BINF2111 – Introduction to Bioinformatics Computing

Python – There can be only one.

Richard Allen White III, PhD
RAW Lab
Lecture 18 – Thursday Nov 4th, 2021

Learning Objectives

- Lab 8 review

- AT and GC content

- Lists with for loops

- Nest lists

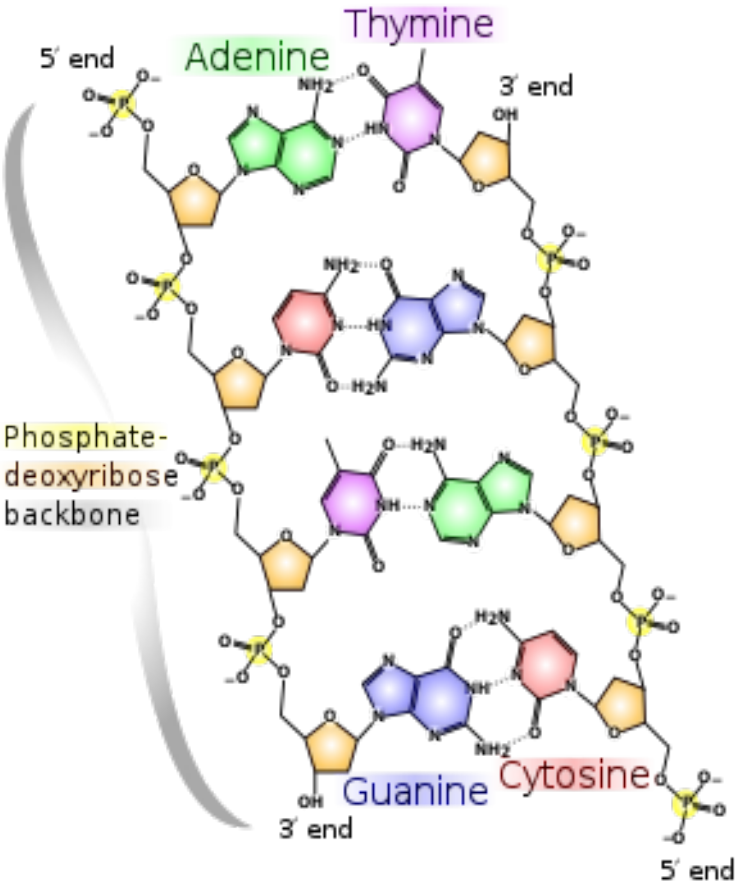
- Cloning/MCS/pUC19

- Quiz 18

Lab Review 8 - AT vs. GT content

$$\text{AT content} = \frac{A+T}{\text{Length}}$$

$$\text{GC content} = \frac{G+C}{\text{Length}}$$



Lab Review 8 - AT vs. GT content

Make a script that count AT content can do this?

Lab Review 8 - AT vs. GT content

```
#!/usr/bin/python3

my_dna = 'ACTGATCCATATATATTTT'
length = len(my_dna)
a_count = my_dna.count(‘A’)
t_count = my_dna.count(‘T’)
at_content = (a_count + t_count) / (length)
print(“Hello…….”)
print("AT content is " + str(at_content))
```

Output is?

Lab Review 8 - AT vs. GT content

```
#!/usr/bin/python3
my_dna = 'ACTGATCCATATATATTTT'
length = len(my_dna)
g_count = my_dna.count(‘G’)
c_count = my_dna.count(‘C’)
gc_content = (g_count + c_count) / (length)
print(“Hello…….”)
print("GC content is " + str(gc_content))
```

Output is? Hello….. ,
GC content is: 0.2105263157894737

Lab Review 8 - AT vs. GT content

```
#!/usr/bin/python3
my_dna = 'ACTGATCCATATATATTTT'
length = len(my_dna)
a_count = my_dna.count('A')
t_count = my_dna.count('T')
g_count = my_dna.count('G')
c_count = my_dna.count('C')
at_content = (a_count + t_count) / (length)
gc_content = (1 - at_content)
print('AT content is: ' + str(at_content))
print('GC content is: ' + str(gc_content))
```

Lab Review 8 - AT vs. GT content

OUTPUT IS

AT content is:
0.7894736842105263
GC content is:
0.21052631578947367

Lists and loops (Python)

Putting all together

```
count = [1, 2, 3, 4, 5]
fruits = ['apples', 'oranges', 'pears', 'apricots']
change = [1, 'pennies', 2, 'dimes', 3, 'quarters']
```

Lists with loops

For loop – number.py

Make a script.

```
#!/usr/bin/python3

count = [1, 2, 3, 4, 5]
for number in count:
    print(number)
```

Output is?

