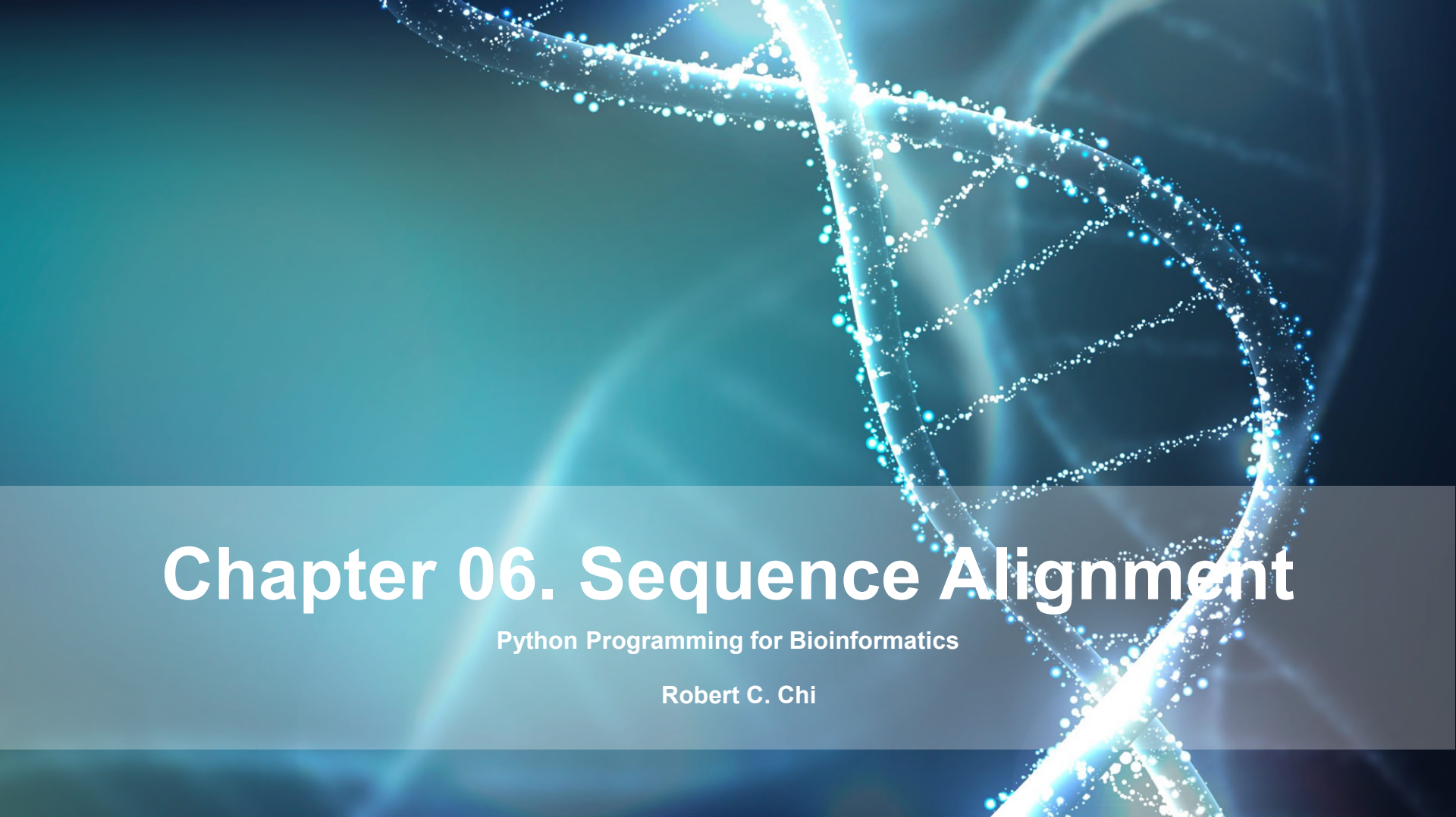


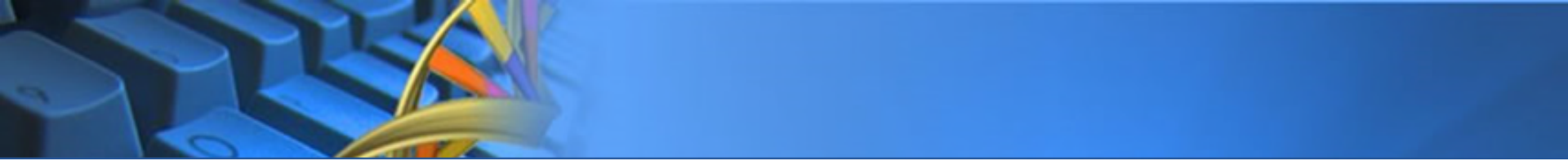
Chapter 06. Sequence Alignment

Python Programming for Bioinformatics

Robert C. Chi

Agenda

- Introduction
- Data Structure for Alignments
- Parse Alignment Files
- Pairwise Alignment
- BLAST
- Write into Alignment Files
- Summary



INTRODUCTION

What is “Sequence Alignment”?

