bio334_day1_part3

May 13, 2025

1 Bio334 Practical Bioinformatics

The 2nd module, 14-16, May, 2025

1.1 Masaomi Hatakeyama

- GitHab https://github/masaomi/bio334_2025
- TAs: Narjes Yousefi, Kenji Yip Tong

2 Day1 Part3

Compare more than two Lists

[ ]:
```
import IPython.display
IPython.display.Audio("voice/day1_part3.mp3")
```

In the previous part, you compared two lists, and you compared two nucleotide sequences in the exercise.

In this part, you will compare more than two nucleotide sequences.

In order to compare more than two sequences, I will introduce double for-loop and how to access the two-dimensional data structure, in other words, how to access the elements of a list of lists.

If you feel complicated, please stop by, break it down, and rewrite the actual process without a for-loop statement, which is the specialization thinking process opposite to the generalization.

Let's see the first example of a list of lists below.

[ ]:
```
# List of Lists

lst1 = [0, 1]
lst2 = [2, 3]

lst = [lst1, lst2]
print(lst[0][0]) # => ??
print(lst[0][1]) # => ??
print(lst[1][0]) # => ??
print(lst[1][1]) # => ??
```

```
# a kind of table, two dimensional list
```

[ ]:
```
# List of Strings

str1 = "AT"
str2 = "GC"

lst = [str1, str2]
print(lst[0][0]) # => ??
print(lst[0][1]) # => ??
print(lst[1][0]) # => ??
print(lst[1][1]) # => ??

# a kind of table, two dimensional list
```

[ ]:
```
# List of Strings

str1 = "AT"
str2 = "GC"

lst = [str1, str2]
for i in range(0, 2):     # select string
    for j in range(0, 2): # select position
        print(lst[i][j])

# "Double loop" is useful to access two dimensional data structure
```

[ ]:
```
IPython.display.Audio("voice/day1_part3_example.mp3")
```

The list of lists is a sort of two-dimensional data structure like an Excel table.

The first index corresponds to the index of the list, the second index corresponds to the position of each element of the list.

In this sort of two-dimensional data structure, you can use a double for-loop to access each element.

Please be careful to look at two indexes, i and j. i is the first index and j is the second index of the list of the lists.

The next example is the permutation. It uses the same double loop but it uses only one single list.

3 Permutation

```
lst = ["a", "b", "c", "d", "e"]
for i in range(0, len(lst)):
    for j in range(0, len(lst)):
        if not lst[i] == lst[j]
            print(lst[i], lst[j])
```

- How many times is *print()* repeated?

[ ]:
```python
# Permutation

lst = ["a", "b", "c", "d", "e"]

print("permutations")
for i in range(0, len(lst)):
    for j in range(0, len(lst)):
        if not lst[i] == lst[j]:
            print(lst[i], lst[j])
```

4 Permutation

```
lst = ["a", "b", "c", "d", "e"]
for i in range(0, len(lst)):
    # process1
```

- How many times is *process1* repeated?

5 Permutation

```
lst = ["a", "b", "c", "d", "e"]
for i in range(0, len(lst)):
    # process1
    for j in range(0, len(lst)):
        # process2
```

- How many times is *process2* repeated?

6 Permutation

```
for i in range(0, len(lst)):
    # process1
    for j in range(0, len(lst)):
        # process2
        if not lst[i] == lst[j]
            # process3
```

- How many times is *process3* repeated?

7 Combination

```
lst = ["a", "b", "c", "d", "e"]
 for i in range(0, len(lst)):    # 5 times
  for j in range(i+1, len(lst)): # 5-(i+1) times
   print(lst[i], lst[j])
```

- Q: How many times is *print()* repeated?

[ ]:
```
IPython.display.Audio("voice/day1_part3_combination.mp3")
```

In the permutation, the order of elements is considered.

For example (a,b) and (b,a) are different elements in the permutation.

On the other hand, in the combination, the order is not considered. (a,b) and (b,a) is the same element.

In order to calculate the nucleotide diversity, we need the pairwise combination of sequences.

So, regardless of the number of sequences, we can use the for-double loop to make a pairwise combination of the sequences, and I remind you back that we can use *for* and *if* structure, in order to compare the elements of two lists.

This is the hint to the final exercise today.

8 Mini-Summary1

Generalization * In order to process two dimensional data, use double loop **for**

```
#List of list
for <varriable> in range(0, len(<list>)):
    for <varriable> in range(0, len(<list>)):
      # process for each element
```

9 Mini-Summary2

Generalization * In order to process the combination of a list, use double loop **for**

```
#Combination of a list
for i in range(0, len(<list>)):
   for j in range(i+1, len(<list>)):
    # process of each combination
```

- Be careful of the second loop index, begin with **i+1**.

[ ]:
```
IPython.display.Audio("voice/summary_day1_part3.mp3")
```

Now you have a chance to generalize your knowledge.

If you want to have a pairwise combination of some lists, you can use the double loop by for-statement with being careful about the second index.

If you understand this technique, you can finally implement the nucleotide diversity.

Please make sure the definition of the nucleotide diversity again. And let's go on to the exercise.

10 Day1 Part3

Nucleotide Diversity

- d: number of nucleotide differences (pair-wise)

- n: number of sequences
- l: length of sequence

Q: How do you design the code?