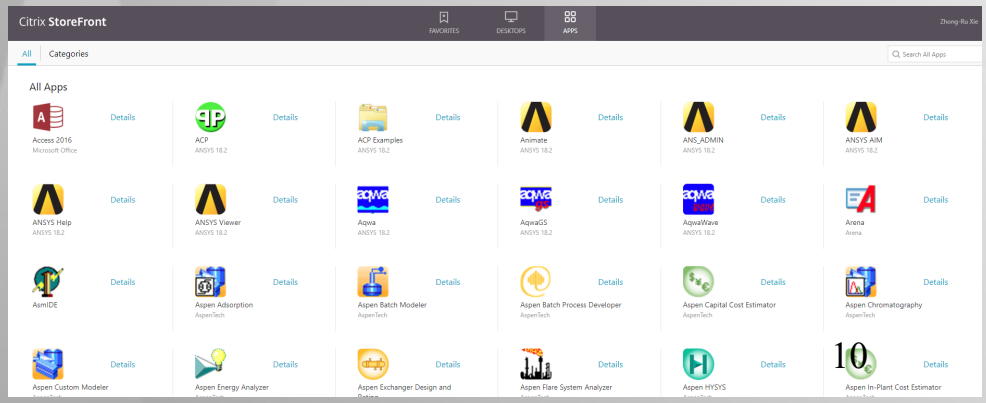
Group Discussion



Python Time

Open PyCharm

- Go to UGA CENGR Mylab
  - mylab.engr.uga.edu
- Login Citrix
  - Apps
  - PyCharm
  - New project



Iterations

The University of Georgia

- Loop
  - While loop
  - For loop
- Another input method – open file
  - F = open(“file path”,”r”)
  - Read and write
  - F.readline()
- Function : len()

Iterations

- String
  - String is a series of characters

Random Number

The University of Georgia

- Import random
- Random.randint()
- Random.randrange()
- Random.random()
- Random.uniform
- Random.choice()

Exercise

- Guess number
  1. Generate random number
  2. Input a guess number
  3. Check if the guess is correct
  4. iterate

Code

```
import random
def main():
    ans = random.randint(1,99)
    for i in range(7):
        guess = int(input("Please enter your guess: "))
        if guess > ans:
            print("go down")
        elif guess < ans:
            print("go up")
        else:
            print("correct!")
            break
    print("the correct answer is %d" %ans)
main()
```

Exercise

- Turtle and Rabbit race
  1. Dice (random number generator: 1~6)
  2. Record (the updated positions of turtle and rabbit)
  3. demonstrate track and current positions (visualized positions)
  4. The unfinished track (visualized track)
  5. The finish line
  6. The loop (iteration of dice rolling and run forward)
  7. Clean the previous line (system("cls")

Code

```
#rabbit and turtle race
import random
def main():
    stop = 50
   (turtle,rabbit) = (0,0)
   while(turtle < stop and rabbit < stop):
       (trail_t, trail_r) = ('', '')
 ----------------------------------
```

Code

The University of Georgia

```
----------------------------------
        turtle+= random.randint(1,6)
        for i in range(max(turtle,stop)):
            if i < turtle-1:
                trail_t+='.'
            elif i == turtle-1:
                trail_t+='T'
            elif i < stop-1:
                trail_t += ' '
            elif i == stop-1:
                trail_t += '|‘
----------------------------------
```

Code

The University of Georgia

```
----------------------------------
        rabbit += random.randint(1, 6)
        for i in range(max(rabbit, stop)):
            if i < rabbit - 1:
                trail_r += '.'
            elif i == rabbit - 1:
                trail_r += 'R'
            elif i < stop - 1:
                trail_r += ' '
            elif i == stop - 1:
                trail_r += '|‘
----------------------------------
```

Code

The University of Georgia

```
----------------------------------
    print(trail_t)
    print(trail_r)
    if (turtle >= stop):
        print('Trutle Wins')
    elif (rabbit >= stop):
        print('Rabbit Wins')
    else:
        input('Please press Enter to continue')

main()
```

Exercise

- Secret message (encryption)
  1. Input a message to deliver
  2. (secret code)
  3. Split the message and put into a random code
  4. Decoder

     ex: I/|u;#\+|DNgi+
     9(5uF!Ivsncof#O_w{JJaF6A6imX<h-D(
     7 NgL#*|Fr5p{(3\LQd-0Pfv8
     +s\T

Code

```
def main():
    import random
    import string
    message = input("input message:")
    l = len(message)
    print(l)
    code = ''
    cor = ''
    for i in range(l*10):
        cor+=str(i%10)
        if i % 10 == 0:
            code += message[i//10]
        else:
            code+=random.choice(string.printable)
    print(cor)
    print(code)
```

Exercise

- Random sequences/proteins?
  1. Randomly generate sequences
  2. Find (and record) the starting code ‘ATG’
  3. Count/Translate each 3 codes
  4. Calculate the length
  5. Find the stop code ‘TGA’
  6. Record the length

Code

```
def main():
        import random
        (start,stop,aa) = (0,0,0)
        seq = ''
      for i in range(10000):
            seq += random.choice('ATCG')
             if i > 1:
                  if start < 1:
                    if seq[i-2] == 'A' and seq[i-1] == 'T' and seq[i] == 'G':
                              start = i-2
                              aa +=1
                              print(start)
                              print(seq)
                    else:
                              if (i-start)%3 == 2:
                                        if seq[i-2] == 'T' and seq[i-1] == 'G' and
                  seq[i] == 'A':
```

Syntax

- Array
  - **List** is a collection which is ordered and changeable. Allows duplicate members. []
  - **Tuple** is a collection which is ordered and unchangeable. Allows duplicate members. ()
  - **Set** is a collection which is unordered and unindexed. No duplicate members.{}
  - **Dictionary** is a collection which is unordered, changeable and indexed. No duplicate members. {key:200}

Code

```
#array
def main():
    cars = ["Ford","Volvo","BMW"]
    for i in cars:
        print(i)

main()
```

Code

```
#array
def show_array(cars):
    print(len(cars),'cars.\nThey are:')
    for i in cars:
        print(i)
    for i in range(10):
        print('-',end='')
    print()
……….
```

Code

```
……..
def main():
    cars = ["Ford","Volvo","BMW"]
    show_array(cars)
    cars.append("Honda")
    show_array(cars)
    cars.pop(2)
    show_array(cars)
    cars.insert(1,"Toyota")
    show_array(cars)
    print(cars.index("Honda"))
    cars.remove("Volvo")
    show_array(cars)
    cars.sort()
    show_array(cars)
    cars.reverse()
    show_array(cars)
main()
```

Practice

- Average and Standard deviation

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i-\overline{x})^2}=\sqrt{\frac{1}{N}\left(\sum_{i=1}^{N}x_i^2\right)-(\overline{x})^2}=\sqrt{\left(\frac{1}{N}\sum_{i=1}^{N}x_i^2\right)-\left(\frac{1}{N}\sum_{i=1}^{N}x_i\right)^2}.$$

Syntax

- Array
- Dictionary
  - Key-value

Code

```
#dictionary
def main():
    cars =  {
        "brand": "Toyota",
        "model": "Corolla",
        "year": 2003
    }
    print(cars)
    cars["year"] = 2018
    cars["color"] = "red"
    cars["owner"] = "paul"
    cars["seller"] = "Toyota Athens"
    cars["plate"] = "Georgia"
    ----------------------------
```

----------Code

```
print(len(cars))
    for i in cars:
        print(i)
    for i in cars.keys():
        print(i)
    for i in cars.values():
        print(i)
    for x,y in cars.items():
        print(x,y)
    cars.popitem()
    cars.pop("seller")
    del cars[]
    print(cars)
```

Exercise

- Calculate the occurrence & frequency of amino acids (or nucleotides)
  - Open file(s)
  - Dictionaries
  - Loops
  - Counting
  - presentation