- **Identify** regions of **similarity** against **DNA**, **RNA**, or **protein** sequences

Types of Sequence Alignment

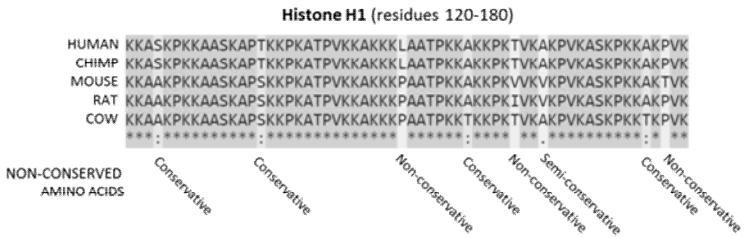
- **Pairwise Alignment**
- **Multiple Alignment**

Local Alignment

```
Target Sequence
5' ACTACTAGATTACTTACGGATCAGGTACTTTAGAGGCTTGCAACCA 3'
             |||| |||||| ||||||||||||||||
Query Sequence 5' TACTCACGGATGAGGTACTTTAGAGGC 3'
```

Global Alignment

```
Target Sequence
5' ACTACTAGATTACTTACGGATCAGGTACTTTAGAGGCTTGCAACCA 3'
   ||||||||||||    ||||||||  |||||||||||||||| |||||||
5' ACTACTAGATT----ACGGATC--GTACTTTAGAGGCTAGCAACCA 3'
Query Sequence
```



Align 2 Sequences

Align 3+ Sequences a Time

Commonly Used Algorithms

- **Smith–Waterman Algorithm**
  - A local alignment method.
  - Based on dynamic programming by given start and end points.
  - Precise but slow.
- **FASTA (@EMBL)**
  - Use a subset of sequence (known as a “word”) with length of k to query.
  - Align "words" to the sequences in databases and find the local similarity among sequences.
  - Faster than Smith–Waterman Algorithm, but slower than BLAST.
  - Prefer for querying shorter sequences.
- **BLAST (@NCBI)**
  - Align "words" with length of k to to the sequences in databases for local similarity.
  - A faster alternative to FASTA without sacrificing much accuracy.
  - Prefer for querying sequences among huge databases.

Commonly Used File Formats

- **FASTA**

```
>COATB_BPIKE/30-81
AEPNAATNYATEAMDSLKTQAIDLISQTWPVVTTVVVAGLVIRLFKKFSSKA
>Q9T0Q8_BPIKE/1-52
AEPNAATNYATEAMDSLKTQAIDLISQTWPVVTTVVVAGLVIKLFKKFVSRA
>COATB_BPI22/32-83
DGTSTATSYATEAMNSLKTQATDLIDQTWPVVTSVAVAGLAIRLFKKFSSKA
>COATB_BPM13/24-72
AEGDDP---AKAAFNSLQASATEYIGYAWAMVVVIVGATIGIKLFKKFTSKA
>COATB_BPZJ2/1-49
AEGDDP---AKAAFDSLQASATEYIGYAWAMVVVIVGATIGIKLFKKFASKA
>Q9T0Q9_BPFD/1-49
AEGDDP---AKAAFDSLQASATEYIGYAWAMVVVIVGATIGIKLFKKFTSKA
>COATB_BPIF1/22-73
FAADDATSQAKAAFDSLTAQATEMSGYAWALVVLVVGATVGIKLFKKFVSRA
```

seq 1
seq 2
seq 3
seq 4
seq 5
seq 6
seq 7

Sequences to be Aligned

Commonly Used File Formats

- **Stockholm Format**

Record x1

```
# STOCKHOLM 1.0
#=GF ID CBS
#=GF AC PF00571
#=GF DE CBS domain
#=GF AU Bateman A
#=GF CC CBS domains are small intracellular modules mostly found
#=GF CC in 2 or four copies within a protein.
#=GF SQ 5
#=GS O31698/18-71 AC O31698
#=GS O83071/192-246 AC O83071
#=GS O83071/259-312 AC O83071
#=GS O31698/88-139 AC O31698
#=GS O31698/88-139 OS Bacillus subtilis
O83071/192-246          MTCRAQLIAVPRASSLAEAIACAQKMRVSRVPVYERS
#=GR O83071/192-246 SA  9998877564535242525515252536463774777
O83071/259-312          MQHVSAPVFVFECTRLAYVQHKLRAHSRAVAIVLDEY
#=GR O83071/259-312 SS  CCCCCHHHHHHHHHHHHEEEEEEEEEEEEEEEEEE
O31698/18-71            MIEADKVAHVQVGNNLEHALLVLTKTGYTAIPVLDPS
#=GR O31698/18-71 SS    CCCHHHHHHHHHHHHHHEEEEEEEEEEEEEEEHHH
O31698/88-139           EVMLTDIPRLHINDPIMKGFGMVINN..GFVCVENDE
#=GR O31698/88-139 SS   CCCCCCCHHHHHHHHHHHHEEEEEEEEEEEEEEEEH
#=GC SS_cons            CCCCCHHHHHHHHHHHHHEEEEEEEEEEEEEEEEEH
O31699/88-139           EVMLTDIPRLHINDPIMKGFGMVINN..GFVCVENDE
#=GR O31699/88-139 AS   ________________*_______________
#=GR O31699/88-139 IN   ____________1_____2_____0____
//
```

seq 1
seq 2
seq 3
seq 4
seq 5

Header

- # STOCKHOLM 1.0

Annotation Lines

- #=GF: Generic File annotations
- #=GS: Generic Sequence annotations
- #=GR: Generic Residue annotations
- #=GC: Generic Column annotations

