BMEG3105
Lecturer: Yu LI (李煜)
Liyu95.com
liyu@cse.cuhk.edu.hk

# Lecture 2: Data & Python & Sequence

Fall 2023
Wed, 13 September 2023
Scriber: Alpha Cheung
1155133865@link.cuhk.edu.hk

## 1.0: Different Data Types

### • Sequential Data

- DNA, RNA, protein sequences. E.g.:

```
1 MLRQVLHRGLRTCFSRLGHFIASHPVFFASAPVLISILLGASFSRYQVEE  50
|||||||||||||||||||||||||||||||||||||||||||||||||||
1 MLRQVLHRGLRTCFSRLGHFIASHPVFFASAPVLISILLGASFSRYQVEE  50
```

(Fig 1: Global alignment between 2 protein sequences)

### • Data Matrix

- Data that consists of a collection of records, each of which consists of a fixed set of attributes
- Data set can be represented by an *n* by *m* matrix, where there are *n* rows, one for each object, and *m* columns, one for each attribute. E.g.:

| Person | Height (m) | Weight (kg) |
|---|---|---|
| P1 | 1.79 | 75 |
| P2 | 1.64 | 54 |
| P3 | 1.70 | 63 |
| P4 | 1.88 | 78 |

(Fig 2: A 4 by 2 matrix. 4 people, each with 2 attributes)

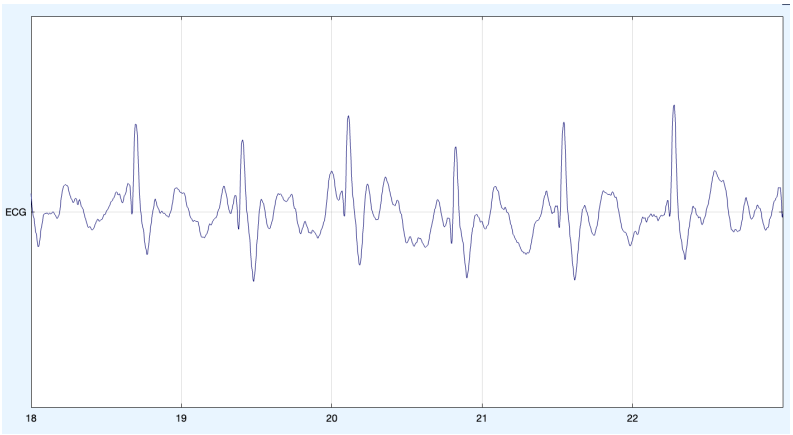
- (#Scriber's note: Other common examples are: Excel files, CSV files)
- If you shuffle an entire column or the entire row independently, the data will not change
- However, if you shuffle both a column and a row at the same time, the data may change
- Images can be stored in a data matrix, in the form of pixels

### • Spatial Data

- Geographic locations
- Spatial information (E.g. images)

## • Temporal Data

- Data involving time. E.g.: stock market trends, ECG/EEG signal recordings




(Fig 3: ECG recording, at time interval 18-23s)

## • Graph or Networks

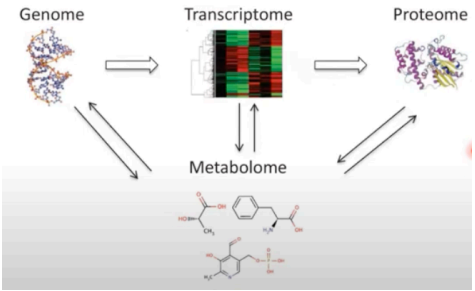
- Objects and connections
- Social network and PPI network

## • Text

- Short sentences, long sentences or documents

## • Multi-Modality Data

- Data that contains multiple data types
  - Video: Temporal images + audio + transcript
  - Electronic health records: Data matrix + images + text
  - Spatial transcriptomics: Spatial data + sequence + data matrix




(Fig 4: Example of a spatial transcriptomic)

- **Unknown Data Type(s)**
  - Data that is not initially shown

# 2.0: Introduction to Python Programming

## 2.1: Definition of "Computer Programming"

- Wikipedia Definition: "The process of designing and building an executable computer program to accomplish a specific computing result or to perform a specific task"
- Easy-to-Understand Definition: "Communication with a computer in order to tell the computer to complete some tasks/solve problems"

## 2.2: Python

- Wikipedia Definition: "An interpreted high-level general-purpose programming language"
- Easy-to-Understand Definition: "A software to allow communication with a computer"
  - Learning Python programming = learning how to use Python to communicate with a computer (like learning another language based on english)
    - Computer does not understand human language; human does not understand computer language
    - Python programming = translator for language defined by python to machine language
- Python can be used to:
  - Calculate mean, variance, distance, gradient…
  - (#Scriber's note: perform data analytics, deep learning by training artificial networks, create web applications)

