Mid-term Exam

April 14, 2023

Manuel Aragon ma1162

0.0.1 CSCE 4290 Mid-term Exam (100 points in total)

Question 1 (5 pt). Make an array called `weird_numbers` containing the following numbers (in the given order):

1. -3
2. the sine of 2.1
3. 5
4. 4 to the power of the cosine of 4.2

Hint: `sin` and `cos` are functions in the `math` module.

[10]:
```python
import math
weird_numbers = [-3, math.sin(2.1), 5, pow(4, math.cos(4.2))]
```

Question 2 (5 pt). Write a simple function that takes in a number (weight in pounds) and returns a number which is the coreesponding conversion to kg.

Test it by calling the function on 15 and 27. E.g., `convert_pounds_to_kg(15)`, `convert_pounds_to_kg(27)`. Print both to screen.

Hint: 1 pound = 0.453592 kg

[11]:
```python
def convert_pounds_to_kg(number):
    """Converts a number in pounds to kg."""
    return number * 0.453592


print(convert_pounds_to_kg(15))
print(convert_pounds_to_kg(27))
```

```
6.8038799999999995
12.246984
```

Question 3 (5 pt). We've loaded an array of temperatures in the next cell. Each number is the highest temperature observed on a day at a climate observation station, mostly from the US. Since they're from the US government agency NOAA, all the temperatures are in Fahrenheit. Convert them all to Celsius by first subtracting 32 from them, then multiplying the results by $\frac{5}{9}$. Make sure to **ROUND** the final result after converting to Celsius to the nearest integer using the `np.round` function. Download the data from canvas: Temperature.csv

[12]:
```python
import numpy as np

# Load the temperature data from a file called 'temperatures.txt'
temperatures_data = np.loadtxt('temperatures.txt', delimiter=',', skiprows=1)

# Extract the maximum temperatures in Fahrenheit from the data
max_temps_fahrenheit = temperatures_data[:, 0]

# Convert the maximum temperatures from Fahrenheit to Celsius
max_temps_celsius = np.round((max_temps_fahrenheit - 32) * (5/9))

# Print the maximum temperatures in Celsius
print(max_temps_celsius)
```

```
[-4. 31. 32. … 17. 23. 16.]
```

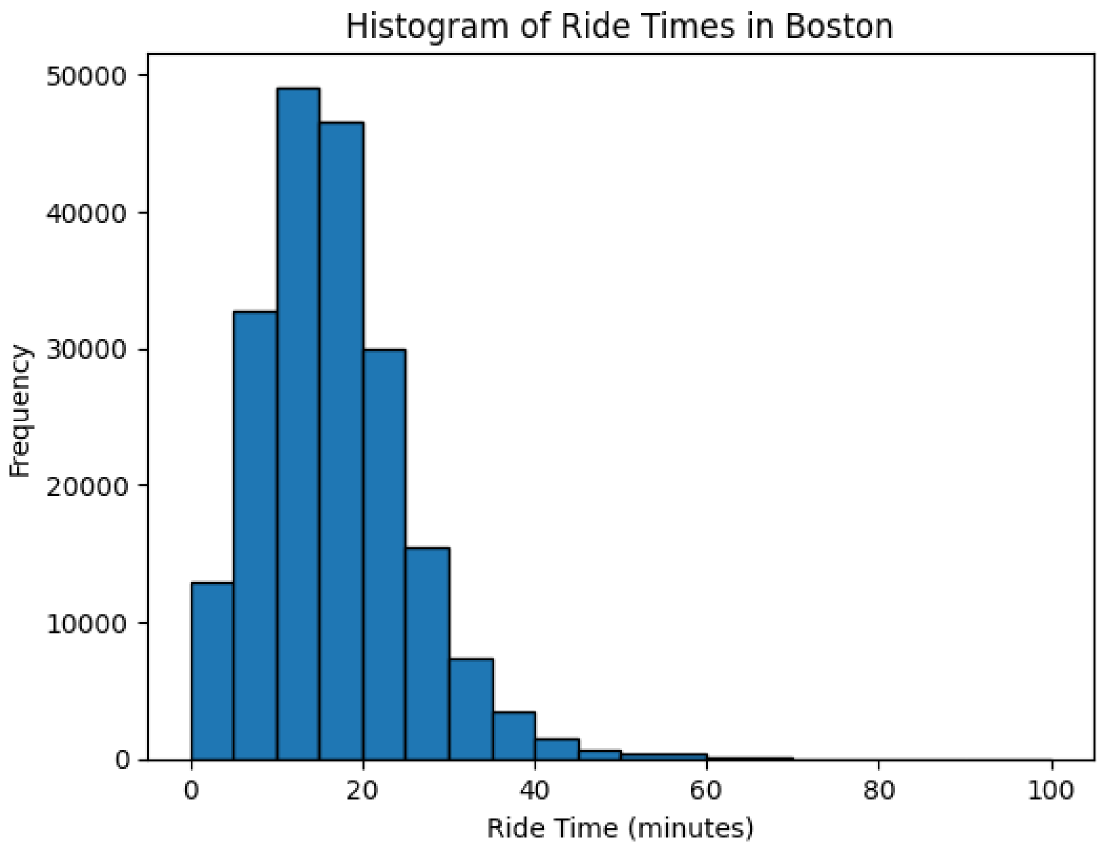
Question 4 (5 pt). Suppose you have 4 apples, 3 oranges, and 3 pineapples. (Perhaps you're using Python to solve a high school Algebra problem.) Create a table that contains this information. It should have two columns: `fruit name` and `count`. Assign the new table to the variable `fruits`.