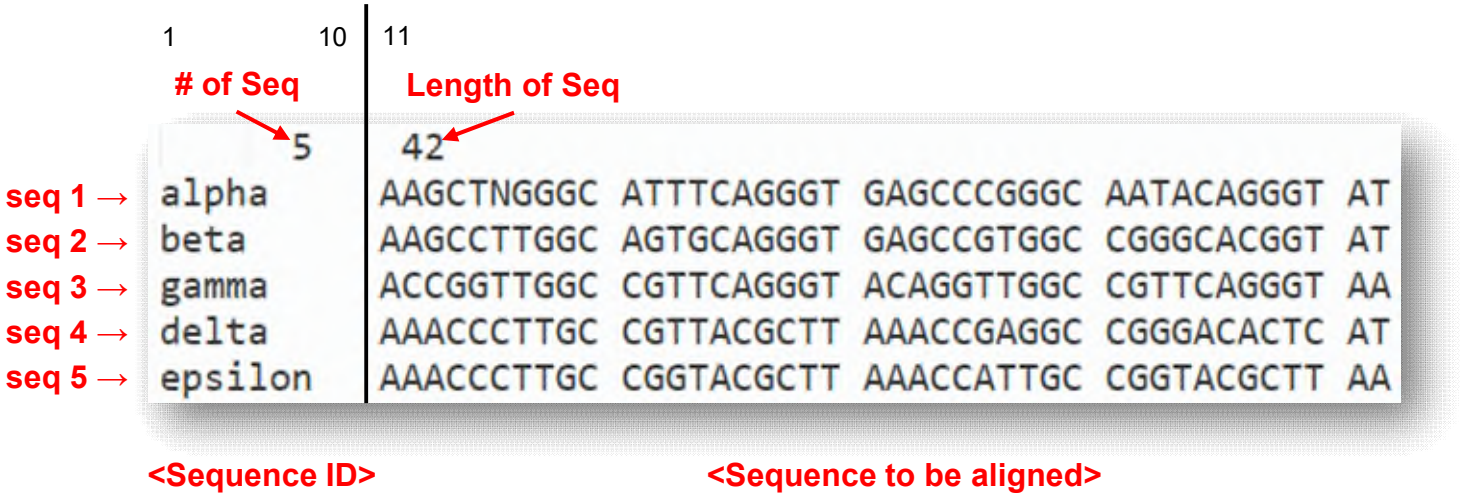
Sequences (to be aligned)

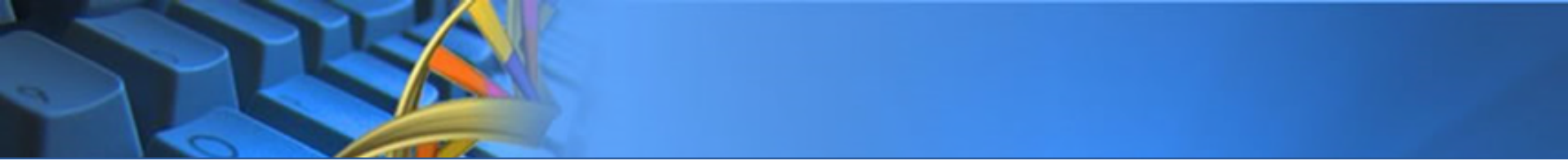
- O83071/192-246 MTCRAQ...ERS

<Sequence ID> <Sequence>

Commonly Used File Formats

- **PHYLIP Format**





Bio.Align.MultipleSeqAlignment

DATA STRUCTURE FOR ALIGNMENTS

Bio.Align.MultipleSeqAlignment

```
1  from Bio.Seq import Seq
2  from Bio.SeqRecord import SeqRecord
3  from Bio.Align import MultipleSeqAlignment
4
5  my_align = MultipleSeqAlignment([
6                 SeqRecord(Seq("ACTGCTAGCTAG"), id="Alpha"),
7                 SeqRecord(Seq("ACT-CTAGCTAG"), id="Beta"),
8                 SeqRecord(Seq("ACTGCTAGDTAG"), id="Gamma"),
9             ])
10
11 print(my_align)
```

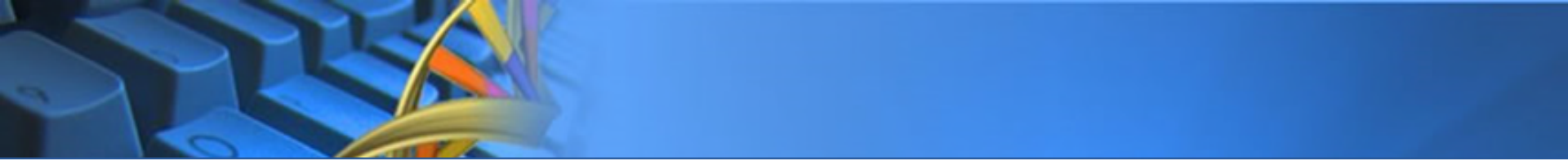
```
Alignment with 3 rows and 12 columns
ACTGCTAGCTAG Alpha
ACT-CTAGCTAG Beta
ACTGCTAGDTAG Gamma
```

