

Discrete Structures: CMPSC 102

Oliver BONHAM-CARTER

Spring 2024
Week 11

Let's Discuss

Discrete
Structures:
CMPSC 102

Oliver
BONHAM-
CARTER

Let's Discuss

Analysis

Consider this!

Key Questions

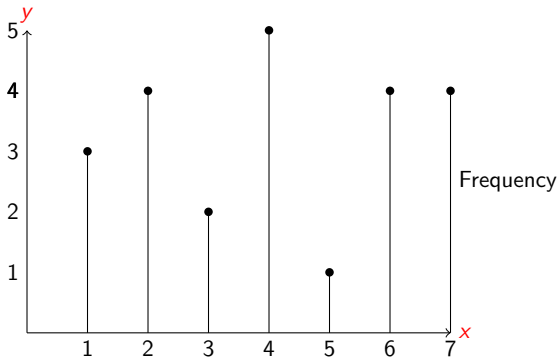
How can I create basic statistics from text and then explain my results using values and plots?

Learning Objectives

To **remember** and **understand** some concepts about **plots**, and the code used to make them from `matplotlib`.

Frequencies

- The frequency can be understood as the number of occurrences of a particular value or range of values.



What characters occur and how often?

Function: calculate_character_frequencies

- This function calculates the frequencies of individual characters in a given list of texts.
- We use Python's Counter class from the collections module.

```
import matplotlib.pyplot as plt
from collections import Counter

def calculate_character_frequencies(texts):
    # Concatenate all texts into a single string
    all_text = "".join(texts)

    # Count the frequencies of each character
    char_freq = Counter(all_text)

    return char_freq
```

Function:

calculate_character_pairs_frequency

- This function converts the list elements into a blob of text for analysis of pairs of characters.

```
def calculate_character_pairs_frequency(text_list):  
    # Remove any non-alphanumeric characters  
    # and convert text to lowercase convert list  
    # of strings to a single blob of text  
  
    text = ""  
    for i in text_list:  
        text += i  
    text = ''.join(filter(str.isalnum, text.lower()))  
    # Generate list of character pairs (bigrams)  
    pairs = [text[i:i+2] for i in range(len(text)-1)]  
    # Count the frequencies of each character pair  
    pairs_frequency = Counter(pairs)  
    return pairs_frequency  
# end of calculate_character_pairs_frequency()
```

Function:

calculate_character_triples_frequency

- This function converts the list elements into a blob of text for analysis of triplets of characters.

```
def calculate_character_triples_frequency(text_list):
    # Remove any non-alphanumeric characters and convert text to lowercase
    # convert list of strings to a single blob of text
    text = ""
    for i in text_list:
        text += i

    text = ''.join(filter(str.isalnum, text.lower()))

    # Generate list of character triples (3-grams)
    triples = [text[i:i+3] for i in range(len(text)-1)]

    # Count the frequencies of each character pair
    triples_frequency = Counter(triples)

    return triples_frequency
# end of calculate_character_triples_frequency()
```

Function: `plot_character_frequencies`

- This function plots the character frequencies calculated by `calculate_character_frequencies`.

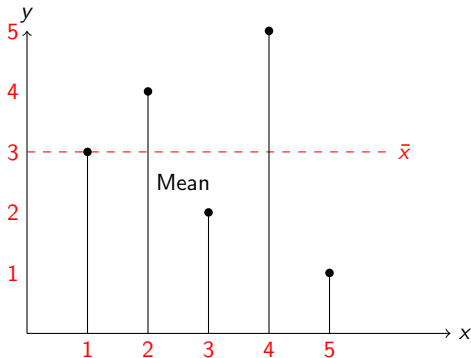
```
import matplotlib.pyplot as plt

def plot_character_frequencies(char_freq):
    # Prepare data for plotting
    characters = list(char_freq.keys())
    frequencies = list(char_freq.values())

    # Plotting
    plt.bar(characters, frequencies)
    plt.title('Character Frequencies')
    plt.xlabel('Character')
    plt.ylabel('Frequency')
    plt.xticks(rotation=45)
    plt.show()
    # end of plot_character_frequencies()
```