For loop – fruit.py

Make a script.

```
#!/usr/bin/python3

fruits = ['apples', 'oranges', 'pears', 'apricots']
for fruit in fruits:
    print(fruit)
```

Output is?

For loop – fruit.py

Make a script.

```
#!/usr/bin/python3

fruits = ['apples', 'oranges', 'pears', 'apricots']
for i in fruits:
    print(i)
```

Output is?

Lists and loops (Python)

```
# we can also build lists, first start with an empty one
>>> fruits = ['apples', 'oranges', 'pears', 'apricots']
>>> elements = []

>>> for i in fruits:
...     elements.append(i)
...     print(elements)
```

space to output!

Nested lists (Python)

Nested lists: This allows you to call lists within lists. Remember each letter in a string is part of a list.

```
>>> fruits = ['apples', 'oranges', 'pears', 'apricots']


>>> print(elements)
['apples', 'oranges', 'pears', 'apricots']


>>> print(fruits[1])
>>> print(fruits[1][4])
```

Nested lists (Python)

Nested lists: This allows you to call lists within lists. Remember each letter in a string is part of a list.

```
>>> fruits = ['apples', 'oranges', 'pears', 'apricots']
>>> print(elements)
['apples', 'oranges', 'pears', 'apricots']
[     0       ,       1     ,    2    ,       3    ]
[ 012345, 0123456,  01234,  01234567]

>>> print(fruits[1][4])
```

Range (Python)

Range is the range from 0 to the number (exclusive) and is useful for looping.

```
for i in range(5):
        print (i)
```

Output :

```
For i in range(3, 6):
        print (i)
```

Output :

```
For i in range (10,19,2):
        print (i)
```

Output :

Range (Python)

Range is the range from 0 to the number (exclusive) and is useful for looping.

```
seq = open('multiN.fastq')
lines = seq.readlines()
for i in range(0, len(lines), 4):
    print(lines[i].rstrip())
```

Strip lines as you read (Python)

The strip method allows you to remove unwanted characters from text that is slurped into python.

We can combine this with a ‘for’ loop and the range function to strip lines as we read them in.

```
DNA = open(“example2.fasta”)
lines = DNA.readlines()
sequence = []
for i in range (0, len(lines)):
    sequence.append(lines[i].strip(“\n”))
    print(sequence)
```

Strip lines as you read (Python)

The strip method allows you to remove unwanted characters from text that is slurped into python.

We can combine this with a ‘for’ loop and the range function to strip lines as we read them in.

```
DNA = open(“example2.fasta”) #open file
lines = DNA.readlines() #read lines one at a time into
                        variable “lines”
Sequence = [] #create a list “sequence”
for i in range (0, len(lines)):
    sequence.append(lines[i].strip(“\n”))
    #for lines 0 to all the lines
    #append a stripped line to the list


DNA.close()
```

Read vs. readlines

.read() – use when you want to read everything in at once

```
ACGATATAGACAAGATCCCGAATCGATCAGACACAGATAAAAGGACATGGGG
AGACAAAATATACAGTAGTTAGACAGATACACAGTAGAGGATATACCCCCAA
TATAGAATTAGACATTTGTGGTGCAGTTAGTCATAGACATATTAGGGCAATT
```

.readlines() – use when you want to read line by line

```
['ACGATATAGACAAGATCCCGAATCGATCAGACACAGATAAAAGGACATGGGG ',
'AGACAAAATATACAGTAGTTAGACAGATACACAGTAGAGGATATACCCCCAA\n',
'TATAGAATTAGACATTTGTGGTGCAGTTAGTCATAGACATATTAGGGCAATT\n']
```