Practice

- **Bio.Align.MultipleSeqAlignment**
  - Write and Run the following codes on a Colab page called “Alignment.ipynb”:

```
1  from Bio.Seq import Seq
2  from Bio.SeqRecord import SeqRecord
3  from Bio.Align import MultipleSeqAlignment
4
5  my_align = MultipleSeqAlignment([
6                 SeqRecord(Seq("ACTGCTAGCTAG"), id="Alpha"),
7                 SeqRecord(Seq("ACT-CTAGCTAG"), id="Beta"),
8                 SeqRecord(Seq("ACTGCTAGDTAG"), id="Gamma"),
9             ])
10
11 print(my_align)
```

(Solution URL of this Practice)



FASTA, Stockholm, PHYLIP

PARSE ALIGNMENT FILES

Parse FASTA File

- **FASTA File to be Parsed**

```
>COATB_BPIKE/30-81
AEPNAATNYATEAMDSLKTQAIDLISQTWPVVTTVVVAGLVIRLFKKFSSKA
>Q9T0Q8_BPIKE/1-52
AEPNAATNYATEAMDSLKTQAIDLISQTWPVVTTVVVAGLVIKLFKKFVSRA
>COATB_BPI22/32-83
DGTSTATSYATEAMNSLKTQATDLIDQTWPVVTSVAVAGLAIRLFKKFSSKA
>COATB_BPM13/24-72
AEGDDP---AKAAFNSLQASATEYIGYAWAMVVVIVGATIGIKLFKKFTSKA
>COATB_BPZJ2/1-49
AEGDDP---AKAAFDSLQASATEYIGYAWAMVVVIVGATIGIKLFKKFASKA
>Q9T0Q9_BPFD/1-49
AEGDDP---AKAAFDSLQASATEYIGYAWAMVVVIVGATIGIKLFKKFTSKA
>COATB_BPIF1/22-73
FAADDATSQAKAAFDSLTAQATEMSGYAWALVVLVVGATVGIKLFKKFVSRA
```

seq 1
seq 2
seq 3
seq 4
seq 5
seq 6
seq 7

Sequences to be Aligned

Parse FASTA File

```
1  import os
2  from Bio import AlignIO
3
4  # Download the FASTA File
5  if not os.path.isfile("PF05371_seed.faa"):
6    os.system("wget -c https://bit.ly/3Ex9Txr -O PF05371_seed.faa")
7
8  # Read the first record of FASTA File
9  alignment = AlignIO.read("PF05371_seed.faa", "fasta")
10 print(alignment)
```

```
Alignment with 7 rows and 52 columns
AEPNAATNYATEAMDSLKTQAIDLISQTWPVVTTVVVAGLVIRL...SKA COATB_BPIKE/30-81
AEPNAATNYATEAMDSLKTQAIDLISQTWPVVTTVVVAGLVIKL...SRA Q9T0Q8_BPIKE/1-52
DGTSTATSYATEAMNSLKTQATDLIDQTWPVVTSVAVAGLAIRL...SKA COATB_BPI22/32-83
AEGDDP---AKAAFNSLQASATEYIGYAWAMVVVIVGATIGIKL...SKA COATB_BPM13/24-72
AEGDDP---AKAAFDSLQASATEYIGYAWAMVVVIVGATIGIKL...SKA COATB_BPZJ2/1-49
AEGDDP---AKAAFDSLQASATEYIGYAWAMVVVIVGATIGIKL...SKA Q9T0Q9_BPFD/1-49
FAADDATSQAKAAFDSLTAQATEMSGYAWALVVLVVGATVGIKL...SRA COATB_BPIF1/22-73
```

Practice

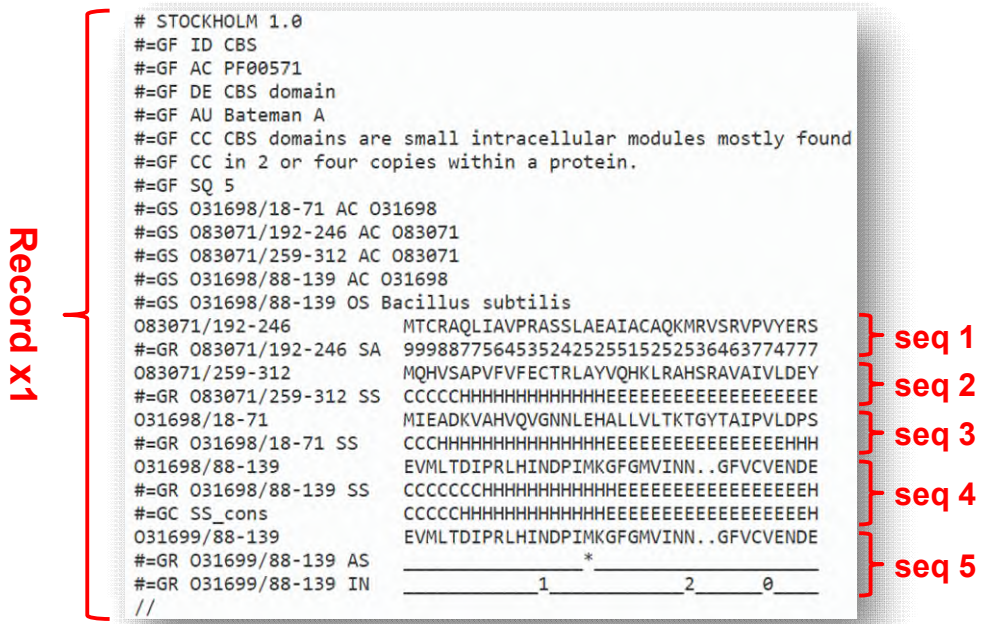
- **Parse FASTA File**
  - Write and Run the following codes on a Colab page called “Alignment.ipynb”:

```
1  import os
2  from Bio import AlignIO
3
4  # Download the FASTA File
5  if not os.path.isfile("PF05371_seed.faa"):
6    os.system("wget -c https://bit.ly/3Ex9Txr -O PF05371_seed.faa")
7
8  # Read the first record of FASTA File
9  alignment = AlignIO.read("PF05371_seed.faa", "fasta")
10 print(alignment)
```

