M50 Homework 1

Alex Craig

Exercise 1.

Part A.

Suppose that Y $Bernoulli(q)$ and let $Z = \frac{1}{1+Y} + Y$. What is the probability function of Z?

Solution

We can start with the probability mass function of Y, defined at the probability mass function of any Bernoulli random variable with parameter q:

$$f_Y(y) = \mathbb{P}(Y = y) = \begin{cases} q & \text{if } y = 1 \\ 1 - q & \text{if } y = 0 \\ 0 & \text{o.w.} \end{cases}$$

We can then use this to find the probability mass function of Z by defining the two cases of when $Y = 1$ and $Y = 0$. When $Y = 1$, we now that $Z = \frac{1}{1+1} + 1 = \frac{3}{2}$. When $Y = 0$, we know that $Z = \frac{1}{1+0} + 0 = 1$. Thus, we can define the probability mass function of Z as:

$$f_Z(z) = \mathbb{P}(Z = z) = \begin{cases} q & \text{if } z = \frac{3}{2} \\ 1 - q & \text{if } z = 1 \\ 0 & \text{o.w.} \end{cases}$$

Part B.

Suppose a coin is flipped. If the coin is heads, we write down 0. If the coin is tails, we roll a dice and write down the number. Write down the probability distribution for Y, the number that we write down.

Solution

Let us start by defining random variables X and Z. Let X be the random variable that represents the result of the coin flip (let 0 denote heads, and 1 denote tails), and let Z be the random variable that represents the dice roll. We know that $X \sim bernoulli(\frac{1}{2})$, and we know that Z is a discrete uniform random variable that can take on values in the set $\{1, 2, 3, 4, 5, 6\}$, each with probability $\frac{1}{6}$ given that we flip tails on the initial coin flip. We can therefore define the probability mass function of X amd Z as:

$$f_X(x) = \mathbb{P}(X = x) = \begin{cases} \frac{1}{2} & \text{if } x = 0 \\ \frac{1}{2} & \text{if } x = 1 \\ 0 & \text{o.w.} \end{cases}$$

$$f_Z(z) = \mathbb{P}(Z = z) = \begin{cases} \frac{1}{6} & \text{if } z \in \{1, 2, 3, 4, 5, 6\} \wedge X = 1 \\ 0 & \text{o.w.} \end{cases}$$

Let us now define our sample space S for the outcomes of Y:

$$S = \{0, 1, 2, 3, 4, 5, 6\}$$

$Y = 0$ in the case where we flip heads on the initial coin flip. Therefore:

$$f_Y(0) = \mathbb{P}(Y = 0) = \mathbb{P}(X = 0) = \frac{1}{2}$$

For all other values of Y in the sample space, we know that we must have flipped tails on the initial coin flip, and then rolled that value on the dice. We can therefore define the probability of Y being these values as the probability of the intersection of flipping tails on the initial coin flip and flipping that value on the dice roll:

$$f_Y(z) = \mathbb{P}(Y = z \cap X = 1) = \mathbb{P}(Z = z \cap X = 1) = \mathbb{P}(Z = z \mid X = 1) \times \mathbb{P}(X = 1)$$
$$= \frac{1}{6} \times \frac{1}{2} = \frac{1}{12} : z \in \{1, 2, 3, 4, 5, 6\}$$

We can therefore define the probability mass function of Y as:

$$f_Y(y) = \mathbb{P}(Y = y) = \begin{cases} \frac{1}{2} & \text{if } y = 0 \\ \frac{1}{12} & \text{if } y \in \{1, 2, 3, 4, 5, 6\} \\ 0 & \text{o.w.} \end{cases}$$

Part C.

For the previous problem, conditioned on the dice rolling a 4, what is the probability we write down 0? Conditioned on the coin being tails, what is the probability the dice rolls a 3?

Solution

Given that the dice rolled a 4, we know that $Y = 4$ because in all the cases where we roll a dice, we write down the value that we roll. Therefore, the probability that we write down 0 is 0.

