# Algorithmic Approaches for Biological Data, Lecture #7

Katherine St. John

City University of New York American Museum of Natural History

10 February 2016

### Outline



#### Patterns in Strings

- Recap: Files
- in and not in
- String methods: find() and count()
- Regular Expressions

• Opening a file:



• Opening a file: infile = open('data.txt', 'r')



• Opening a file: infile = open('data.txt', 'r') outfile = open('log.txt', 'w')





- Opening a file: infile = open('data.txt', 'r') outfile = open('log.txt', 'w')
- Reading from a file:



- Opening a file: infile = open('data.txt', 'r') outfile = open('log.txt', 'w')
- Reading from a file:
  - infile.read(): reads the entire file into a single string.



- Opening a file: infile = open('data.txt', 'r') outfile = open('log.txt', 'w')
- Reading from a file:
  - infile.read(): reads the entire file into a single string.
  - infile.readline(): read the next line of the file.



- Opening a file: infile = open('data.txt', 'r') outfile = open('log.txt', 'w')
- Reading from a file:
  - infile.read(): reads the entire file into a single string.
  - infile.readline(): read the next line of the file.
  - infile.readlines(): read the file into a list of strings.



- Opening a file: infile = open('data.txt', 'r') outfile = open('log.txt', 'w')
- Reading from a file:
  - infile.read(): reads the entire file into a single string.
  - infile.readline(): read the next line of the file.
  - infile.readlines(): read the file into a list of strings.
- Closing a file:



- Opening a file: infile = open('data.txt', 'r') outfile = open('log.txt', 'w')
- Reading from a file:
  - infile.read(): reads the entire file into a single string.
  - infile.readline(): read the next line of the file.
  - infile.readlines(): read the file into a list of strings.
- Closing a file: infile.close()
- Writing to a file:



- Opening a file: infile = open('data.txt', 'r') outfile = open('log.txt', 'w')
- Reading from a file:
  - infile.read(): reads the entire file into a single string.
  - infile.readline(): read the next line of the file.
  - infile.readlines(): read the file into a list of strings.
- Closing a file: infile.close()
- Writing to a file: outfile.write(s)



- Opening a file: infile = open('data.txt', 'r') outfile = open('log.txt', 'w')
- Reading from a file:
  - infile.read(): reads the entire file into a single string.
  - infile.readline(): read the next line of the file.
  - infile.readlines(): read the file into a list of strings.
- Closing a file: infile.close()
- Writing to a file: outfile.write(s)

• Write a program will double space a file.

Write a program will double space a file.
 f = open('first.txt', 'r')

```
f = open('first.txt', 'r')
data = f.read().replace('\n','\n\n')
print data
f.close()
```

• Write a program will double space a file. f = open('first.txt', 'r') data = f.read().replace('\n','\n\n') print data f.close()

 Write a program that asks the user for a input and output file. Your program should copy the contents of the input file to the output and number the lines.

• Write a program will double space a file.

```
f = open('first.txt', 'r')
data = f.read().replace('\n','\n\n')
print data
f.close()
```

 Write a program that asks the user for a input and output file. Your program should copy the contents of the input file to the output and number the lines.

```
inName = raw.input('Please enter input file:')
outName = raw.input('Please enter output file:')
infile = open(inName, 'r')
outfile = open(outName, 'w')
lines = infile.readlines()
for i in len(lines):
    print i+':\t'+lines[i],
infile.close()
outfile.close()
```

• Write a program will double space a file.

```
f = open('first.txt', 'r')
data = f.read().replace('\n','\n\n')
print data
f.close()
```

 Write a program that asks the user for a input and output file. Your program should copy the contents of the input file to the output and number the lines.

```
inName = raw.input('Please enter input file:')
outName = raw.input('Please enter output file:')
infile = open(inName, 'r')
outfile = open(outName, 'w')
lines = infile.readlines()
for i in len(lines):
    print i+':\t'+lines[i],
infile.close()
outfile.close()
```

 Write a program that takes as input a FASTA file and prints out the number of sequences in the file.