(Solution URL of this Practice)

Parse Stockholm File

- **Stockholm File to be Parsed**



Parse Stockholm File

```python
import os
from Bio import AlignIO

# Download the Stockholm File
if not os.path.isfile("PF05371_seed.sth"):
    os.system("wget -c https://bit.ly/31rbIxb -O PF05371_seed.sth")

# Parse all the records of Stockholm File
alignments = list(AlignIO.parse("PF05371_seed.sth", "stockholm"))

# Iterate each set of sequences to be aligned
for alignment in alignments:
    print(alignment)
```

```
Alignment with 7 rows and 52 columns
AEPNAATNYATEAMDSLKTQAIDLISQTWPVVTTVVVAGLVIRL...SKA COATB_BPIKE/30-81
AEPNAATNYATEAMDSLKTQAIDLISQTWPVVTTVVVAGLVIKL...SRA Q9T0Q8_BPIKE/1-52
DGTSTATSYATEAMNSLKTQATDLIDQTWPVVTSVAVAGLAIRL...SKA COATB_BPI22/32-83
AEGDDP---AKAAFNSLQASATEYIGYAWAMVVVIVGATIGIKL...SKA COATB_BPM13/24-72
AEGDDP---AKAAFDSLQASATEYIGYAWAMVVVIVGATIGIKL...SKA COATB_BPZJ2/1-49
AEGDDP---AKAAFDSLQASATEYIGYAWAMVVVIVGATIGIKL...SKA Q9T0Q9_BPFD/1-49
FAADDATSQAKAAFDSLTAQATEMSGYAWALVVLVVGATVGIKL...SRA COATB_BPIF1/22-73
```

Practice

- **Parse Stockholm File**
  - Write and Run the following codes on a Colab page called “Alignment.ipynb”:

```
1  import os
2  from Bio import AlignIO
3
4  # Download the Stockholm File
5  if not os.path.isfile("PF05371_seed.sth"):
6    os.system("wget -c https://bit.ly/31rbIxb -O PF05371_seed.sth")
7
8  # Parse all the records of Stockholm File
9  alignments = list(AlignIO.parse("PF05371_seed.sth", "stockholm"))
10
11 # Iterate each set of sequences to be aligned
12 for alignment in alignments:
13   print(alignment)
```

(Solution URL of this Practice)

Parse Stockholm File

- **Get more information**

```python
# Get only the first set of sequences to be aligned
alignment = alignments[0]

# Overall Information
print("Total Records:", len(alignment))
print("Sequence Length:", alignment.get_alignment_length())
print()

# Show the attributes of a record in an alignment
for record in alignment:
  print("{} (ID: {})".format(record.name, record.id))
  print("Description:", record.description)
  print("Features:", record.features)
  print("Annotations:", record.annotations)
  print("Sequence:", record.seq)
  print("DSSP:", record.letter_annotations.get("secondary_structure", "N/A"))
  print()
```

```
Total Records: 7
Sequence Length: 52
```

```
COATB_BPIKE (ID: COATB_BPIKE/30-81)
Description: COATB_BPIKE/30-81
Features: []
Annotations: {'accession': 'P03620.1', 'start': 30, 'end': 81}
Sequence: AEPNAATNYATEAMDSLKTQAIDLISQTWPVVTTVVVAGLVIRLFKKFSSKA
DSSP: -HHHHHHHHHHHHH--HHHHHHHH--HHHHHHHHHHHHHHHHHHHHH----
```

[DSSP (Dictionary of Secondary Structure of Proteins)] **Classification Details**

Practice

- **Get More Information from Stockholm File**
  - Write and Run the following codes on a Colab page called “Alignment.ipynb”:

```
# Get only the first set of sequences to be aligned
alignment = alignments[0]

# Overall Information
print("Total Records:", len(alignment))
print("Sequence Length:", alignment.get_alignment_length())
print()

# Show the attributes of a record in an alignment
for record in alignment:
  print("{} (ID: {})".format(record.name, record.id))
  print("Description:", record.description)
  print("Features:", record.features)
  print("Annotations:", record.annotations)
  print("Sequence:", record.seq)
  print("DSSP:", record.letter_annotations.get("secondary_structure", "N/A"))
  print()
```