Given that the coin is tails, we know that $X = 1$. We may still define the probability that the dice rolls a 3 as the probability of the intersection of the dice rolling a 3 and the coin being tails, but in this case $\mathbb{P}(X = 1) = 1$:

$$\mathbb{P}(Z = 3 \cap X = 1) = \mathbb{P}(Z = 3 \mid X = 1) \times \mathbb{P}(X = 1) = \frac{1}{6} \times 1 = \frac{1}{6}$$

Part D.

Consider the geometric distribution discussed in lecture. What are 3 examples of variables in the real world for which this might be a good model and what are some limitations of these models.

Solution

Examples:

1. Shooting basketball free throws until you make one.
2. Playing lottery scratch tickets until you win.
3. Flying on an airplane until you get into a fatal crash.

Limitations:

1. The probability of success must be constant for each trial, so changes in the probability of success over time are not accounted for.
2. The trials must be independent, so the outcome of one trial cannot affect the outcome of another trial.

Exercise 2.

Consider the following code:

```
for i in range(5):
    for j in range(i + 1):
        print(i, end = "")
    print("")
```

```
0
11
222
3333
44444
```

Modify the code so that it prints out:

```
0
01
012
0123
01234
012345
```

Solution

```
for i in range(6):
    for j in range(i + 1):
        print(j, end = "")
    print("")
```

```
0
01
012
0123
01234
012345
```

Exercise 3.

(Washington post data): Below I load some data on homocide victims in US from the washington post. Don't worry about how I process it, all you need to work with is the DataFrame "data" on the very last line.

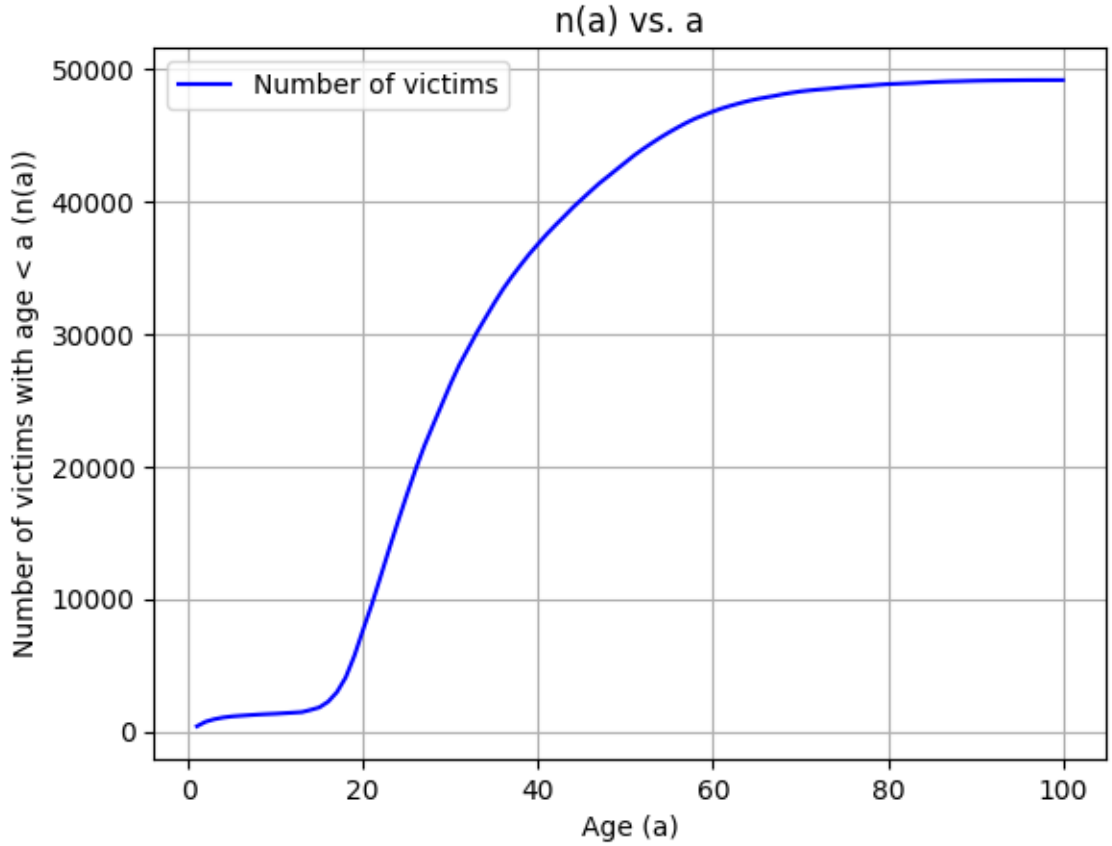
```
import pandas as pd

washington_data = pd.read_csv("https://raw.githubusercontent.com/washingtonpost/data-homicides/master/homicide

washington_data["victim_age"] = pd.to_numeric(washington_data["victim_age"], errors="coerce")
```