Note: Use lower-case and singular words for the name of each fruit, like `"apple"`.

[13]:
```python
fruits = [("apple", 4), ("orange", 3), ("pineapple", 3)]
```

Question 5 (20 pt). Below we load a table containing 200,000 weekday Uber rides in the Boston, Massachusetts metropolitan area from the Uber Movement project. The `sourceid` and `dstid` columns contain codes corresponding to start and end locations of each ride. The `hod` column contains codes corresponding to the hour of the day the ride took place. The `ride time` column contains the length of the ride, in minutes. Produce a histogram of all ride times in Boston using the given bins. Download the data from canvas: boston.csv

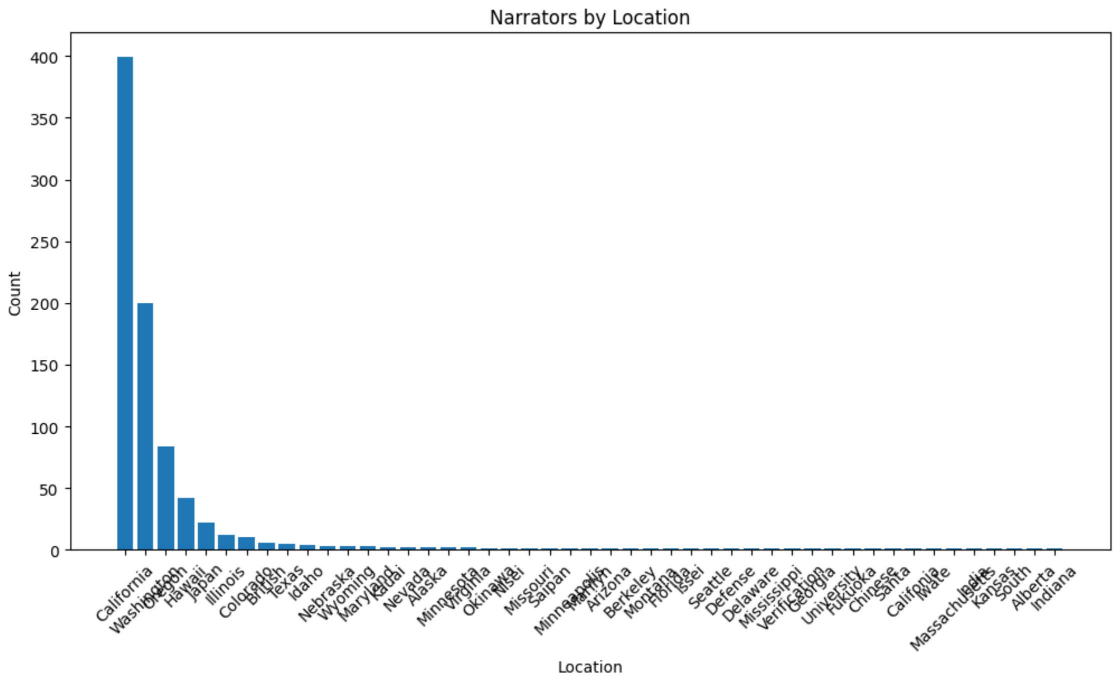
[14]:
```python
import numpy as np
import matplotlib.pyplot as plt

# Load the ride data
# The data was a txt file not csv
boston_ride_data = np.loadtxt('boston.txt', delimiter=',', skiprows=1)

# Extract the ride times in minutes
boston_ride_times = boston_ride_data[:,3]

# Define the bins for the histogram
ride_time_bins = [0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 60, 70, 80, 90, 100]

# Create the histogram
plt.hist(boston_ride_times, bins=ride_time_bins, edgecolor='black')

# Set the axis labels and title
plt.xlabel('Ride Time (minutes)')
plt.ylabel('Frequency')
plt.title('Histogram of Ride Times in Boston')

# Show the plot
plt.show()
```