**Example:**

English: Please calculate the mean of 1, 2 and 3 and then return the solution to me

Python:
```
import numpy
numpy.mean([1,2,3])
```

Computer: 2.0

## 2.3: Python Plugins (Libraries)

- To make Python more powerful
- Commonly used: Numpy, Scipy, Pandas
- #Scriber's Note: other examples of commonly used Python plugins: Matplotlib Pyplot (for plotting graphs), Scikit-Learn, Pytorch, Tensorflow (for machine learning and deep learning)
- To use these libraries, you have to import them at the start of your Python code. E.g.:

```
import numpy
from matplotlib import pyplot as plt
import tensorflow as tf
from tensorflow import keras
```

- #Scriber's Note 2: Before you can import these libraries, you have to install them onto your computer, either by downloading their packages from the internet, or using code. E.g.:

```
pip install numpy
conda install -c anaconda pandas
```

## 2.4: Variables in Python

- Numbers, arrays and strings can be stored in "variables" in Python. Variables are defined by the user. E.g.:
  - x = 0.5 (variable "x" stores integer value 0.5)
  - y = [1, 2, 3, 4, 5] (variable "y" stores a 1D array consisting of five numbers 1, 2, 3, 4, 5)
  - z = "Hello World!" (variable "z" stores a sentence Hello World!)
- Variables are useful as they can be reused in the user's code at any time and place, reducing repetitiveness and tediousness of typing out the variable's content over and over (e.g. it is much more simple to just type "y" instead of "[1, 2, 3, 4, 5]" 50 times in the same code)
- Notation matters in Python. E.g. if you want the variable "y" to store the array [1, 2, 3, 4, 5], then you MUST initialize the variable y as "y = [1, 2, 3, 4, 5]", or else Python will return an error

## 2.5: Printing in Python

- You can ask the computer to print out numbers, arrays, strings etc. in Python. E.g.

```
print(5.3)
```

```
5.3
```

- You can also print out the content of variables (NOT the variable name) you have defined. E.g.:

```
n = [1,2,3,4,5]
print(n)
```

```
[1, 2, 3, 4, 5]
```

- If you want to print out an exact phrase, use quotation marks “ ” or ‘’ around the phrase. E.g.:

```
print("I want to get grade A for BMEG3105!")
```

```
I want to get grade A for BMEG3105!
```

# 3.0: Sequence Data

## 3.1: Why do we want to study sequence data?

- Central dogma of molecular biology - genetic information only flows in one direction: DNA -> RNA -> Protein
- In other words, all genetic information originates from DNA
- These sequences determine a lot of biological information, like phenotypes
- Phenotype = Genotype + environment (genotype is determined by sequences)

## 3.2: What are the sequence data?

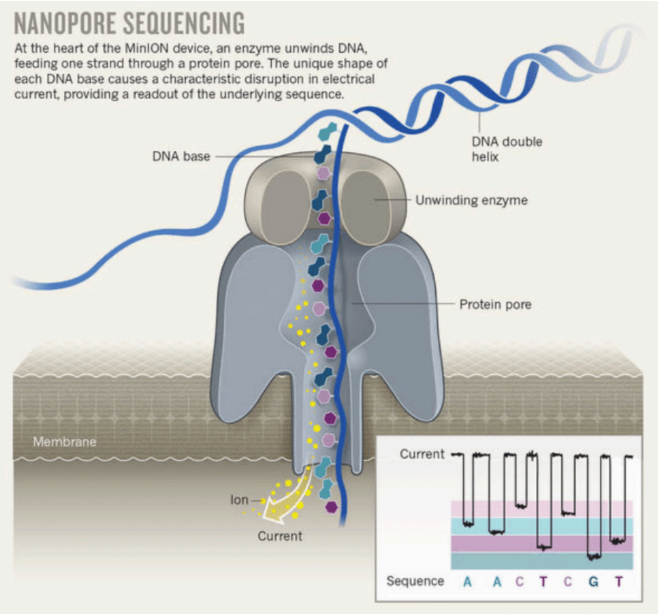
- DNA Sequence
  - Represented by A, T, C, G
  - Complementary double helix structure
  - Approximately 3 billion base pairs
  - E.g., TATACATTAG
- RNA Sequence
  - Represented by A, T, C, U
  - E.g., UAUACAUUAG
- Protein Sequence
  - Represented by A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y
  - E.g., YTL

BMEG3105
Lecturer: Yu LI (李煜)
Liyu95.com
liyu@cse.cuhk.edu.hk

Fall 2023
Wed, 13 September 2023
Scriber: Alpha Cheung
1155133865@link.cuhk.edu.hk