.readlines()
.rstrip(‘\n’)

```
['ACGATATAGACAAGATCCCGAATCGATCAGACACAGATAAAAGGACATGGGG',
'AGACAAAATATACAGTAGTTAGACAGATACACAGTAGAGGATATACCCCCAA',
'TATAGAATTAGACATTTGTGGTGCAGTTAGTCATAGACATATTAGGGCAATT']
```

Join lines you read in

The join method allows you to join together different things in a list.

```
DNA = open(“example2.fasta”)
lines = DNA.readlines()
sequence = []
for i in range (0, len(lines)):
    sequence.append(lines[i].strip(“\n”))

sequence = "".join(sequence)
print(sequence)
```

Joining the text together

.readlines() – use when you want to read line by line

['ACGATATAGACAAGATCCCGAATCGATCAGACACAGATAAAAGGACATGGGG ',
'AGACAAAATATACAGTAGTTAGACAGATACACAGTAGAGGATATACCCCCAA\n',
'TATAGAATTAGACATTTGTGGTGCAGTTAGTCATAGACATATTAGGGCAATT\n']

.readlines()

.rstrip('\n')

['ACGATATAGACAAGATCCCGAATCGATCAGACACAGATAAAAGGACATGGGG',
'AGACAAAATATACAGTAGTTAGACAGATACACAGTAGAGGATATACCCCCAA',
'TATAGAATTAGACATTTGTGGTGCAGTTAGTCATAGACATATTAGGGCAATT']

.readlines()

.rstrip('\n'), "".join(sequenceName)

ACGATATAGACAAGATCCCGAATCGATCAGACACAGATAAAAGGACATGGGGAGACAAAATATACAGTAG
TTAGACAGATACACAGTAGAGGATATACCCCCAATATAGAATTAGACATTTGTGGTGCAGTTAGTCATAGAC
ATATTAGGGCAATT

Strip lines as you read input using with

We can perform a for loop within a list. This allows us to strip lines into a list as we read them in.

Using the with command (shorthand to automatically close file):

```
with open('filename.txt') as f:
        alist = [line.rstrip() for line in f]
        print(alist)

# opens the file as f.  For line in f, strip, and put in list.
# ['ACGATATAGACAAGATCCCGAATCGATCAGACACAGATAAAAGGACATGGGG',
'AGACAAAATATACAGTAGTTAGACAGATACACAGTAGAGGATATACCCCCAA',
'TATAGAATTAGACATTTGTGGTGCAGTTAGTCATAGACATATTAGGGCAATT']

# you can also join the list of lines using sequence = “”.join(alist)
```

Strip lines as you read input using with

Often times you want to split text into a list. Split is the opposite of join.

Split a string into variables:

```
sitelist = “GCGGCCGC,CTGCAG,AACAAG”
NotI,PstI,Fke2 = sitelist.split(“,”)
print(sitelist)
```

```
>>> sitelist="GCGGCCGC,CTGCAG,AACAAG"
>>> NotI,PstI,Fke2 = sitelist.split(",")
>>> print (sitelist)
GCGGCCGC,CTGCAG,AACAAG
>>>
>>> print (NotI)
GCGGCCGC
>>> print (PstI)
CTGCAG
>>> print (Fke2)
AACAAG
```

Strip lines as you read input using with

Often times you want to split text into a list. Split is the opposite of join.