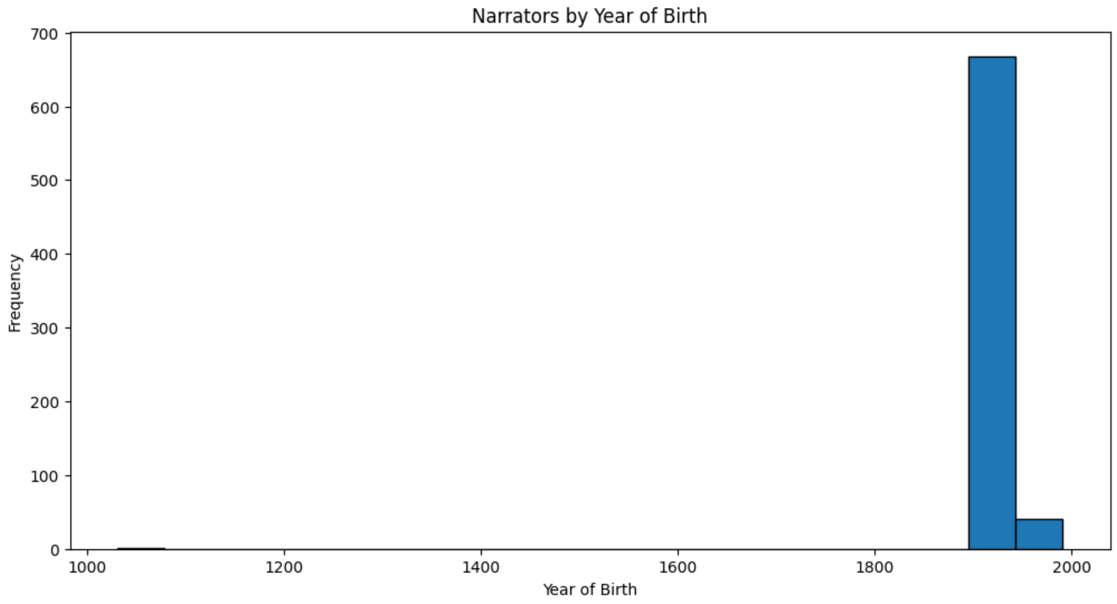


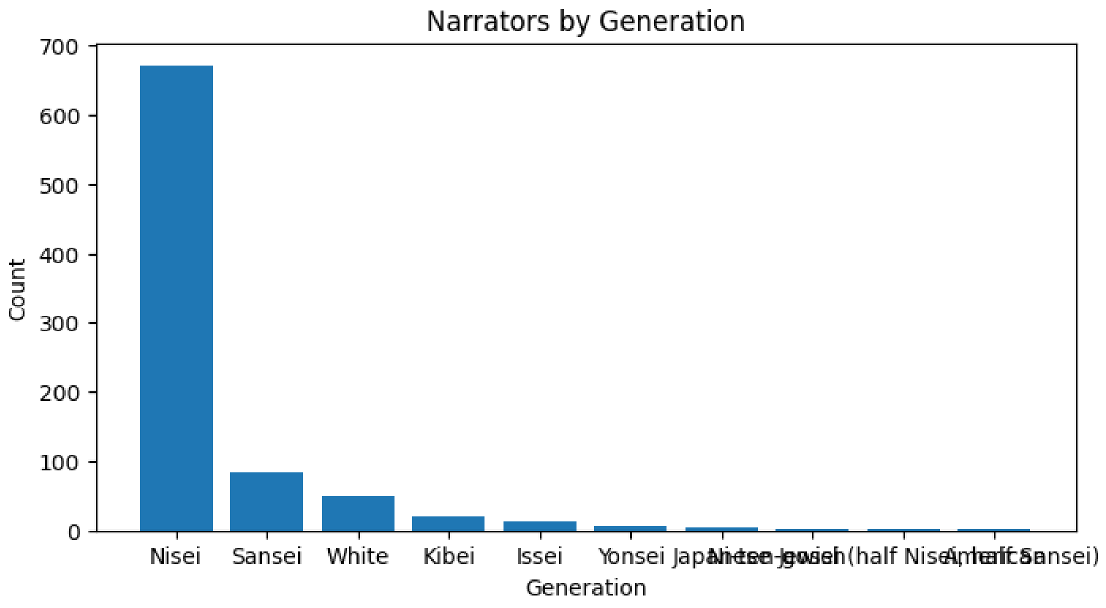
Question 6 (20 pt). Below is a dataset we collected from this website: https://ddr.densho.org/narrators/?page=1. Narrators are the interview subjects of oral histories contained in the Densho Digital Repository. The interviewees, or narrators, share their life histories to preserve history, educate the public, and promote tolerance. We urge our users to approach these materials in the same spirit. You are required to conduct the exploratory data analysis on the location, year of born, generation, and gender. Please select the best visualiztions to present your results. Download the data from here: https://github.com/unt-iialab/info5502-spring2022/blob/main/datasets/Combined-data.xlsx