Code

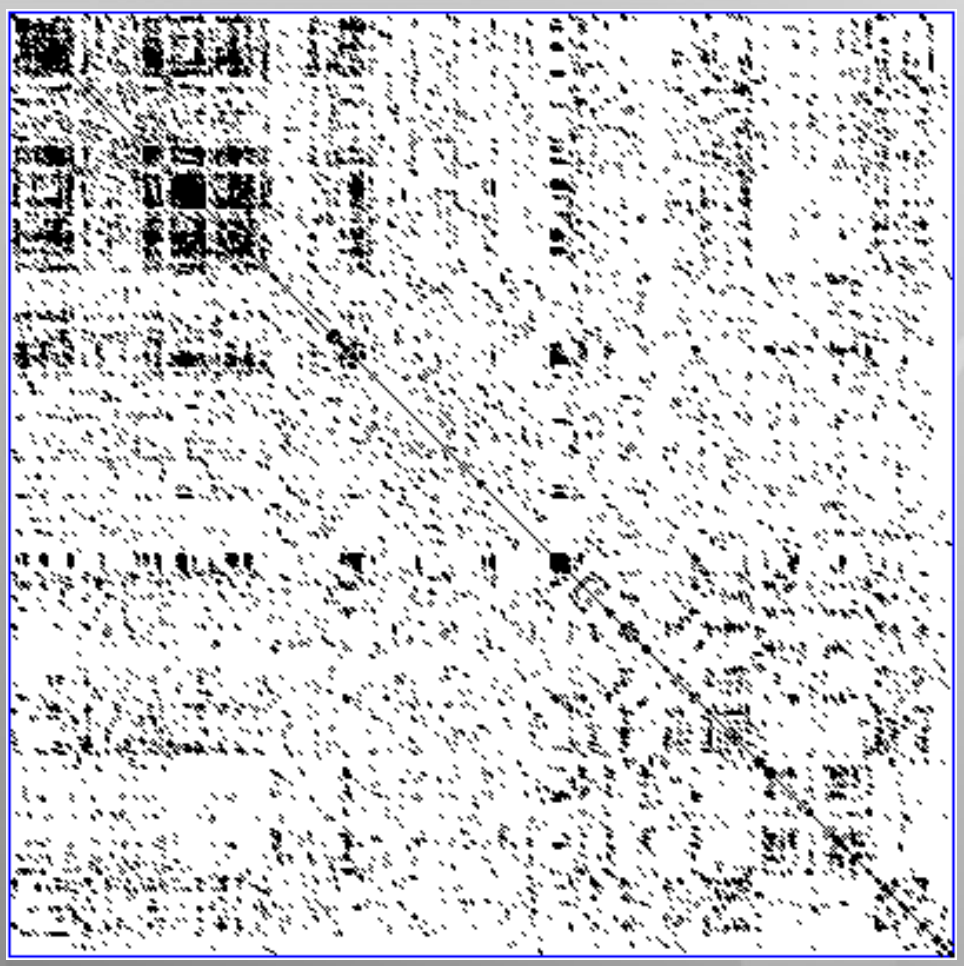
```
#dictionary
def main():
    f1 = open("M:\\tubulin_a.txt","r")
    seq1 = f1.readline()
    f2 = open("M:\\tubulin_b.txt", "r")
    seq2 = f2.readline()
    aa1 = {}
    aa2 = {}
-------------------------------------
```

Code

The University of Georgia

```
----------------------------------------
for i in range(len(seq1)):
    if seq1[i] in aa1:
        aa1[seq1[i]]+=1
    else:
        aa1[seq1[i]]=1
for i in range(len(seq2)):
    if seq2[i] in aa2:
        aa2[seq2[i]] += 1
    else:
        aa2[seq2[i]] = 1
for i in aa1:
    print(i,aa1[i],aa1[i]/len(seq1),aa2[i],aa2[i]/len(seq2))
```

Dot Plot



- Dot matrix approach
- Visually identify certain features
  - Insertions (indel)
  - repeats
- Degree of similarity
- Visually
- Manually vs. automatically

Needleman-Wunsch

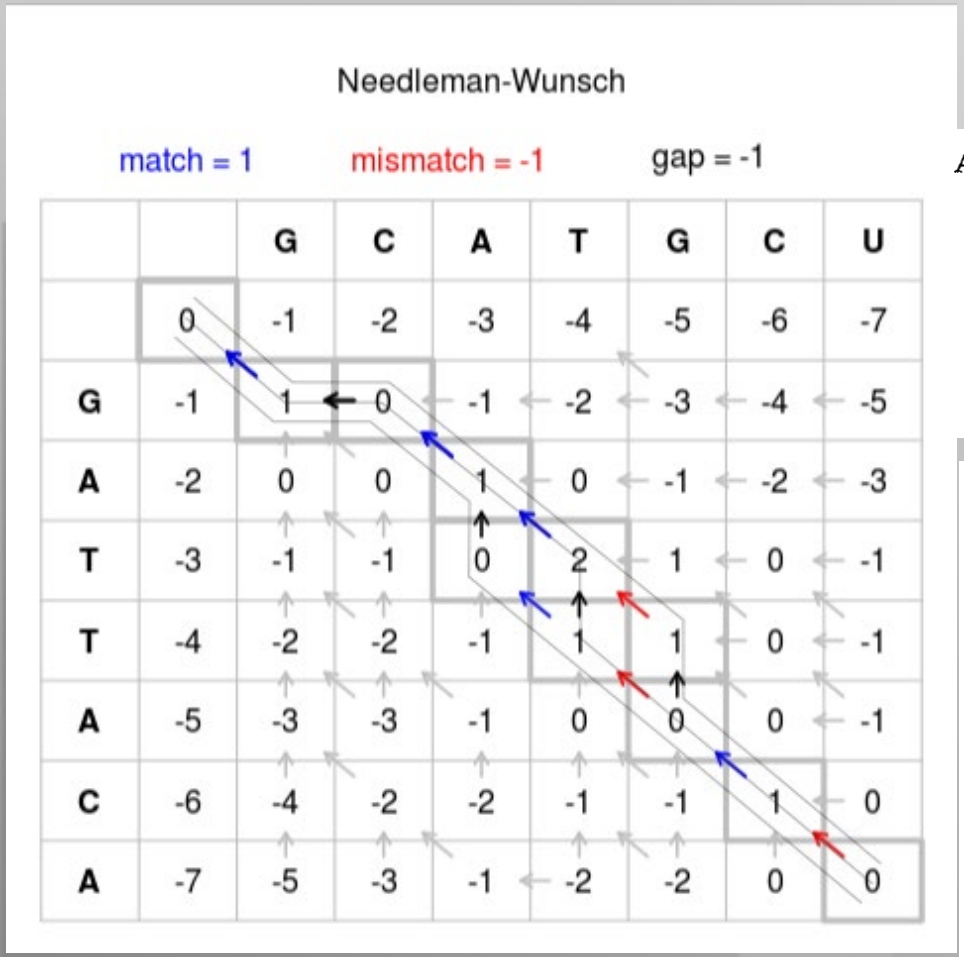
Needleman-Wunsch

match = 1 mismatch = -1 gap = -1

		G	C	A	T	G	C	U
	0	-1	-2	-3	-4	-5	-6	-7
G	-1	1	0	-1	-2	-3	-4	-5
A	-2	0	0	1	0	-1	-2	-3
T	-3	-1	-1	0	2	1	0	-1
T	-4	-2	-2	-1	1	1	0	-1
A	-5	-3	-3	-1	0	0	0	-1
C	-6	-4	-2	-2	-1	-1	1	0
A	-7	-5	-3	-1	-2	-2	0	0

- Dynamic programming
  - a method for solving a complex problem by breaking it down into a collection of simpler subproblems, solving each of those subproblems just once, and storing their solutions.
  - GCA-TGCU
  - | | | |
  - G- ATTACA

Needleman-Wunsch



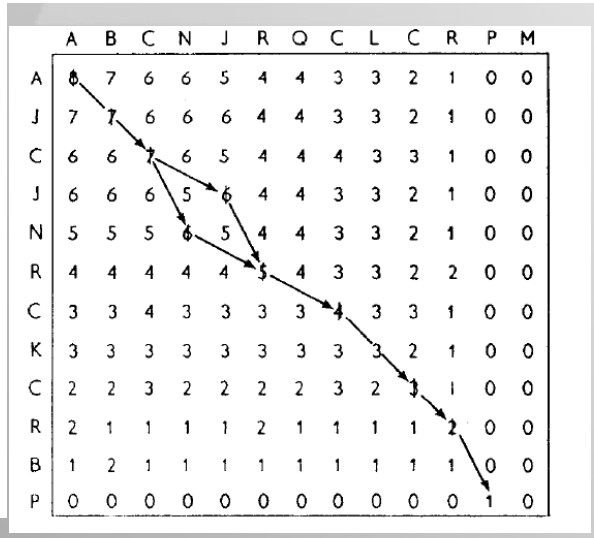
- Global alignment

A General Method Applicable to the Search for Similaritie in the Amino Acid Sequence of Two Proteins

SAUL B. NEEDLEMAN AND CHRISTIAN D. WUNSCH

Department of Biochemistry, Northwestern University, and Nuclear Medicine Service, V. A. Research Hospital Chicago, Ill. 60611, U.S.A.