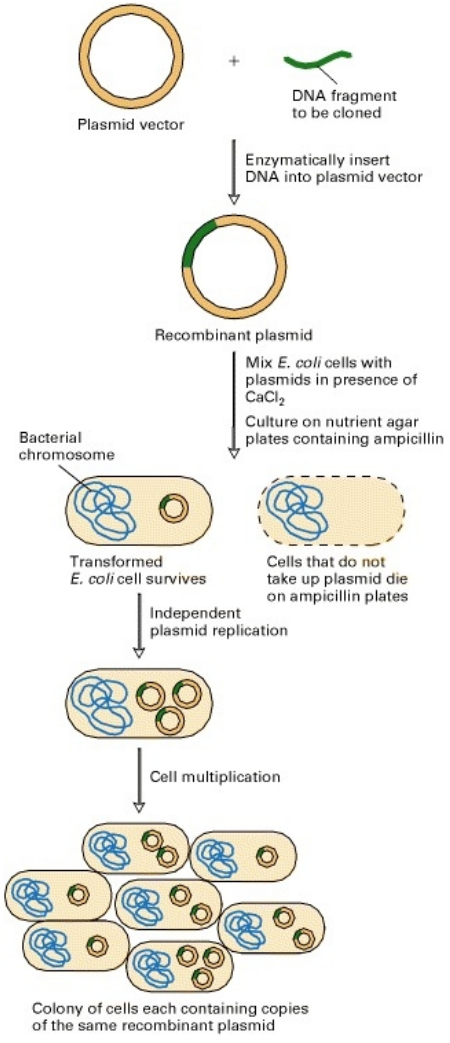
Split a string into a list

```
>>> sitelist = "GCGGCCGC,CTGCAG,AACAAG"
>>> restriction_sites = sitelist.split(",")
>>> print(restriction_sites)
['GCGGCCGC', 'CTGCAG', 'AACAAG']
>>>
```

Cloning

DNA cloning

- Insert DNA into plasmid
- Transfect *e coli* with plasmid
- Select for plasmid using marker
- Expressed desired protein
- Collect protein

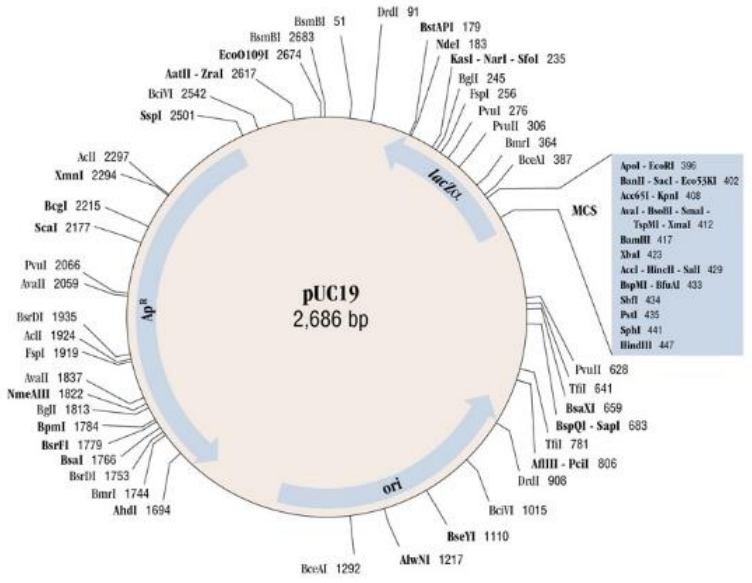


Cloning - pUC19

Cloning vector pUC19c, complete sequence

E. coli

Genback id : L09137



pUC19 MCS

```
        SacI      SmaI      XbaI      SbfI
  EcoRI      KpnI      BamHI     SalI  PstI  SphI  HindIII
agtgAATTCGAGCTCGGTACCCGGGGATCCTCTAGAGTCGACCTGCAGGCATGCAAGCTTGGcgtaatcatggtcat
     .         .         .         .         .         .         .
    400       410       420       430       440       450       460
...S  N  S  S  P  V  R  P  D  E  L  T  S  R  C  A  H  L  S  P  T  I  M  T  M
```