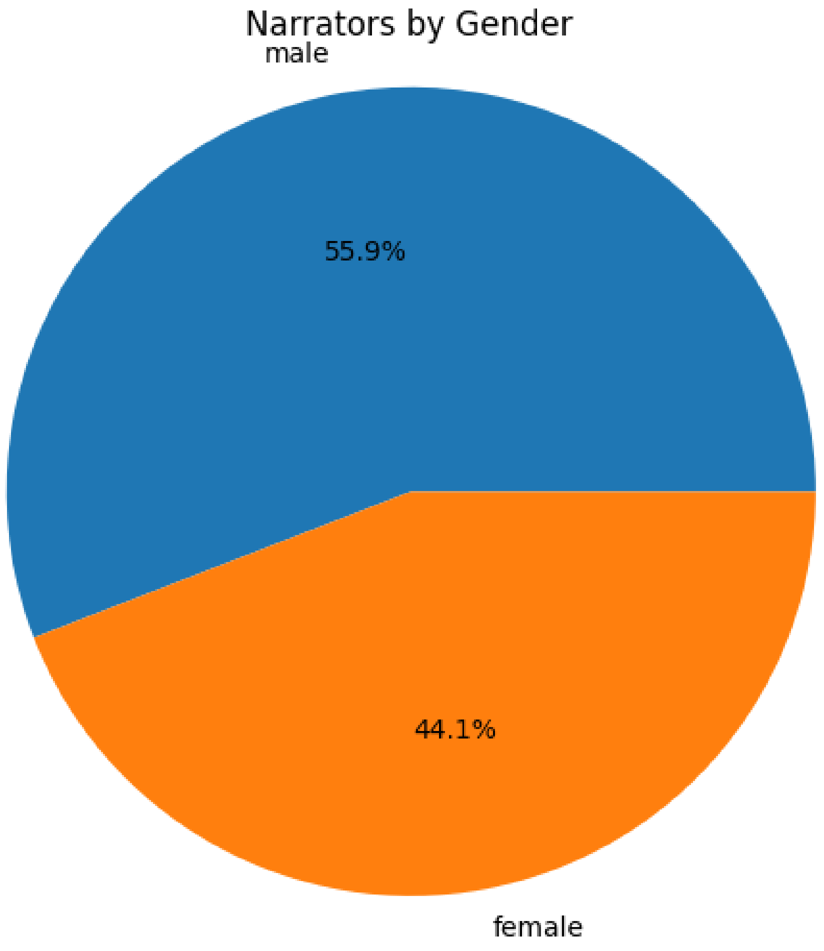
[15]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Read the dataset
data = pd.read_excel('Combined-data.xlsx')
```

```
# Location
location_counts = data['Location'].value_counts().reset_index()
location_counts.columns = ['Location', 'Count']
plt.figure(figsize=(12, 6))
plt.bar(location_counts['Location'], location_counts['Count'])
plt.title('Narrators by Location')
plt.xticks(rotation=45)
plt.xlabel('Location')
plt.ylabel('Count')
plt.show()

# Year of birth
plt.figure(figsize=(12, 6))
plt.hist(data['Year'], bins=20, edgecolor='black')
plt.title('Narrators by Year of Birth')
plt.xlabel('Year of Birth')
plt.ylabel('Frequency')
plt.show()