(Solution URL of this Practice)

Parse PHYLIP File

- **PHYLIP File to be Parsed**

```
     5    42
alpha     AAGCTNGGGC ATTTCAGGGT GAGCCCGGGC AATACAGGGT AT
beta      AAGCCTTGGC AGTGCAGGGT GAGCCGTGGC CGGGCACGGT AT
gamma     ACCGGTTGGC CGTTCAGGGT ACAGGTTGGC CGTTCAGGGT AA
delta     AAACCCTTGC CGTTACGCTT AAACCGAGGC CGGGACACTC AT
epsilon   AAACCCTTGC CGGTACGCTT AAACCATTGC CGGTACGCTT AA
```

seq 1 →
seq 2 →
seq 3 →
seq 4 →
seq 5 →

Parse PHYLIP File

```python
import os
from Bio import AlignIO

# Download the PHYLIP File
if not os.path.isfile("resampled.phy"):
  os.system("wget -c https://bit.ly/3xL6XdX -O resampled.phy")

# Parse all the records of PHYLIP File
alignments = list(AlignIO.parse("resampled.phy", "phylip"))

# Get only the first set of sequences to be aligned
alignment = alignments[0]

# Overall Information
print("Total Records:", len(alignment))
print("Sequence Length:", alignment.get_alignment_length())
print()

# Show the attributes of a record in an alignment
for record in alignment:
  print("{} (ID: {})".format(record.name, record.id))
  print("Description:", record.description)
  print("Features:", record.features)
  print("Annotations:", record.annotations)
  print("Sequence:", record.seq)
  print("DSSP:", record.letter_annotations.get("secondary_structure", "N/A"))
  print()
```

```
Total Records: 5
Sequence Length: 6
```

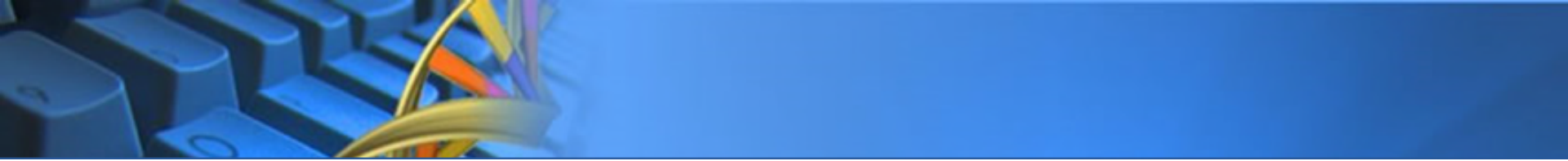
```
Alpha (ID: Alpha)
Description: Alpha
Features: []
Annotations: {}
Sequence: AAACCA
DSSP: N/A
```

Practice

- **Parse PHYLIP File**
  - Write and Run the following codes on a Colab page called “Alignment.ipynb”:

```python
import os
from Bio import AlignIO

# Download the PHYLIP File
if not os.path.isfile("resampled.phy"):
  os.system("wget -c https://bit.ly/3xL6XdX -O resampled.phy")

# Parse all the records of PHYLIP File
alignments = list(AlignIO.parse("resampled.phy", "phylip"))

# Get only the first set of sequences to be aligned
alignment = alignments[0]

# Overall Information
print("Total Records:", len(alignment))
print("Sequence Length:", alignment.get_alignment_length())
print()

# Show the attributes of a record in an alignment
for record in alignment:
  print("{} (ID: {})".format(record.name, record.id))
  print("Description:", record.description)
  print("Features:", record.features)
  print("Annotations:", record.annotations)
  print("Sequence:", record.seq)
  print("DSSP:", record.letter_annotations.get("secondary_structure", "N/A"))
  print()
```

(Solution URL of this Practice)



PAIRWISE ALIGNMENT

Prepare Sequences

```
1  from Bio import Align
2  from Bio import SeqIO
3
4  # Prepare Sequences to Align
5  seq_records = list(SeqIO.parse("PF05371_seed.faa", "fasta"))
6  seq1 = seq_records[0]
7  seq2 = seq_records[1]
8
9  print("{}: {}".format(seq1.id, seq1.seq))
10 print("{}: {}".format(seq2.id, seq2.seq))
11 print()
```

```
COATB_BPIKE/30-81: AEPNAATNYATEAMDSLKTQAIDLISQTWPVVTTVVVAGLVIRLFKKFSSKA
Q9T0Q8_BPIKE/1-52: AEPNAATNYATEAMDSLKTQAIDLISQTWPVVTTVVVAGLVIKLFKKFVSRA
```

Practice

- **Prepare Sequences to be Aligned**
  - Write and Run the following codes on a Colab page called “Alignment.ipynb”:

```
1  from Bio import Align
2  from Bio import SeqIO
3
4  # Prepare Sequences to Align
5  seq_records = list(SeqIO.parse("PF05371_seed.faa", "fasta"))
6  seq1 = seq_records[0]
7  seq2 = seq_records[1]
8
9  print("{}: {}".format(seq1.id, seq1.seq))
10 print("{}: {}".format(seq2.id, seq2.seq))
11 print()
```

(Solution URL of this Practice)