Function: `calculate_mean`

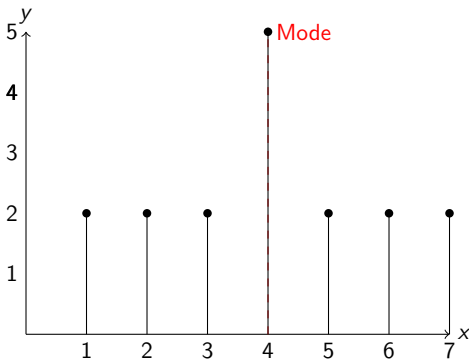
- Data points on a graph, and a dashed line representing the mean (average) of the data points



```
def calculate_mean(frequencies):  
    return sum(frequencies) / len(frequencies)
```


Function: `calculate_mode`

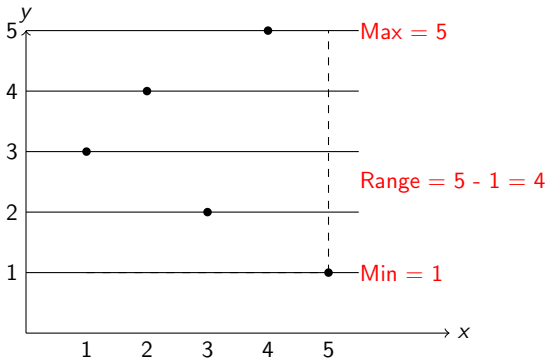
- The mode is the most commonly occurring value, labelled in the data



```
def calculate_mode(frequencies):  
    mode = max(frequencies, key=frequencies.count)  
    return mode
```

Function: calculate_range

- Data points on a graph, and dashed lines representing the range of the data points



```
def calculate_range(frequencies):  
    return max(frequencies) - min(frequencies)
```

Function: `singles()`

- This function handles the singles analysis

```
def singles(texts):  
  
    ## singles analysis  
    # Calculate character frequencies  
    char_freq = calculate_character_frequencies(texts)  
  
    # Plot character frequencies  
    plot_character_frequencies(char_freq)  
  
    # Calculate mean, mode, and range of frequencies  
    frequencies = list(char_freq.values())  
    mean_frequency = calculate_mean(frequencies)  
    mode_frequency = calculate_mode(frequencies)  
    frequency_range = calculate_range(frequencies)  
  
    print("Mean Frequency:", mean_frequency)  
    print("Mode Frequency:", mode_frequency)  
    print("Frequency Range:", frequency_range)  
    # end of singles()
```

Function: pairs()

Discrete
Structures:
CMPSC 102

Oliver
BONHAM-
CARTER

Let's Discuss

Analysis

Consider this!

- This function handles the pairs analysis

```
def pairs(texts):  
    ## pairs analysis  
    charPair_freq = calculate_character_pairs_frequency(texts)  
    # Plot character frequencies  
    plot_character_frequencies(charPair_freq)  
  
    # Calculate mean, mode, and range of frequencies  
    frequencies = list(charPair_freq.values())  
    mean_frequency = calculate_mean(frequencies)  
    mode_frequency = calculate_mode(frequencies)  
    frequency_range = calculate_range(frequencies)  
  
    print("Mean Frequency:", mean_frequency)  
    print("Mode Frequency:", mode_frequency)  
    print("Frequency Range:", frequency_range)  
# end of pairs()
```

Function: triples()

Discrete
Structures:
CMPSC 102

Oliver
BONHAM-
CARTER

Let's Discuss

Analysis

Consider this!