# Generation
generation_counts = data['Generation'].value_counts().reset_index()
generation_counts.columns = ['Generation', 'Count']
plt.figure(figsize=(8, 4))
plt.bar(generation_counts['Generation'], generation_counts['Count'])
plt.title('Narrators by Generation')
plt.xlabel('Generation')
plt.ylabel('Count')
plt.show()

# Gender
gender_counts = data['Gender'].value_counts().reset_index()
gender_counts.columns = ['Gender', 'Count']
plt.figure(figsize=(6, 6))
plt.pie(gender_counts['Count'], labels=gender_counts['Gender'], autopct='%1.
  ↪1f%%')
plt.title('Narrators by Gender')
plt.axis('equal')
plt.show()
```


Narrators by Location
Count
400
350
300
250
200
150
100
50
0
California
Washington
Oregon
Hawaii
Japan
Illinois
Colorado
British
Texas
Idaho
Nebraska
Wyoming
Maryland
Kauai
Nevada
Alaska
Minnesota
Virginia
Okinawa
Nisei
Missouri
Saipan
Minneapolis
Manilyn
Arizona
Berkeley
Montana
Florida
Issei
Seattle
Defense
Delaware
Mississippi
Verification
Georgia
University
Fukuoka
Chinese
Santa
California
Iwate
Massachusetts
India
Kansas
South
Alberta
Indiana
Location


Narrators by Year of Birth
Frequency
700
600
500
400
300
200
100
0
1000
1200
1400
1600
1800
2000
Year of Birth


Narrators by Generation
Count
700
600
500
400
300
200
100
0
Nisei
Sansei
White
Kibei
Issei
Yonsei
Generation


Narrators by Gender
male
55.9%
44.1%
female

Question 7. Monkeys Typing Shakespeare A monkey is banging repeatedly on the keys of a typewriter. Each time, the monkey is equally likely to hit any of the 26 lowercase letters of the English alphabet, 26 uppercase letters of the English alphabet, and any number between 0-9 (inclusive), regardless of what it has hit before. There are no other keys on the keyboard.

This question is inspired by a mathematical theorem called the Infinite monkey theorem (`https://en.wikipedia.org/wiki/Infinite_monkey_theorem`), which postulates that if you put a monkey in the situation described above for an infinite time, they will eventually type out all of Shakespeare's works.

Question 7-1 (10 pt). Suppose the monkey hits the keyboard 5 times. Compute the chance that the monkey types the sequence `Data8`. (Call this `data_chance`.) Use algebra and type in an arithmetic equation that Python can evalute.

[16]:
```python
# Probability of hitting 'D' on the first keystroke
p_D = 1/62

# Probability of hitting 'a' on the second keystroke
p_a1 = 1/62

# Probability of hitting 't' on the third keystroke
p_t = 1/62

# Probability of hitting 'a' on the fourth keystroke
p_a2 = 1/62

# Probability of hitting '8' on the fifth keystroke
p_8 = 1/62

# Probability of typing the sequence 'Data8' in 5 keystrokes
data_chance = p_D * p_a1 * p_t * p_a2 * p_8
```

Question 7-2 (15 pt). Write a function called `simulate_key_strike`. It should take **no arguments**, and it should return a random one-character string that is equally likely to be any of the 26 lower-case English letters, 26 upper-case English letters, or any number between 0-9 (inclusive).

[17]:
```python
import random

def simulate_key_strike():
    # Define the pool of characters to choose from
    char_pool = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789'

    # Randomly choose a character from the pool
    random_char = random.choice(char_pool)

    # Return the random character
    return random_char
```

Question 7-3 (15 pt). Write a function called `simulate_several_key_strikes`. It should take one argument: an integer specifying the number of key strikes to simulate. It should return a string containing that many characters, each one obtained from simulating a key strike by the monkey.

Hint: If you make a list or array of the simulated key strikes called `key_strikes_array`, you can convert that to a string by calling `"".join(key_strikes_array)`

[18]:
```
import random
def simulate_several_key_strikes( num_strikes):
    # Create a list to store the key strikes
    key_strikes_array = []

    # simulate the key strikes
    for i in range(num_strikes):
        # Simulate a key strike
        key_strike = chr(random.randint(32, 126))
        # add to list
        key_strikes_array.append(key_strike)

    # return string
    return "".join(key_strikes_array)
```