← *lacZα* translational start

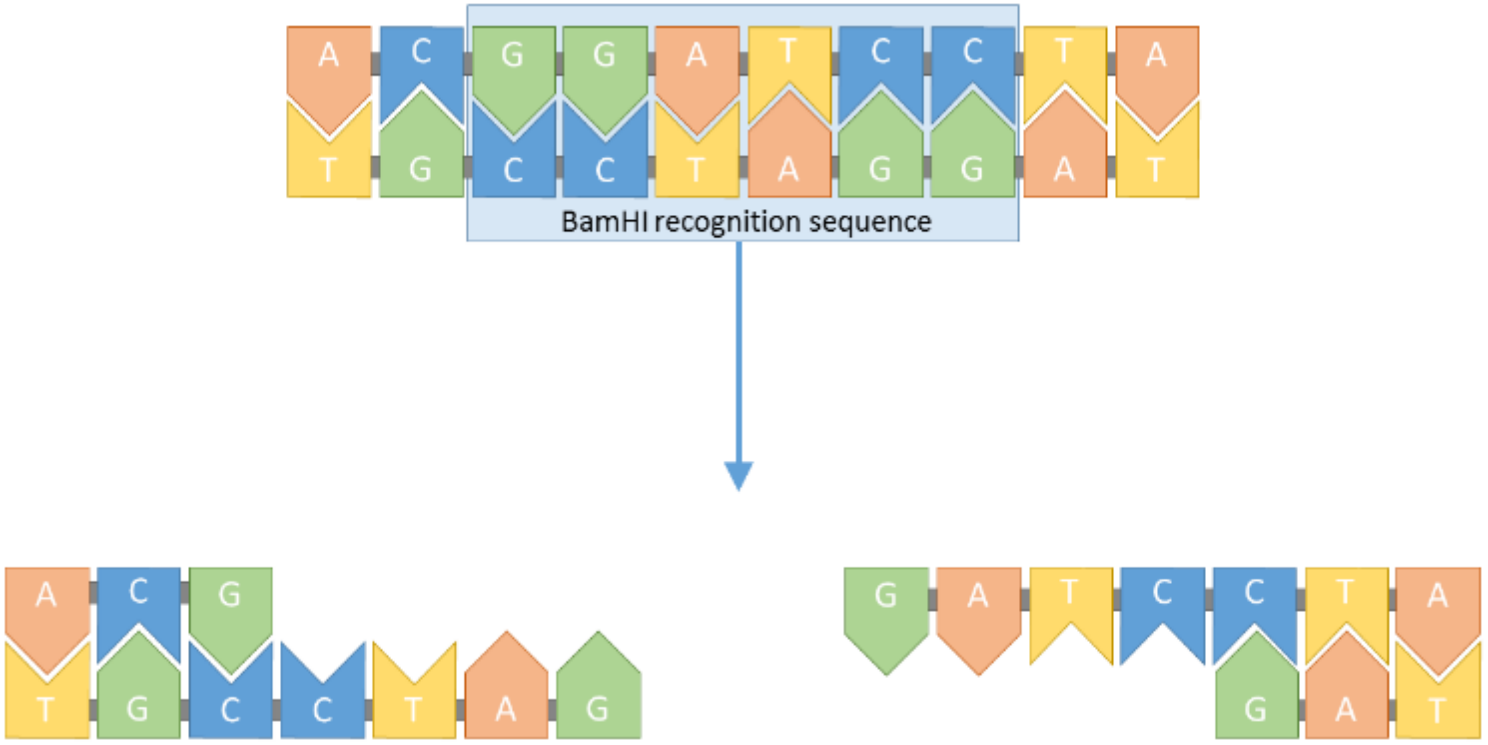
pNEB193 MCS

```
        SacI      SmaI   BssHII      PacI                        SbfI
  EcoRI      KpnI      AscI   BamHI          XbaI  SalI  PmeI    PstI  SphI  HindIII
agtgAATTCGAGCTCGGTACCCGGGGGCGCGCCGGATCCTTAATTAAGTCTAGAGTCGACTGTTTAAACCTGCAGGCATGCAAGCTTGGcgtaatcatggtcat
     .         .         .         .         .         .         .         .         .         .
    400       410       420       430       440       450       460       470       480       490
...S  N  S  S  P  V  R  P  R  A  P  D  K  I  L  D  L  T  S  Q  K  F  R  C  A  H  L  S  P  T  I  M  T  M
```