(*Received 21 July 1969*)



	A	B	C	N	J	R	Q	C	L	C	R	P	M
A	8	7	6	6	5	4	4	3	3	2	1	0	0
J	7	7	6	6	6	4	4	3	3	2	1	0	0
C	6	6	7	6	5	4	4	4	3	3	1	0	0
J	6	6	6	5	6	4	4	3	3	2	1	0	0
N	5	5	5	6	5	4	4	3	3	2	1	0	0
R	4	4	4	4	4	5	4	3	3	2	2	0	0
C	3	3	4	3	3	3	3	4	3	3	1	0	0
K	3	3	3	3	3	3	3	3	3	2	1	0	0
C	2	2	3	2	2	2	2	3	2	3	1	0	0
R	2	1	1	1	1	2	1	1	1	1	2	0	0
B	1	2	1	1	1	1	1	1	1	1	1	0	0
P	0	0	0	0	0	0	0	0	0	0	0	1	0

Needleman-Wunsch

The University of Georgia

- Global alignment

-	-	A	T	C	G	A	C
-	0	-4	-8	-12	-16	-20	-24
C	-4	-3	-7	-3	-7	-11	-15
A	-8	1	-3	-7	-6	-2	-6
T	-12	-3	6	2	-2	-6	-5
A	-16	-7	2	3	-1	3	-1
C	-20	-11	-2	-1	0	-1	8

Exercise

- The 1st step of sequence alignment
  - Open file(s) or define 2 sequences
  - Lengths of 2 sequences
  - 2D list
  - Parameters: match, mismatch, gap
  - Loops
  - presentation

Code

```
#sequence alignment step 1
def main():
    seq1 = 'GCATGCT'
    seq2 = 'GATTACA'
    l1 = len(seq1)
    l2 = len(seq2)
    match = 1
    m = [[0]*l1 for i in range(l2)]
    print(",seq1)
    for i in range(l2):
        print(seq2[i],end=")
        for j in range(l1):
            print(m[j][i],end=")
        print()
main()
```

Syntax

- Function
  - Define a function
  - Call a function
  - Parameter(s), e.g. def function(parameters):
  - return

Code

```
#function
def hello(name):
    print('hello',name)
def addition(a,b):
    x = a + b
    return(x)
def main():
    student = ['Lei','Meichen','Yifei','Priyanka','Jen-Hung','Chih-Kai','Jiahui']
    for i in student:
        hello(i)
    x = addition(2,3)
    print(x)
main()
```

Exercise

- The 2nd step of sequence alignment
  - Open file(s) or define 2 sequences
  - Lengths of 2 sequences
  - 2D list
  - Parameters: match, mismatch, gap
  - Loops
  - Presentation
  - Function(s)

Code

```
#sequence alignment step 2
def show_align(seq1, seq2, l1, l2, m):
    print(", seq1)
    for i in range(l2):
        print(seq2[i], end=")
        for j in range(l1):
            print(m[j][i], end=")
        print()
-------------------------------------
```

Code

The University of Georgia

```
def main():
    seq1 = 'GCATGCT'
    seq2 = 'GATTACA'
    l1 = len(seq1)
    l2 = len(seq2)
    match = 1
    G = [[0] *l1 for i in range(l2)]
    S = [[' '] * l1 for i in range(l2)]
    show_align(seq1, seq2, l1, l2, G)
    for i in range(l2):
        for j in range(l1):
            if seq2[i] == seq1[j]:
                G[j][i] = match
                S[j][i] = 'X'
    show_align(seq1, seq2, l1, l2, G)
    show_align(seq1, seq2, l1, l2, S)
main()
```

Exercise

- The 3rd step of sequence alignment
  - Open file(s) or define 2 sequences
  - Lengths of 2 sequences
  - 2D list
  - Parameters: match, mismatch, gap
  - Loops
  - Presentation
  - Function(s)
  - Add 1 more row + column to the matrix

Code

```
#sequence alignment step 3
def show_align(seq1, seq2, l1, l2, m, num):
    print('     ',end='')
    for j in range(l1-1):
        print(seq1[j],end=' ')
    print()
    for i in range(l2):
        if i == 0:
            print('  ',end='')
        else:
            print("%2s" %seq2[i-1], end='')
        for j in range(l1):
            if num:
                print("%2d" %m[j][i], end='')
            else:
                print("%2s" %m[j][i], end='')
        print()
-----------------------------------------------
```

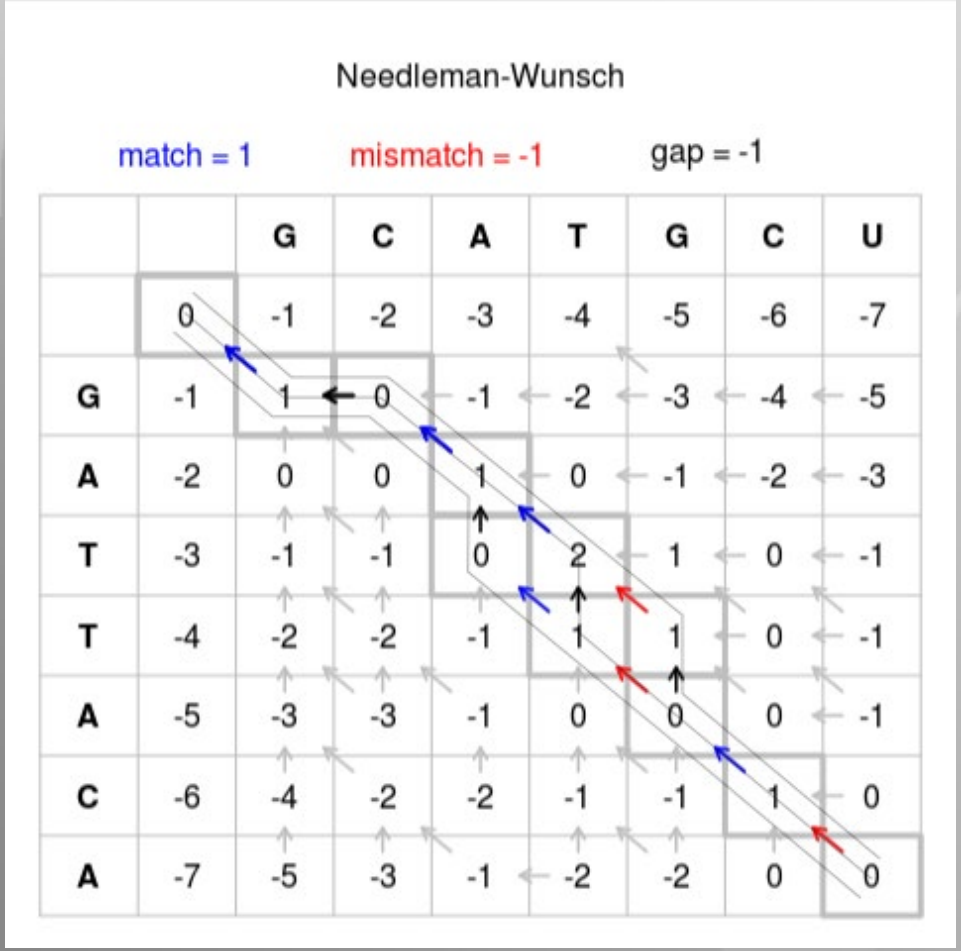
Code

```
#sequence alignment step 3
-----------------------------------------------
def alignment(seq1,seq2,l1,l2,match,mismatch):
    G = [[0] *l1 for i in range(l2)]
    S = [[' '] * l1 for i in range(l2)]
    show_align(seq1, seq2, l1, l2, G,'True')
    for i in range(1,l2):
        for j in range(1,l1):
            if seq2[i-1] == seq1[j-1]:
                G[j][i] = match
                S[j][i] = 'X'
            else:
                G[j][i] = mismatch
    show_align(seq1, seq2, l1, l2, G,True)
    show_align(seq1, seq2, l1, l2, S,False)
-----------------------------------------------
```

Code

```
#sequence alignment step 3
-----------------------------------------------
def main():
    seq1 = 'GCATGCT'
    seq2 = 'GATTACA'
    l1 = len(seq1) + 1
    l2 = len(seq2) + 1
    match = 4
    mismatch = -3
    alignment(seq1,seq2,l1,l2,match,mismatch)
main()
```