Write a program will double space a file.

```
f = open('first.txt', 'r')
data = f.read().replace('\n','\n\n')
print data
f.close()
```

 Write a program that asks the user for a input and output file. Your program should copy the contents of the input file to the output and number the lines.

```
inName = raw.input('Please enter input file:')
outName = raw.input('Please enter output file:')
infile = open(inName, 'r')
outfile = open(outName, 'w')
lines = infile.readlines()
for i in len(lines):
    print i+':\t'+lines[i],
infile.close()
outfile.close()
```

 Write a program that takes as input a FASTA file and prints out the number of sequences in the file.

```
f = open('fasta.txt', 'r')
count = f.read().count('>')
print count
f.close()
```



• Another useful string method: in



- Another useful string method: in
- Tests if a string is part of another.



- Another useful string method: in
- Tests if a string is part of another.
- e.g. if 'd' in 'teddy':



- Another useful string method: in
- Tests if a string is part of another.
- e.g. if 'd' in 'teddy':
- Also not in



- Another useful string method: in
- Tests if a string is part of another.
- e.g. if 'd' in 'teddy':
- Also not in
- e.g. if 'a' not in 'roosevelt':



- Another useful string method: in
- Tests if a string is part of another.
- e.g. if 'd' in 'teddy':
- Also not in
- e.g. if 'a' not in 'roosevelt':
- Text book demo



• Very useful string methods.



- Very useful string methods.
- Can find and count exact matches.



- Very useful string methods.
- Can find and count exact matches.
- e.g. dna.find('TA') or strand.count('AUG').



- Very useful string methods.
- Can find and count exact matches.
- e.g. dna.find('TA') or strand.count('AUG').
- Harder to use for approximate matches or patterns with varying lengths.



- Very useful string methods.
- Can find and count exact matches.
- e.g. dna.find('TA') or strand.count('AUG').
- Harder to use for approximate matches or patterns with varying lengths.
- e.g. many motifs or any number of TA: TA, TATA, TATATA



Short-hand for writing down patterns:

• GET, NET, and SET all end in -ET.



Short-hand for writing down patterns:

- GET, NET, and SET all end in -ET.
  - ► The pattern is [GNS]ET



#### Short-hand for writing down patterns:

- GET, NET, and SET all end in -ET.
  - ► The pattern is [GNS]ET
  - ▶ [GNS] matches one letter of G, N, or S.
  - ▶ The match must end in ET.



#### Short-hand for writing down patterns:

- GET, NET, and SET all end in -ET.
  - ► The pattern is [GNS]ET
  - ▶ [GNS] matches one letter of G, N, or S.
  - ▶ The match must end in ET.
  - Example of a regular expression.

### More Patterns

#### Some ways to write patterns:

• [a-z] matches all lower case letters.



#### More Patterns

#### Some ways to write patterns:

- [a-z] matches all lower case letters.
- [0-1] [0-9] matches all two digit numbers 00 to 19.

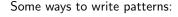


- [a-z] matches all lower case letters.
- [0-1] [0-9] matches all two digit numbers 00 to 19. Alternatively [0-1]\d



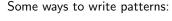
- [a-z] matches all lower case letters.
- [0-1] [0-9] matches all two digit numbers 00 to 19. Alternatively [0-1]\d
- [0-9A-Fa-f] matches any hexadecimal digit.





- [a-z] matches all lower case letters.
- [0-1] [0-9] matches all two digit numbers 00 to 19. Alternatively [0-1] \d
- [0-9A-Fa-f] matches any hexadecimal digit.
- matches any character





- [a-z] matches all lower case letters.
- [0-1] [0-9] matches all two digit numbers 00 to 19. Alternatively [0-1] \d
- [0-9A-Fa-f] matches any hexadecimal digit.
- matches any character
- ah+ matches ah, ahh, ahhh, ahhhh, ...