Align Sequences

```
1  # Create PairwiseAligner object
2  aligner = Align.PairwiseAligner()
3  alignments = aligner.align(seq1.seq, seq2.seq)
4
5  # Make Matching Score more strict
6  #aligner.match_score = 100.0
7
8  # Show the overall information
9  print("Score:", aligner.score(seq1.seq, seq2.seq))
10 print("Length:", len(alignments))
11 print()
12
13 # Print each alignment result
14 for alignment in alignments:
15   print(alignment)
```

```
Score: 49.0
Length: 42
```

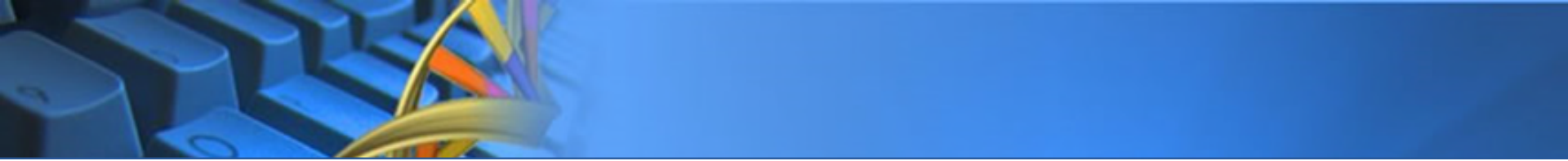
```
AEPNAATNYATEAMDSLKTQAIDLISQTWPVVTTVVVAGLVIR-LFKKF-SSK-A
||||||||||||||||||||||||||||||||||||||||--|||||-|---|
AEPNAATNYATEAMDSLKTQAIDLISQTWPVVTTVVVAGLVI-KLFKKFVS--RA
```

Practice

- **Align Sequences**
  - Write and Run the following codes on a Colab page called “Alignment.ipynb”:

```
# Create PairwiseAligner object
aligner = Align.PairwiseAligner()
alignments = aligner.align(seq1.seq, seq2.seq)

# Make Matching Score more strict
#aligner.match_score = 100.0

# Show the overall information
print("Score:", aligner.score(seq1.seq, seq2.seq))
print("Length:", len(alignments))
print()

# Print each alignment result
for alignment in alignments:
  print(alignment)
```

(Solution URL of this Practice)



BLAST

Query & Save the Result

```
1  import os
2
3  # Download the FASTA File
4  if not os.path.isfile("opuntia.fasta"):
5    os.system("wget https://raw.githubusercontent.com/biopython/biopython/master/Doc/examples/opuntia.fasta")
6
7  # Query BLAST by Sequence
8  from Bio import SeqIO
9  from Bio.Blast import NCBIWWW
10
11 records = list(SeqIO.parse("opuntia.fasta", format="fasta"))
12 record = records[0]
13 result_handle = NCBIWWW.qblast("blastn", "nt", record.seq)
14
15 # Query BLAST by ID
16 #result_handle = NCBIWWW.qblast("blastn", "nt", "8332116")
17
18 out_handle = open("my_blast.xml", "w")
19 out_handle.write(result_handle.read())
20
21 out_handle.close()
22 result_handle.close()
```

blastn, blastp, blastx, tblastn, tblastx

nr, refseq, nt...

NCBI Databases
(Full List)

my_blast.xml

Practice

- **Query BLAST and Save the Result**
  - Write and Run the following codes on a Colab page called “Alignment.ipynb”:

```
import os

# Download the FASTA File
if not os.path.isfile("opuntia.fasta"):
  os.system("wget https://raw.githubusercontent.com/biopython/biopython/master/Doc/examples/opuntia.fasta")

# Query BLAST by Sequence
from Bio import SeqIO
from Bio.Blast import NCBIWWW

records = list(SeqIO.parse("opuntia.fasta", format="fasta"))
record = records[0]
result_handle = NCBIWWW.qblast("blastn", "nt", record.seq)

# Query BLAST by ID
#result_handle = NCBIWWW.qblast("blastn", "nt", "8332116")

out_handle = open("my_blast.xml", "w")
out_handle.write(result_handle.read())

out_handle.close()
result_handle.close()
```

Takes 3~5 minutes

(Solution URL of this Practice)

Show the BLAST Result

```python
from Bio.Blast import NCBIXML

result_handle = open("my_blast.xml")
blast_records = NCBIXML.parse(result_handle)

# Threshold for Significance
E_VALUE_THRESH = 0.05

for blast_record in blast_records:
  for alignment in blast_record.alignments:
      # high-scoring segment pair, HSP
      for hsp in alignment.hsps:
          # If Expect Score E < Threshold = significant enough
          if hsp.expect < E_VALUE_THRESH:
              print("****Alignment****")
              print("sequence:", alignment.title)
              print("length:", alignment.length)
              print("e value:", hsp.expect)
              print(hsp.query[0:75] + "...")
              print(hsp.match[0:75] + "...")
              print(hsp.sbjct[0:75] + "...")
              print()

result_handle.close()
```