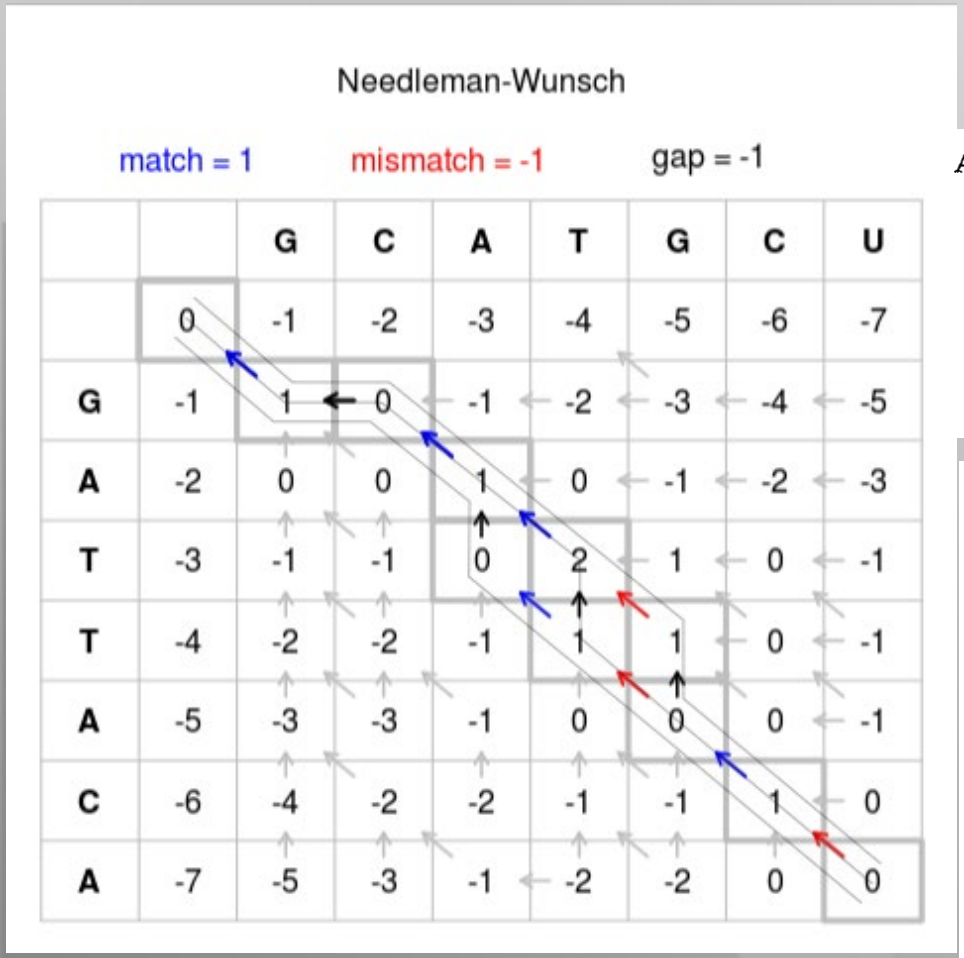
Needleman-Wunsch



		G	C	A	T	G	C	U
	0	-1	-2	-3	-4	-5	-6	-7
G	-1	1	0	-1	-2	-3	-4	-5
A	-2	0	0	1	0	-1	-2	-3
T	-3	-1	-1	0	2	1	0	-1
T	-4	-2	-2	-1	1	1	0	-1
A	-5	-3	-3	-1	0	0	0	-1
C	-6	-4	-2	-2	-1	-1	1	0
A	-7	-5	-3	-1	-2	-2	0	0

- Dynamic programming
  - a method for solving a complex problem by breaking it down into a collection of simpler subproblems, solving each of those subproblems just once, and storing their solutions.

Needleman-Wunsch



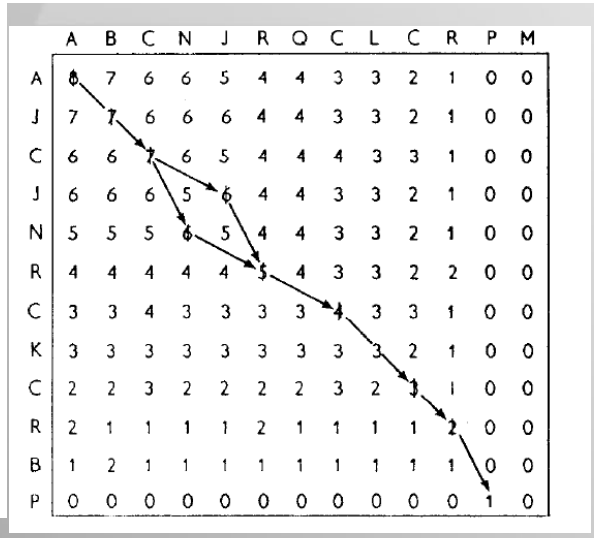
- Global alignment

A General Method Applicable to the Search for Similaritie in the Amino Acid Sequence of Two Proteins

SAUL B. NEEDLEMAN AND CHRISTIAN D. WUNSCH

Department of Biochemistry, Northwestern University, and Nuclear Medicine Service, V. A. Research Hospital Chicago, Ill. 60611, U.S.A.

(*Received 21 July 1969*)



	A	B	C	N	J	R	Q	C	L	C	R	P	M
A	8	7	6	6	5	4	4	3	3	2	1	0	0
J	7	7	6	6	6	4	4	3	3	2	1	0	0
C	6	6	7	6	5	4	4	4	3	3	1	0	0
J	6	6	6	5	6	4	4	3	3	2	1	0	0
N	5	5	5	6	5	4	4	3	3	2	1	0	0
R	4	4	4	4	4	5	4	3	3	2	2	0	0
C	3	3	4	3	3	3	3	4	3	3	1	0	0
K	3	3	3	3	3	3	3	3	3	2	1	0	0
C	2	2	3	2	2	2	2	3	2	3	1	0	0
R	2	1	1	1	1	2	1	1	1	1	2	0	0
B	1	2	1	1	1	1	1	1	1	1	1	0	0
P	0	0	0	0	0	0	0	0	0	0	0	1	0

Needleman-Wunsch

The University of Georgia

- Global alignment

-	-	A	T	C	G	A	C
-	0	-4	-8	-12	-16	-20	-24
C	-4	-3	-7	-3	-7	-11	-15
A	-8	1	-3	-7	-6	-2	-6
T	-12	-3	6	2	-2	-6	-5
A	-16	-7	2	3	-1	3	-1
C	-20	-11	-2	-1	0	-1	8

Exercise

- The 4th step of sequence alignment
  - Open file(s) or define 2 sequences
  - Lengths of 2 sequences
  - 2D list
  - Parameters: match, mismatch, gap
  - Loops
  - Presentation
  - Function(s)
  - Dynamic Programming

Code

```
#sequence alignment step 4
def show_align(seq1, seq2, l1, l2, m, num):
    print('     ',end='')
    for j in range(l1-1):
        print(seq1[j],end=' ')
    print()
    for i in range(l2):
        if i == 0:
            print('  ',end='')
        else:
            print("%2s" %seq2[i-1], end='')
        for j in range(l1):
            if num:
                print("%2d" %m[j][i], end='')
            else:
                print("%2s" %m[j][i], end='')
        print()
-----------------------------------------------
```

Code

```
-----------------------------------------------------
def align_seq(seq1,seq2,l1,l2,S):
    (s1,s2,a)=("","","")
    while l1>0 or l2>0:
        if S[l2][l1] == 'D':
            s1 = seq1[l1 - 1]+s1
            s2 = seq2[l2 - 1]+s2
            if seq1[l1 - 1] == seq2[l2 - 1]:
                a = '|'+a
            else:
                a = ' '+a
            l1=l1-1
            l2=l2-1
-------------------------------------------
        elif S[l2][l1] == 'U':
            s1 = '-'+s1
            s2 = seq2[l2 - 1]+s2
            a = ' '+a
            l2=l2-1
```

Code

```
-----------------------------------------------
elif S[l2][l1] == 'U':
        s1 = '-'+s1
        s2 = seq2[l2 - 1]+s2
        a = ' '+a
        l2=l2-1
    elif S[l2][l1] == 'L':
        s1 = seq1[l1 - 1]+s1
        s2 = '-'+s2
        a = ' '+a
        l1=l1-1
  print(s1)
  print(a)
  print(s2)
---------------------------------------------
```