← *lacZα* translational start

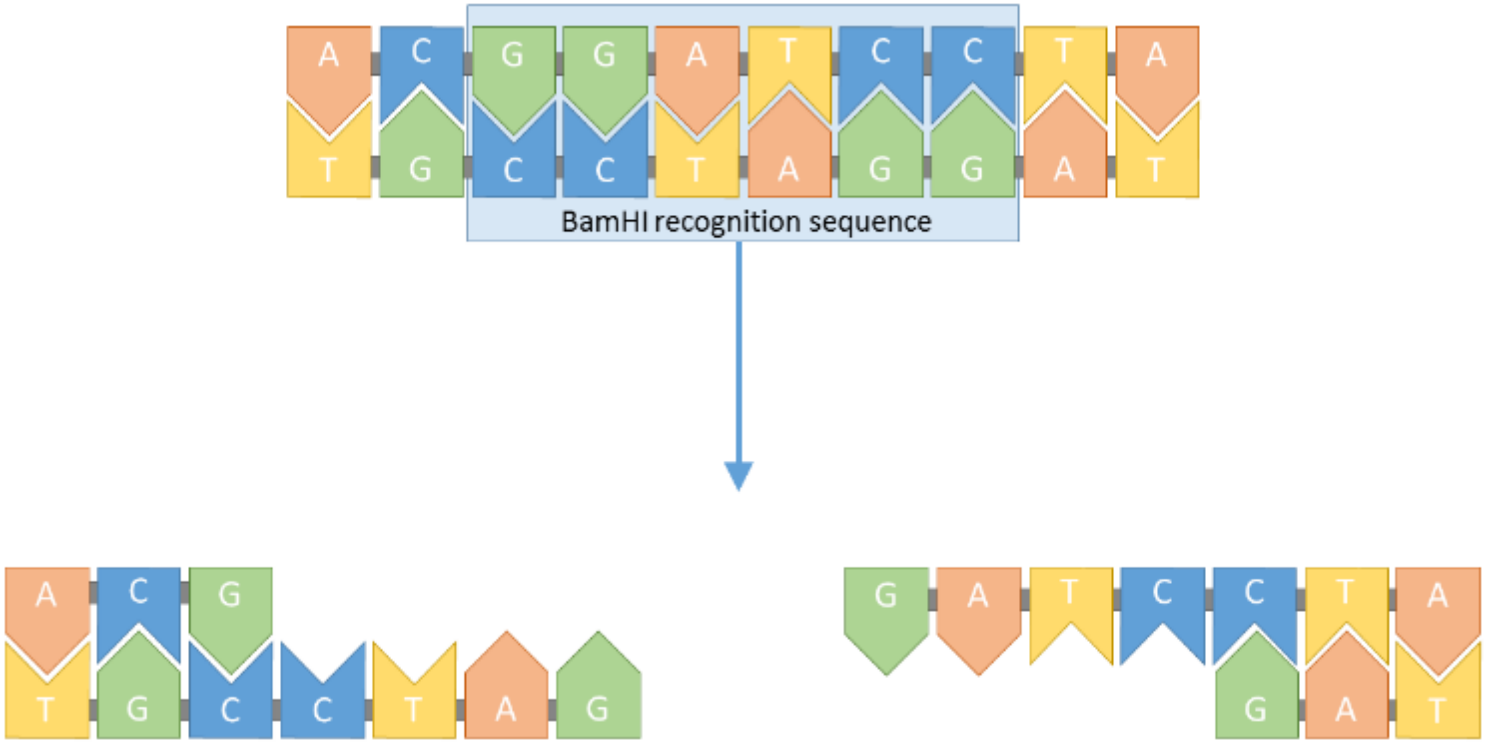
Restriction digestion – Multiple Cloning Site

The lines on the plasmid indicate restriction sites, or particular sequences that will be cleaved when using a restriction enzyme (BamHI).



Restriction digestion – Multiple Cloning Site

DNA can be ligated to these sequences overhanging or blunt ends using DNA ligase.



In silico modification

In bioinformatics, genetic recombineering is first performed in silico prior to biological cloning.

We can use read, split, and lists to mimic what happens during the biological process.

In lab today, you will perform in silico genetic engineering by inserting a piece of DNA into a plasmid.

Quiz 18

- On canvas now