## 3.3: How do we get the sequence data?

- DNA/RNA sequencing
  - Still under active development (from 1977)
  - From short reads to long reads
- Examples: Sanger method (1977), PacBIO RS (2011), Nanopore sequencing (2014)
- Nanopore sequencing (for DNA sequences):
  - Works by monitoring changes to an electrical current as nucleic acids are passed through a protein nanopore
  - The resulting signal is decoded to provide the specific DNA or RNA sequence
  - Very long (up to 3Mb, vs 1000bp for Sanger sequencing)
  - Error rate is high (5%, vs 0.001% for Sanger sequencing)
  - Under active development




(Fig 5: Visualization of nanopore sequencing)

- Protein sequencing (for protein sequences):
  - Based on mass spectrometry
    - Break the long sequence into short pieces
    - Determine the weight of each short piece
    - Assemble the short pieces into the raw complete sequence

### 3.4: What are the raw data and what do we do to them?

- DNA sequences
  - Detect phenotype associated variants
  - Map reads to reference genome
- Protein sequence
  - Sequence comparison
  - Multiple sequence alignment
  - Similar sequence -> similar structure -> similar function
    - To infer the function and properties of newly discovered protein -> drug development
    - Most likely common ancestor -> investigating mechanism, identifying conservative regions

# 4.0: Sequence Comparison and Alignment Score

## 4.1: Sequence alignment

- Definition: Given a set of sequences, an alignment is the same set of sequences with zero or more gaps inserted into them so that:
  1. They all have the same length afterwards
  2. For each column (also called a position or a site), at least one of the resulting sequences is not a gap
- To determine the similarity between sequences and identify regions of similarity
- Pairwise sequence alignment -> arrange two sequences to maximise the similarity between them
- How to define sequence similarity?
  - Match (e.g.: A <-> A, C <->C)
  - Mismatch (substitution) (e.g.: A <-> C, T <-> G)
  - Gap (indel - insertion or deletion) (e.g.: C <-> _)
- How to score an alignment?
  - Use a scoring matrix, that defines scores for match, mismatch and gaps. E.g.:

BMEG3105
Lecturer: Yu LI (李煜)
Liyu95.com
liyu@cse.cuhk.edu.hk

Fall 2023
Wed, 13 September 2023
Scriber: Alpha Cheung
1155133865@link.cuhk.edu.hk

| | A | C | G | T |
|---|---|---|---|---|
| A | 2 | -7 | -5 | -7 |
| C | -7 | 2 | -7 | -5 |
| G | -5 | -7 | 2 | -7 |
| T | -7 | -5 | -7 | 2 |

Gap penalty = -10

Alignment score 1 = 2 + (-7) + 2 + 2 + (-10) + 2
= **-9**

A G G C C G
A T G C _ G

Alignment score 2 = 2 + (-7) + 2 + 2 + (-7) + (-10)
= **-18**

A G G C C G
A T G C G _

(Fig 6: Example scoring matrix, and calculation of alignment score between some example DNA sequences)

## 4.2: How to find the best pairwise alignment?

- Straightforward solution: enumerate all possible alignments between two sequences
  - Calculate the scores for all the alignments
  - Select the one with the highest score (highest similarity)
- Problem: too many possible alignments
  - Given first sequence of length m and second sequence of length n ($m \leq n$):
    - Minimum number of gaps that can be added to first sequence: n-m
    - Maximum number of gaps that can be added to first sequence: n
  - For x gaps inserted into the first sequence:
    - There are ${}_{x+m}C_x$ ways to insert the gaps
    - Need to insert x+m-n gaps into the second sequence
  - Therefore, the total number of possible alignments is:

$$\sum_{x=n-m}^{n} \binom{x+m}{x}\binom{m}{x+m-n} = \sum_{x=n-m}^{n} \frac{(x+m)!}{x!\,m!}\frac{m!}{(x+m-n)!\,(n-x)!} = \sum_{x=n-m}^{n} \frac{(x+m)!}{x!\,(x+m-n)!\,(n-x)!}$$

| n \ m | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 3 | | | | | | | | | |
| 2 | 5 | 13 | | | | | | | | |
| 3 | 7 | 25 | 63 | | | | | | | |
| 4 | 9 | 41 | 129 | 321 | | | | | | |
| 5 | 11 | 61 | 231 | 681 | 1683 | | | | | |
| 6 | 13 | 85 | 377 | 1289 | 3653 | 8989 | | | | |
| 7 | 15 | 113 | 575 | 2241 | 7183 | 19825 | 48639 | | | |
| 8 | 17 | 145 | 833 | 3649 | 13073 | 40081 | 108545 | 265729 | | |
| 9 | 19 | 181 | 1159 | 5641 | 22363 | 75517 | 224143 | 598417 | 1462563 | |
| 10 | 21 | 221 | 1561 | 8361 | 36365 | 134245 | 433905 | 1256465 | 3317445 | 8097453 |

(Fig 7: Table demonstrating the exponential growth of possible alignments as sequences get longer)