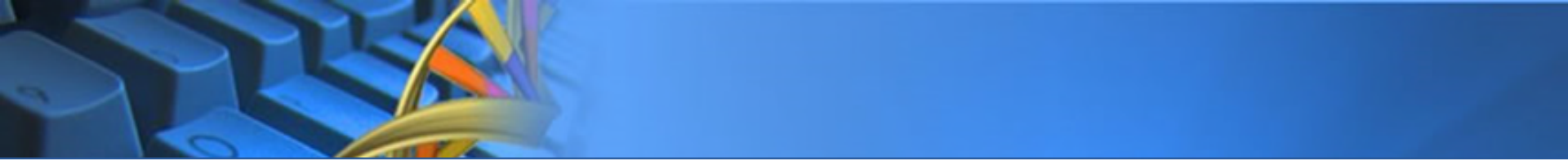
```
****Alignment****
sequence: gi|6273291|gb|AF191665.1|AF191665 Opuntia marenae rpl16 gene; chloroplast gene for chloroplast product, partial intron sequence
length: 902
e value: 0.0
TATACATTAAAGGAGGGGGATGCGGATAAATGGAAAGGCGAAAGAAAGAAAAAAATGAATCTAAATGATATAGGA...
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||...
TATACATTAAAGGAGGGGGATGCGGATAAATGGAAAGGCGAAAGAAAGAAAAAAATGAATCTAAATGATATAGGA...
```

Practice

- **Show the BLAST Result**
  - Write and Run the following codes on a Colab page called “Alignment.ipynb”:

```
from Bio.Blast import NCBIXML

result_handle = open("my_blast.xml")
blast_records = NCBIXML.parse(result_handle)

# Threshold for Significance
E_VALUE_THRESH = 0.05

for blast_record in blast_records:
  for alignment in blast_record.alignments:
      # high-scoring segment pair, HSP
      for hsp in alignment.hsps:
          # If Expect Score E < Threshold = significant enough
          if hsp.expect < E_VALUE_THRESH:
              print("****Alignment****")
              print("sequence:", alignment.title)
              print("length:", alignment.length)
              print("e value:", hsp.expect)
              print(hsp.query[0:75] + "...")
              print(hsp.match[0:75] + "...")
              print(hsp.sbjct[0:75] + "...")
              print()

result_handle.close()
```

(Solution URL of this Practice)



WRITE INTO ALIGNMENT FILES

Create a Set of Alignments

```
from Bio.Seq import Seq
from Bio.SeqRecord import SeqRecord
from Bio.Align import MultipleSeqAlignment

align1 = MultipleSeqAlignment([
             SeqRecord(Seq("ACTGCTAGCTAG"), id="Alpha"),
             SeqRecord(Seq("ACT-CTAGCTAG"), id="Beta"),
             SeqRecord(Seq("ACTGCTAGDTAG"), id="Gamma"),
         ])

align2 = MultipleSeqAlignment([
             SeqRecord(Seq("GTCAGC-AG"), id="Delta"),
             SeqRecord(Seq("GACAGCTAG"), id="Epsilon"),
             SeqRecord(Seq("GTCAGCTAG"), id="Zeta"),
         ])

align3 = MultipleSeqAlignment([
             SeqRecord(Seq("ACTAGTACAGCTG"), id="Eta"),
             SeqRecord(Seq("ACTAGTACAGCT-"), id="Theta"),
             SeqRecord(Seq("-CTACTACAGGTG"), id="Iota"),
         ])

my_alignments = [align1, align2, align3]
```

Write as a PHYLIP File

```
1  from Bio import AlignIO
2
3  num_alignments = AlignIO.write(my_alignments, "my_example.phy", "phylip")
4  print("{} Alignments wrote successfully!".format(num_alignments))
```

3 Alignments wrote successfully!

Practice

- **Write as a PHYLIP File**
  - Write and Run the following codes on a Colab page called “Alignment.ipynb”:

```
1  from Bio import AlignIO
2
3  num_alignments = AlignIO.write(my_alignments, "my_example.phy", "phylip")
4  print("{} Alignments wrote successfully!".format(num_alignments))
```

(Solution URL of this Practice)

Converting Files

- **Convert from Stockholm to Clustal Format**

```
1  from Bio import AlignIO
2
3  num_alignments = AlignIO.convert("PF05371_seed.sth", "stockholm", "PF05371_seed.aln", "clustal")
4  print("Converted {} alignments".format(num_alignments))
```

```
Converted 1 alignments
```

Practice

- **Converting Files**
  - Write and Run the following codes on a Colab page called “Alignment.ipynb”:

```
1  from Bio import AlignIO
2
3  num_alignments = AlignIO.convert("PF05371_seed.sth", "stockholm", "PF05371_seed.aln", "clustal")
4  print("Converted {} alignments".format(num_alignments))
```

(Solution URL of this Practice)

Summary

- **Commonly Used Alignment Algorithms**
  - Smith–Waterman, FASTA, BLAST
- **Commonly Used File Formats**
  - FASTA, Stockholm Format, PHYLIP Format
- **Data Structure for Alignments**
  - Bio.Align.MultipleSeqAlignment
- **Parse Alignment Files**
  - Bio.AlignIO.read()
  - Bio.AlignIO.parse()
- **Pairwise Alignment**
  - Bio.Align.PairwiseAligner()
- **BLAST**
  - Bio.Blast.NCBIWWW.qblast()
- **Write into Alignment Files**
  - Bio.AlignIO.write()