Part A.

For each age $a = 1, ..., 100$ determine the number of victims $n(a)$ with an age $< a$. You can ignore the effects of those entries with missing ages. Make a plot of $n(a)$ vs. a.

Solution

```
import matplotlib.pyplot as plt

# Function to calculate number of victims for a given age range and data
def calculate_victims(df):
    n_values = []
    for a in range(1, 101):
        n = df[df["victim_age"] < a].shape[0]
        n_values.append(n)
    return n_values

# Create an array to store number of victims for each age a
all_n_values = calculate_victims(washington_data)

# Plotting the results
plt.plot(range(1, 101), all_n_values, label='Number of victims', color='blue')
plt.xlabel('Age (a)')
plt.ylabel('Number of victims with age < a (n(a))')
plt.title('n(a) vs. a')
plt.legend()
plt.grid(True)
plt.show()
```



Part B.

What do you notice about the plot. Does it have the expected behavior?

Solution

There are very few homicide victims under the age of 18, but the number of homicide victims increases most rapidly between the ages of 20 and 40. This makes sense because this is about the age range where people are most likely to be involved in gang violence or other violent crime, which is a common cause of homicide.

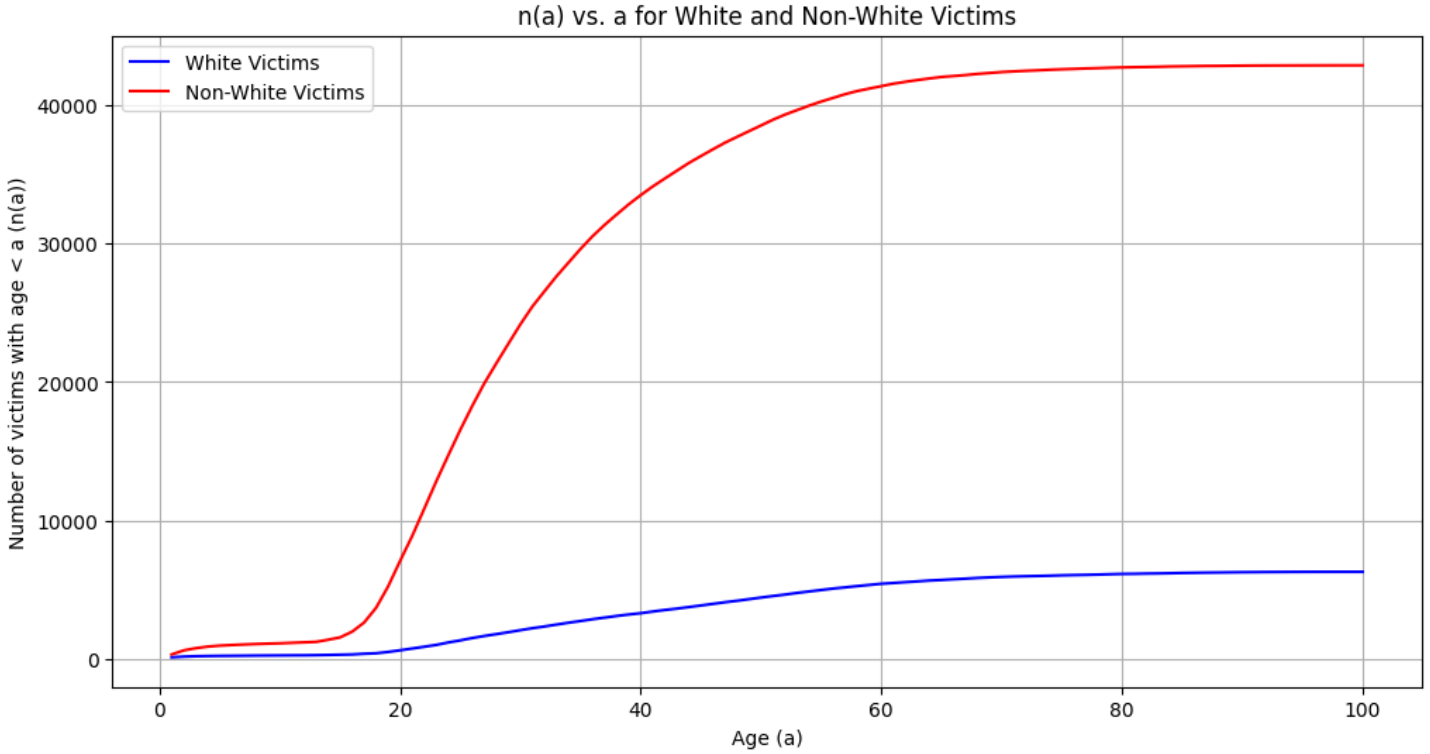
Part C.

Break the data up into white and non-white victims. Then, for each group make the plot from part **(a)**. Comment on what you find.

Solution

```
# Filter data for white victims and non-white victims
white_victims = washington_data[washington_data["victim_race"] == "White"]
non_white_victims = washington_data[washington_data["victim_race"] != "White"]

# Calculate for white and non-white victims
white_n_values = calculate_victims(white_victims)