Code

The University of Georgia

```
-----------------------------------------------
def alignment(seq1,seq2,l1,l2,match,mismatch,gap):
    G = [[0] *l1 for i in range(l2)]
    S = [[' '] * l1 for i in range(l2)]
    show_align(seq1, seq2, l1, l2, G,'True')
    (l,u,d) = (0,0,0)
for i in range(l2):
    for j in range(l1):
        if i == 0:
            if j == 0:
                G[i][j] = 0
            else:
                G[i][j] = G[i][j-1]+gap
                S[i][j] = 'L'
        else:
            if j == 0:
                G[i][j] = G[i-1][j]+gap
                S[i][j] = 'U‘
-----------------------------------------------
```

Code

The University of Georgia

```
---------------------------------------------------
        else:
            u = G[i-1][j]+gap
            l = G[i][j-1]+gap
            if seq2[i-1] == seq1[j-1]:
                d = G[i-1][j-1] + match
            else:
                d = G[i - 1][j - 1] + mismatch
            G[i][j] = max(u,l,d)
            if G[i][j] == d:
                S[i][j] = 'D'
            elif G[i][j] == u:
                S[i][j] = 'U'
            else:
                S[i][j] = 'L'
show(seq1,seq2, l1, l2, G, True)
show(seq1, seq2, l1, l2, S,False)
align_seq(seq1,seq2,l1-1,l2-1,S)
---------------------------------------------------
```

Code

```
#sequence alignment step 4
-----------------------------------------------
def main():
    seq1 = 'GCATGCT'
    seq2 = 'GATTACA'
    l1 = len(seq1) + 1
    l2 = len(seq2) + 1
    match = 1
    mismatch = -1
    gap = -1
    alignment(seq1,seq2,l1,l2,match,mismatch,gap)
main()
```

Exercise

- The 5th step of sequence alignment
  - Open file(s) or define 2 sequences
  - Lengths of 2 sequences
  - 2D list
  - Parameters: match, mismatch, gap
  - Loops
  - Presentation
  - Function(s)
  - Dynamic Programming
  - Final aligned sequences

Code

```
#sequence alignment step 5
def show_align(seq1, seq2, l1, l2, m, num):
    print('     ',end='')
    for j in range(l1-1):
        print(seq1[j],end=' ')
    print()
    for i in range(l2):
        if i == 0:
            print('  ',end='')
        else:
            print("%2s" %seq2[i-1], end='')
        for j in range(l1):
            if num:
                print("%2d" %m[j][i], end='')
            else:
                print("%2s" %m[j][i], end='')
        print()
-----------------------------------------------
```

Code

The University of Georgia

```
sequence alignment step 5
-----------------------------------------------
def align_seq(seq1,seq2,l1,l2,S):
    (s1,s2,a)=("","","")
    while l1>0 or l2>0:
        if S[l2][l1] == 'D':
            s1 = seq1[l1 - 1]+s1
            s2 = seq2[l2 - 1]+s2
            if seq1[l1 - 1] == seq2[l2 - 1]:
                a = '|'+a
            else:
                a = ' '+a
            l1=l1-1
            l2=l2-1
-------------------------------------------
```

Code

sequence alignment step 5

```
-----------------------------------------------
elif S[l2][l1] == 'U':
        s1 = '-'+s1
        s2 = seq2[l2 - 1]+s2
        a = ' '+a
        l2=l2-1
    elif S[l2][l1] == 'L':
        s1 = seq1[l1 - 1]+s1
        s2 = '-'+s2
        a = ' '+a
        l1=l1-1
  print(s1)
  print(a)
  print(s2)
---------------------------------------------
```

Code

```
-----------------------------------------------
def alignment(seq1,seq2,l1,l2,match,mismatch,gap):
    G = [[0] *l1 for i in range(l2)]
    S = [[' '] * l1 for i in range(l2)]
    show_align(seq1, seq2, l1, l2, G,'True')
    (l,u,d) = (0,0,0)
for i in range(l2):
    for j in range(l1):
        if i == 0:
            if j == 0:
                G[i][j] = 0
            else:
                G[i][j] = G[i][j-1]+gap
                S[i][j] = 'L'
        else:
            if j == 0:
                G[i][j] = G[i-1][j]+gap
                S[i][j] = 'U‘

---------------------------------------------
```

Code

The University of Georgia

```
---------------------------------------------------
        else:
            u = G[i-1][j]+gap
            l = G[i][j-1]+gap
            if seq2[i-1] == seq1[j-1]:
                d = G[i-1][j-1] + match
            else:
                d = G[i - 1][j - 1] + mismatch
            G[i][j] = max(u,l,d)
            if G[i][j] == d:
                S[i][j] = 'D'
            elif G[i][j] == u:
                S[i][j] = 'U'
            else:
                S[i][j] = 'L'
show(seq1,seq2, l1, l2, G, True)
show(seq1, seq2, l1, l2, S,False)
align_seq(seq1,seq2,l1-1,l2-1,S)
---------------------------------------------------
```

Code

```
#sequence alignment step 4
-----------------------------------------------
def main():
    seq1 = 'GCATGCT'
    seq2 = 'GATTACA'
    l1 = len(seq1) + 1
    l2 = len(seq2) + 1
    match = 1
    mismatch = -1
    gap = -1
    alignment(seq1,seq2,l1,l2,match,mismatch,gap)
main()
```

DNA → AA

The University of Georgia

Second Codon Letter

First Codon Letter	T	C	A	G	Third Codon Letter
T	TTT – phe TTC – phe TTA – leu TTG – leu	TCT – ser TCC – ser TCA – ser TCG – ser	TAT – tyr TAC – tyr TAA – stop TAG – stop	TGT – cys TGC – cys TGA – stop TGG – trp	T C A G
C	CTT – leu CTC – leu CTA – leu CTG – leu	CCT – pro CCC – pro CCA – pro CCG – pro	CAT – his CAC – his CAA – gln CAG – gln	CGT – arg CGC – arg CGA – arg CGG – arg	T C A G
A	ATT – ile ATC – ile ATA – ile ATG – start/met	ACU – thr ACC – thr ACA – thr ACG – thr	AAT – asn AAC – asn AAA – lys AAG – lys	AGT – ser AGC – ser AGA – arg AGG – arg	T C A G
G	GTT – val GTC – val GTA – val GTG – val	GCT – ala GCC – ala GCA – ala GCG – ala	GAT – asp GAC – asp GAA – glu GAG – glu	GGT – gly GGC – gly GGA – gly GGG – gly	T C A G

Code

```
#dictionary
import random
def DNA_generate(l):
    dna = ''
    for i in range(l):
        dna+= random.choice('ATCG')
    return dna
-------------------------
def translation(dna,d2a):
    pro = ''
    for i in range(0,100,3):
```

Code

The University of Georgia

```
def translation(dna,d2a):
    pro = ''
    for i in range(0,100,3):
        if d2a[dna[i:i+3]] == 'stop':
            print(i, dna[i:i + 3], d2a[dna[i:i + 3]])
            break
        else:
            pro+=d2a[dna[i:i+3]]
            print(i,dna[i:i+3],d2a[dna[i:i+3]])
            input()
    print(pro)
    return pro
```

Code

```
def main():
dna2aa = {
        "TTT": "F", "TTC": "F", "TTA": "L", "TTG": "L",
        "TCT": "S", "TCC": "S", "TCA": "S", "TCG": "S",
        "TAT": "Y", "TAC": "Y", "TAA": "stop", "TAG": "stop",
        "TGT": "C", "TGC": "C", "TGA": "stop", "TGG": "W",
        "CTT": "L", "CTC": "L", "CTA": "L", "CTG": "L",
        "CCT": "P", "CCC": "P", "CCA": "P", "CCG": "P",
        "CAT": "H", "CAC": "H", "CAA": "Q", "CAG": "Q",
        "CGT": "R", "CGC": "R", "CGA": "R", "CGG": "R",
        "ATT": "I", "ATC": "I", "ATA": "I", "ATG": "M",
        "ACT": "T", "ACC": "T", "ACA": "T", "ACG": "T",
        "AAT": "N", "AAC": "N", "AAA": "K", "AAG": "K",
        "AGT": "S", "AGC": "S", "AGA": "R", "AGG": "R",
        "GTT": "V", "GTC": "V", "GTA": "V", "GTG": "V",
        "GCT": "A", "GCC": "A", "GCA": "A", "GCG": "A",
        "GAT": "D", "GAC": "D", "GAA": "E", "GAG": "E",
        "GGT": "G", "GGC": "G", "GGA": "G", "GGG": "G"}
```