- This function handles the triples analysis

```
def triples(texts):
    ## triples analysis
    charTriples_freq = calculate_character_triples_frequency(texts)
    # Plot character frequencies
    plot_character_frequencies(charTriples_freq)

    # Calculate mean, mode, and range of frequencies
    frequencies = list(charTriples_freq.values())
    mean_frequency = calculate_mean(frequencies)
    mode_frequency = calculate_mode(frequencies)
    frequency_range = calculate_range(frequencies)

    print("Mean Frequency:", mean_frequency)
    print("Mode Frequency:", mode_frequency)
    print("Frequency Range:", frequency_range)
# end of triples()
```

Function: main()

- This function introduces the text and calls to the other functions

```
def main():  
  
    texts = [  
        "hello world",  
        "python programming",  
        "data analysis is fun",  
        "machine learning"  
    ]  
    singles(texts)  
    pairs(texts)  
    triples(texts)  
# end of main()
```

Output: the single characters

Discrete
Structures:
CMPSC 102

Oliver
BONHAM-
CARTER

Let's Discuss

Analysis

Consider this!

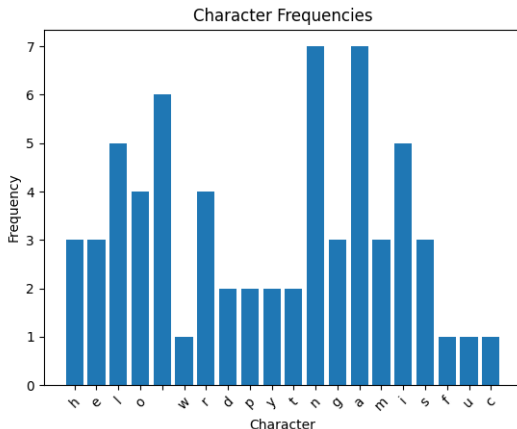


Figure: The single characters shown as frequencies.

Output: the pairs of characters

Discrete
Structures:
CMPSC 102

Oliver
BONHAM-
CARTER

Let's Discuss

Analysis

Consider this!

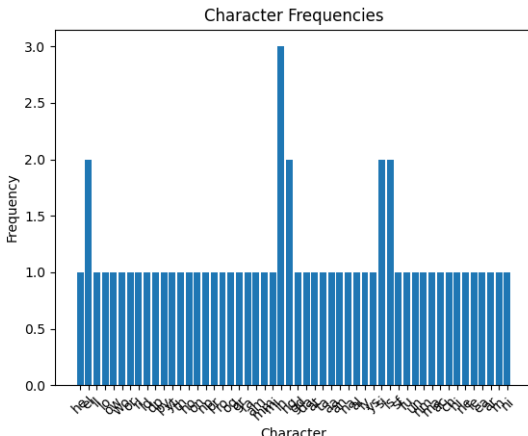
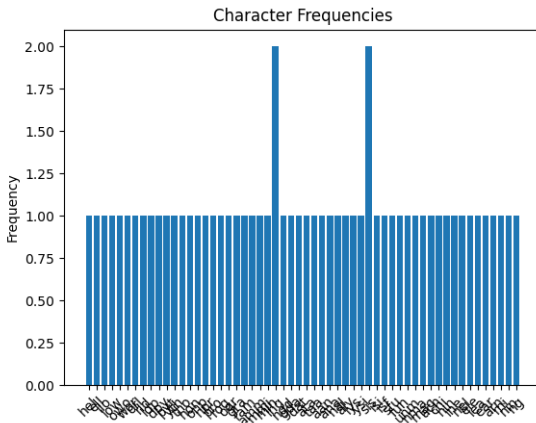


Figure: The pairs of characters shown as frequencies.



Discrete
Structures:
CMPSC 102



Your Turn!

Discrete
Structures:
CMPSC 102

Oliver
BONHAM-
CARTER

Let's Discuss

Analysis

Consider this!

- Find some text from a news article to add to your code (note, you may need to create a large string declaration!!)
- What are the top three most common characters in English?
- Can you find English words having the following combinations:
 - { "aa", "ea", "th", "zz", "ty" }
 - { "aeo", "eab", "pho", "gea", "tyr" }



THINK