- [a-z] matches all lower case letters.
- [0-1] [0-9] matches all two digit numbers 00 to 19.
   Alternatively [0-1] \d
- [0-9A-Fa-f] matches any hexadecimal digit.
- matches any character
- ah+ matches ah, ahh, ahhh, ahhhh, ...
- ah\* matches a, ah, ahh, ahhh, ahhhh, ...



- [a-z] matches all lower case letters.
- [0−1] [0−9] matches all two digit numbers 00 to 19.
   Alternatively [0−1]\d
- [0-9A-Fa-f] matches any hexadecimal digit.
- matches any character
- ah+ matches ah, ahh, ahhh, ahhhh, ...
- ah\* matches a, ah, ahh, ahhh, ahhhh, ...
- (Ha)+ matches Ha, HaHa, HaHaHa,...



- [a-z] matches all lower case letters.
- [0-1] [0-9] matches all two digit numbers 00 to 19. Alternatively [0-1]\d
- [0-9A-Fa-f] matches any hexadecimal digit.
- matches any character
- ah+ matches ah, ahh, ahhh, ahhhh, ...
- ah\* matches a, ah, ahh, ahhh, ahhhh, ...
- (Ha)+ matches Ha, HaHa, HaHaHa,...
- (Ha) {3,5} matches HaHaHa, HaHaHaHa,
   HaHaHaHaHa.

## Biomolecular Sequence Challenges

- 1. ATG CAA TGG GGA AAT GTT ACC AGG TCC GAA CTT ATT GAG GTA AGA CAG ATT EAR
  2. A TGC AAT GGG GAA ATG TTA CCA GGT CCG AAC TTA TTG AGG EAR GAC AGA TTT AA
- 3. AT GCA ATG GGG AAA TGT TAC CAG GTC CGA ACT TAT TGA GGT AAG ACA GAT TTA A

(wiki)

### Many interesting patterns in biomolecular sequences

Codons (3 letter sequences) correspond to amino acids.

# Biomolecular Sequence Challenges

- 1. ATG CAA TGG GGA AAT GTT ACC AGG TCC GAA CTT ATT GAG GTA AGA CAG ATT EAR
  2. A TGC AAT GGG GAA ATG TTA CCA GGT CCG AAC TTA TTG AGG EAR GAC AGA TTT AA
- 3. AT GCA ATG GGG AAA TGT TAC CAG GTC CGA ACT TAT TGA GGT AAG ACA GAT TTA A

(wiki)

### Many interesting patterns in biomolecular sequences

- Codons (3 letter sequences) correspond to amino acids.
- Open reading frames (ORF) can potentially code for a protein:

```
\begin{array}{ccc} \textbf{start codon} & \mathsf{codon, codon, \dots, codon, codon} & \textbf{stop codon} \\ (\mathsf{ATG}) & & (\mathsf{TAA, TAG, or TGA}) \end{array}
```

# Group Work

In pairs/triples: Fill in the table:

# Group Work

In pairs/triples: Fill in the table:

Regular Expression	Description of Matching Strings
[ACGT]*	A DNA sequence— any string consisting only of A,
	C, G, and T
	A RNA sequence— any string consisting only of A,
	C, G, and U
[AT]+	
ATG[ATGC]{30,1000}A{5,10}	
	A DNA sequence that exactly breaks into codons
	(3-letter sequences).
	An open reading frame (ORF): a sequence that
	starts with the start codon ATG, followed by any
	number of codons, and ending with a stop codon
	(TAA, TAG, or TGA).

# Recap



Grand Prismatic Spring (amnh)

• Lab at 3pm.

# Recap



Grand Prismatic Spring (amnh)

- Lab at 3pm.
- Email lab reports to kstjohn@amnh.org

## Recap



Grand Prismatic Spring (amnh)

- Lab at 3pm.
- Email lab reports to kstjohn@amnh.org
- Challenges available at rosalind.info