Code

```
-------------------------------------
gene = DNA_generate(100)
    print(gene)
    print("gene: %s\nprotein:%s"
%(gene,translation(gene,dna2aa)))
main()
```

Random Number

The University of Georgia

- Import random
  - Random.random()
  - Random.randint(a,b)
  - Random.uniform(a,b)
  - Random.range(a,b,c)
  - Random.choice(‘abc’)
  - Random.choice([‘a’,’b’,’c’])
  - Random.sample(‘abc’,n)

Smith-Waterman

The University of Georgia

Initialize the scoring matrix

		T	G	T	T	A	C	G	G
	0	0	0	0	0	0	0	0	0
G	0								
G	0								
T	0								
T	0								
G	0								
A	0								
C	0								
T	0								
A	0								

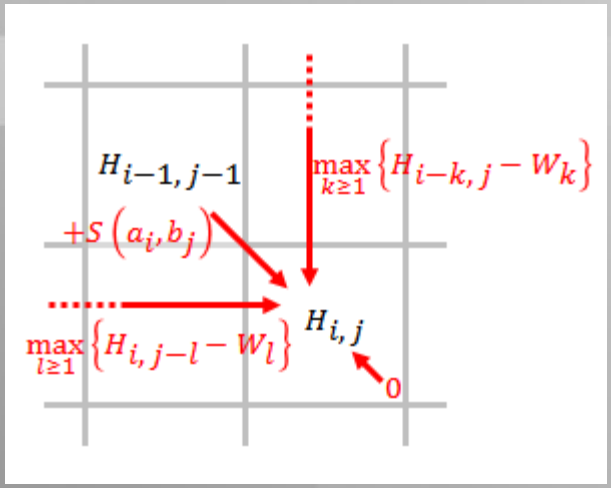
Substitution matrix: $S(a_i, b_j) = \begin{cases} +3, & a_i = b_j \\ -3, & a_i \neq b_j \end{cases}$

Gap penalty: $W_k = kW_1$
$W_1 = 2$

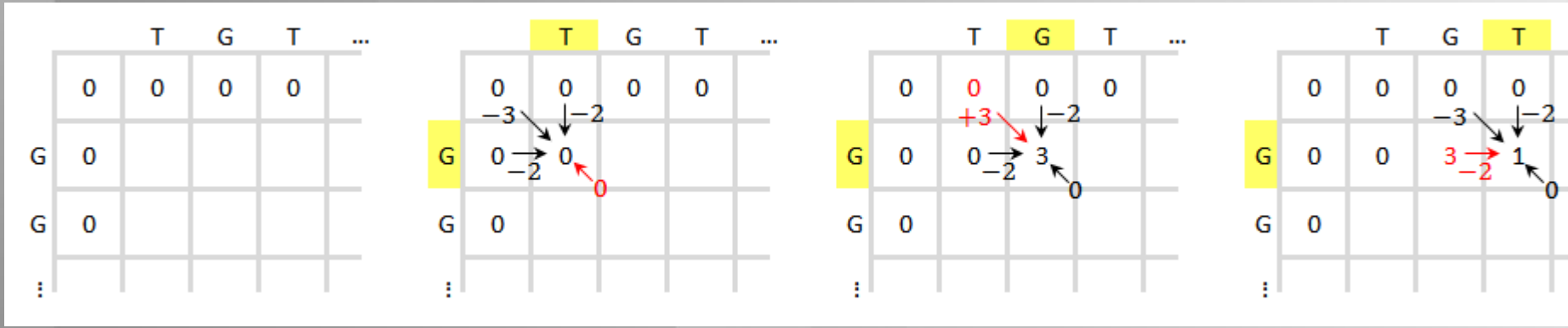
- Local alignment

	Δ	C	A	G	C	C	U	C	G	C	U	U	A	G
Δ	0·0	0·0	0·0	0·0	0·0	0·0	0·0	0·0	0·0	0·0	0·0	0·0	0·0	0·0
A	0·0	0·0	1·0	0·0	0·0	0·0	0·0	0·0	0·0	0·0	0·0	0·0	1·0	0·0
A	0·0	0·0	1·0	0·7	0·0	0·0	0·0	0·0	0·0	0·0	0·0	0·0	1·0	0·7
U	0·0	0·0	0·0	0·7	0·3	0·0	1·0	0·0	0·0	0·0	1·0	1·0	0·0	0·7
G	0·0	0·0	0·0	1·0	0·3	0·0	0·0	0·7	1·0	0·0	0·0	0·7	0·7	1·0
C	0·0	1·0	0·0	0·0	2·0	1·3	0·3	1·0	0·3	2·0	0·7	0·3	0·3	0·3
C	0·0	1·0	0·7	0·0	1·0	3·0	1·7	1·3	1·0	1·3	1·7	0·3	0·0	0·0
A	0·0	0·0	2·0	0·7	0·3	1·7	2·7	1·3	1·0	0·7	1·0	1·3	1·3	0·0
U	0·0	0·0	0·7	1·7	0·3	1·3	2·7	2·3	1·0	0·7	1·7	2·0	1·0	1·0
U	0·0	0·0	0·3	0·3	1·3	1·0	2·3	2·3	2·0	0·7	1·7	2·7	1·7	1·0
G	0·0	0·0	0·0	1·3	0·0	1·0	1·0	2·0	3·3	2·0	1·7	1·3	2·3	2·7
A	0·0	0·0	1·0	0·0	1·0	0·3	0·7	0·7	2·0	3·0	1·7	1·3	2·3	2·0
C	0·0	1·0	0·0	0·7	1·0	2·0	0·7	1·7	1·7	3·0	2·7	1·3	1·0	2·0
G	0·0	0·0	0·7	1·0	0·3	0·7	1·7	0·3	2·7	1·7	2·7	2·3	1·0	2·0
G	0·0	0·0	0·0	1·7	0·7	0·3	0·3	1·3	1·3	2·3	1·3	2·3	2·0	2·0

Smith-Waterman



- Mismatch: -3
- Gap : -2
- Match: +3



Smith-Waterman

The University of Georgia

		T	G	T	T	A	C	G	G
	0	0	0	0	0	0	0	0	0
G	0	0	3	1	0	0	0	3	3
G	0	0	3	1	0	0	0	3	6
T	0	3	1	6	4	2	0	1	4
T	0	3	1	4	9	7	5	3	2
G	0	1	6	4	7	6	4	8	6
A	0	0	4	3	5	10	8	6	5
C	0	0	2	1	3	8	13	11	9
T	0	3	1	5	4	6	11	10	8
A	0	1	0	3	2	7	9	8	7

		T	G	T	T	A	C	G	G
	0	0	0	0	0	0	0	0	0
G	0	0	3	1	0	0	0	3	3
G	0	0	3	1	0	0	0	3	6
T	0	3	1	6	4	2	0	1	4
T	0	3	1	4	9	7	5	3	2
G	0	1	6	4	7	6	4	8	6
A	0	0	4	3	5	10	8	6	5
C	0	0	2	1	3	8	13	11	9
T	0	3	1	5	4	6	11	10	8
A	0	1	0	3	2	7	9	8	7

3	6	9	7	10	13
G	T	T	-	A	C
G	T	T	G	A	C

Statistics Model

The University of Georgia

- Is the alignment result significant or produced by chance
- Randomize sequences
- Alignment score
- Normal distribution
- Z (or t) score and p-value

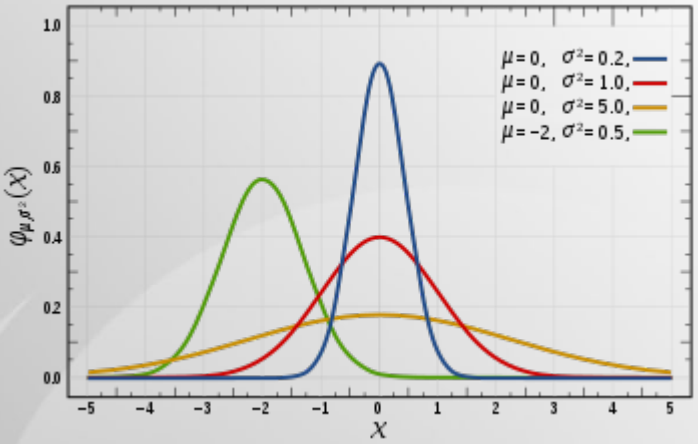
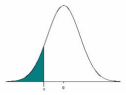


Table of Standard Normal Probabilities for Negative Z-scores



z	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
-3.4	0.0003	0.0003	0.0003	0.0003	0.0003	0.0003	0.0003	0.0003	0.0003	0.0002
-3.3	0.0005	0.0005	0.0005	0.0004	0.0004	0.0004	0.0004	0.0004	0.0004	0.0003
-3.2	0.0007	0.0007	0.0006	0.0006	0.0006	0.0006	0.0006	0.0005	0.0005	0.0005
-3.1	0.0010	0.0009	0.0009	0.0009	0.0008	0.0008	0.0008	0.0008	0.0007	0.0007
-3.0	0.0013	0.0013	0.0013	0.0012	0.0012	0.0011	0.0011	0.0011	0.0010	0.0010
-2.9	0.0019	0.0018	0.0018	0.0017	0.0016	0.0016	0.0015	0.0015	0.0014	0.0014
-2.8	0.0026	0.0025	0.0024	0.0023	0.0023	0.0022	0.0021	0.0021	0.0020	0.0019
-2.7	0.0035	0.0034	0.0033	0.0032	0.0031	0.0030	0.0029	0.0028	0.0027	0.0026
-2.6	0.0047	0.0045	0.0044	0.0043	0.0041	0.0040	0.0039	0.0038	0.0037	0.0036
-2.5	0.0062	0.0060	0.0059	0.0057	0.0055	0.0054	0.0052	0.0051	0.0049	0.0048
-2.4	0.0082	0.0080	0.0078	0.0075	0.0073	0.0071	0.0069	0.0068	0.0066	0.0064
-2.3	0.0107	0.0104	0.0102	0.0099	0.0096	0.0094	0.0091	0.0089	0.0087	0.0084
-2.2	0.0139	0.0136	0.0132	0.0129	0.0125	0.0122	0.0119	0.0116	0.0113	0.0110
-2.1	0.0179	0.0174	0.0170	0.0166	0.0162	0.0158	0.0154	0.0150	0.0146	0.0143
-2.0	0.0228	0.0222	0.0217	0.0212	0.0207	0.0202	0.0197	0.0192	0.0188	0.0183
-1.9	0.0287	0.0281	0.0274	0.0268	0.0262	0.0256	0.0250	0.0244	0.0239	0.0233
-1.8	0.0359	0.0351	0.0344	0.0336	0.0329	0.0322	0.0314	0.0307	0.0301	0.0294
-1.7	0.0446	0.0436	0.0427	0.0418	0.0409	0.0401	0.0392	0.0384	0.0375	0.0367
-1.6	0.0548	0.0537	0.0526	0.0516	0.0505	0.0495	0.0485	0.0475	0.0465	0.0455
-1.5	0.0668	0.0655	0.0643	0.0630	0.0618	0.0606	0.0594	0.0582	0.0571	0.0559
-1.4	0.0808	0.0793	0.0778	0.0764	0.0749	0.0735	0.0721	0.0708	0.0694	0.0681
-1.3	0.0968	0.0951	0.0934	0.0918	0.0901	0.0885	0.0869	0.0853	0.0838	0.0823
-1.2	0.1151	0.1131	0.1112	0.1093	0.1075	0.1056	0.1038	0.1020	0.1003	0.0985
-1.1	0.1357	0.1335	0.1314	0.1292	0.1271	0.1251	0.1230	0.1210	0.1190	0.1170
-1.0	0.1587	0.1562	0.1539	0.1515	0.1492	0.1469	0.1446	0.1423	0.1401	0.1379
-0.9	0.1841	0.1814	0.1788	0.1762	0.1736	0.1711	0.1685	0.1660	0.1635	0.1611
-0.8	0.2119	0.2090	0.2061	0.2033	0.2005	0.1977	0.1949	0.1922	0.1894	0.1867
-0.7	0.2420	0.2389	0.2358	0.2327	0.2296	0.2266	0.2236	0.2206	0.2177	0.2148
-0.6	0.2743	0.2709	0.2676	0.2643	0.2611	0.2578	0.2546	0.2514	0.2483	0.2451
-0.5	0.3085	0.3050	0.3015	0.2981	0.2946	0.2912	0.2877	0.2843	0.2810	0.2776
-0.4	0.3446	0.3409	0.3372	0.3336	0.3300	0.3264	0.3228	0.3192	0.3156	0.3121
-0.3	0.3821	0.3783	0.3745	0.3707	0.3669	0.3632	0.3594	0.3557	0.3520	0.3483
-0.2	0.4207	0.4168	0.4129	0.4090	0.4052	0.4013	0.3974	0.3936	0.3897	0.3859
-0.1	0.4602	0.4562	0.4522	0.4483	0.4443	0.4404	0.4364	0.4325	0.4286	0.4247
-0.0	0.5000	0.4960	0.4920	0.4880	0.4840	0.4801	0.4761	0.4721	0.4681	0.4641

Table of Standard Normal Probabilities for Positive Z-scores

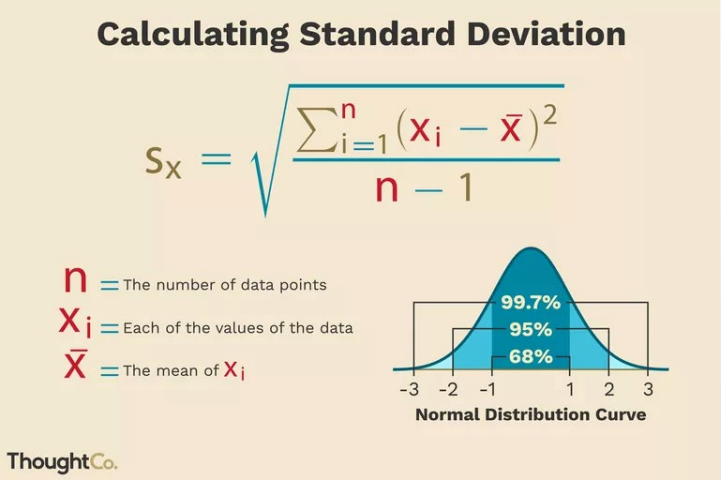
z	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.0	0.5000	0.5040	0.5080	0.5120	0.5160	0.5199	0.5239	0.5279	0.5319	0.5359
0.1	0.5398	0.5438	0.5478	0.5517	0.5557	0.5596	0.5636	0.5675	0.5714	0.5753
0.2	0.5793	0.5832	0.5871	0.5910	0.5948	0.5987	0.6026	0.6064	0.6103	0.6141
0.3	0.6179	0.6217	0.6255	0.6293	0.6331	0.6368	0.6406	0.6443	0.6480	0.6517
0.4	0.6554	0.6591	0.6628	0.6664	0.6700	0.6736	0.6772	0.6808	0.6844	0.6879
0.5	0.6915	0.6950	0.6985	0.7019	0.7054	0.7088	0.7123	0.7157	0.7190	0.7224
0.6	0.7257	0.7291	0.7324	0.7357	0.7389	0.7422	0.7454	0.7486	0.7517	0.7549
0.7	0.7580	0.7611	0.7642	0.7673	0.7704	0.7734	0.7764	0.7794	0.7823	0.7852
0.8	0.7881	0.7910	0.7939	0.7967	0.7995	0.8023	0.8051	0.8078	0.8106	0.8133
0.9	0.8159	0.8186	0.8212	0.8238	0.8264	0.8289	0.8315	0.8340	0.8365	0.8389
1.0	0.8413	0.8438	0.8461	0.8485	0.8508	0.8531	0.8554	0.8577	0.8599	0.8621
1.1	0.8643	0.8665	0.8686	0.8708	0.8729	0.8749	0.8770	0.8790	0.8810	0.8830
1.2	0.8849	0.8869	0.8888	0.8907	0.8925	0.8944	0.8962	0.8980	0.8997	0.9015
1.3	0.9032	0.9049	0.9066	0.9082	0.9099	0.9115	0.9131	0.9147	0.9162	0.9177
1.4	0.9192	0.9207	0.9222	0.9236	0.9251	0.9265	0.9279	0.9292	0.9306	0.9319
1.5	0.9332	0.9345	0.9357	0.9370	0.9382	0.9394	0.9406	0.9418	0.9429	0.9441
1.6	0.9452	0.9463	0.9474	0.9484	0.9495	0.9505	0.9515	0.9525	0.9535	0.9545
1.7	0.9554	0.9564	0.9573	0.9582	0.9591	0.9599	0.9608	0.9616	0.9625	0.9633
1.8	0.9641	0.9649	0.9656	0.9664	0.9671	0.9678	0.9686	0.9693	0.9699	0.9706
1.9	0.9713	0.9719	0.9726	0.9732	0.9738	0.9744	0.9750	0.9756	0.9761	0.9767
2.0	0.9772	0.9778	0.9783	0.9788	0.9793	0.9798	0.9803	0.9808	0.9812	0.9817
2.1	0.9821	0.9826	0.9830	0.9834	0.9838	0.9842	0.9846	0.9850	0.9854	0.9857
2.2	0.9861	0.9864	0.9868	0.9871	0.9875	0.9878	0.9881	0.9884	0.9887	0.9890
2.3	0.9893	0.9896	0.9898	0.9901	0.9904	0.9906	0.9909	0.9911	0.9913	0.9916
2.4	0.9918	0.9920	0.9922	0.9925	0.9927	0.9929	0.9931	0.9932	0.9934	0.9936
2.5	0.9938	0.9940	0.9941	0.9943	0.9945	0.9946	0.9948	0.9949	0.9951	0.9952
2.6	0.9953	0.9955	0.9956	0.9957	0.9959	0.9960	0.9961	0.9962	0.9963	0.9964
2.7	0.9965	0.9966	0.9967	0.9968	0.9969	0.9970	0.9971	0.9972	0.9973	0.9974
2.8	0.9974	0.9975	0.9976	0.9977	0.9977	0.9978	0.9979	0.9979	0.9980	0.9981
2.9	0.9981	0.9982	0.9982	0.9983	0.9984	0.9984	0.9985	0.9985	0.9986	0.9986
3.0	0.9987	0.9987	0.9987	0.9988	0.9988	0.9989	0.9989	0.9989	0.9990	0.9990
3.1	0.9990	0.9991	0.9991	0.9991	0.9992	0.9992	0.9992	0.9992	0.9993	0.9993
3.2	0.9993	0.9993	0.9994	0.9994	0.9994	0.9994	0.9994	0.9995	0.9995	0.9995
3.3	0.9995	0.9995	0.9995	0.9996	0.9996	0.9996	0.9996	0.9996	0.9996	0.9997
3.4	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9998

Note that the probabilities given in this table represent the area to the LEFT of the z-score.
The area to the RIGHT of a z-score = 1 – the area to the LEFT of the z-score

Statistically Significant

The University of Georgia

- How to determine if the alignment is significant?
- Random alignment
- Normal distribution
- Average and standard deviation
- T test



Sample Variance

$$s^2 = \frac{\sum x^2 - \frac{(\sum x)^2}{n}}{n-1}$$

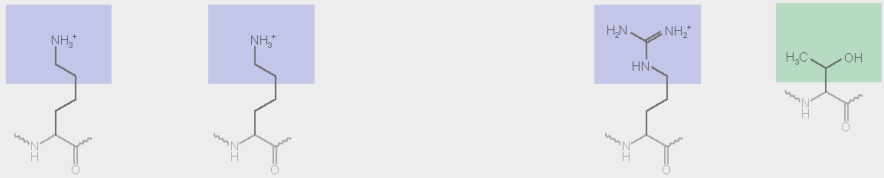
Sample Standard Deviation

$$s = \sqrt{\frac{\sum x^2 - \frac{(\sum x)^2}{n}}{n-1}}$$

$$t = \frac{\overline{X} - \mu_{\overline{X}}}{\sigma_{\overline{X}}}$$

PAM

	No mutation	Point mutations			
		Silent	Nonsense	Missense	
				conservative	non-conservative
DNA level	TTC	TTT	ATC	TCC	TGC
mRNA level	AAG	AAA	UAG	AGG	ACG
protein level	**Lys**	**Lys**	**STOP**	**Arg**	**Thr**



- Point Accepted Mutation
- PAM1 matrix
- PAM2 = PAM1*PAM1
- PAM250 matrix

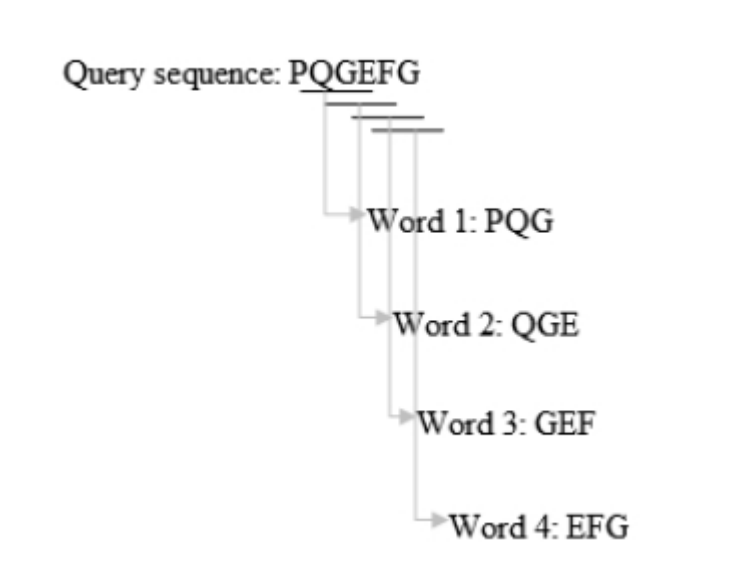
BLOSUM

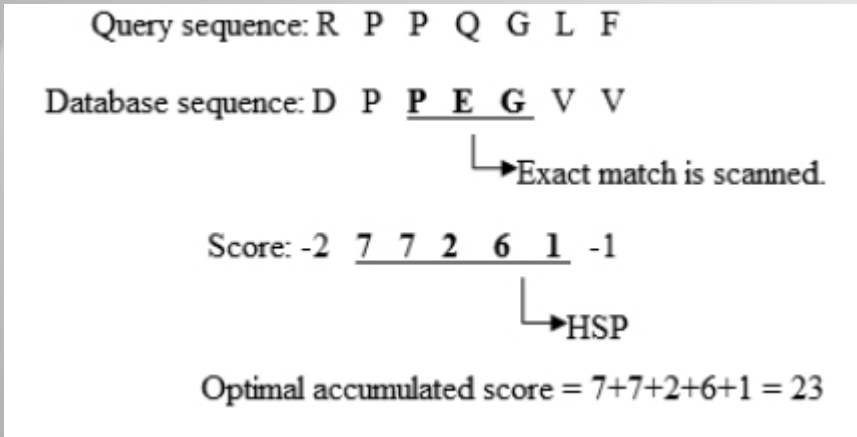
	Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val
Ala	4																			
Arg	-1	5																		
Asn	-2	0	6																	
Asp	-2	-2	1	6																
Cys	0	-3	-3	-3	9															
Gln	-1	1	0	0	-3	5														
Glu	-1	0	0	2	-4	2	5													
Gly	0	-2	0	-1	-3	-2	-2	6												
His	-2	0	1	-1	-3	0	0	-2	8											
Ile	-1	-3	-3	-3	-1	-3	-3	-4	-3	4										
Leu	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4									
Lys	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5								
Met	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5							
Phe	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6						
Pro	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7					
Ser	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4				
Thr	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5			
Trp	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11		
Tyr	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	
Val	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4

- BLOcks Substituion Matrix

BLAST

- **B**asic **L**ocal **A**lignment **S**earch **T**ool
- Library or Database searching
- Most widely used (published in 1990)
- High-scoring Segment Pairs (HSP)





BLAST

The University of Georgia

- Nucleotide-nucleotide BLAST (blastn)
  - query: nucleotide search: nucleotide
- Protein-protein BLAST (blastp)
  - query: protein search: protein
- Position-Specific Iterative BLAST (PSI-BLAST) (blastpgp)
- Nucleotide 6-frame translation-protein (blastx)
  - Query: 6-frame of nucleotide sequence search: protein
- Nucleotide 6-frame translation-nucleotide 6-frame translation (tblastx)
  - Query: nucleotide search: nucleotide
- Protein-nucleotide 6-frame translation (tblastn)
  - Query: protein search: nucleotide
- Large numbers of query sequences (megablast)

Summary

- Needleman-Wunch algorithm
- Smith-Waterman algorithm
- Statistical model
- Mutation matrix
- Modern sequence alignment tool: BLAST
- Challenges:
  - Distance-related
  - Multiple alignment


Thank
You


TIME
FOR A
BREAK