non_white_n_values = calculate_victims(non_white_victims)

# Plotting the results
plt.figure(figsize=(12,6))

plt.plot(range(1, 101), white_n_values, label='White Victims', color='blue')
plt.plot(range(1, 101), non_white_n_values, label='Non-White Victims', color='red')
plt.xlabel('Age (a)')
plt.ylabel('Number of victims with age < a (n(a))')
plt.title('n(a) vs. a for White and Non-White Victims')
```

```
plt.grid(True)
plt.legend()
plt.show()
```



The results between White and Non-White victims are quite different. The Non-White victims look very similar to the graph we previously saw, where there is a steep increase in homicide victims between the ages of 20 and 40, and a less steep increase between 40 and 60. The White victims, however, see a fairly consistent increase in homicide victims between the ages of 20 and 60, and then very few more homicide victims after that.

Exercise 4.

(Getting a sequence of wins): Let J denote a random variable representing the number of times a fair coin is flipped before two heads appear in a row. As we saw in class, the following code generates simulations of J:

```
import numpy as np

def flip_until_two():
    num_heads = 0
    total_flips = 0
    while num_heads < 2:
        y = np.random.choice([0,1])
        if y == 0:
            num_heads = 0
        else:
            num_heads = num_heads + 1
        total_flips = total_flips + 1
    return total_flips
```

Part A.

By changing the code above, write a function `roll_until_n()` that rolls a dice until we get n ones in a row You should change the variable names accordingly. We will call this random variable `Roll_Count(n)`.

Solution

```
def roll_until_n(n: int):
    num_ones = 0
```

```
    total_rolls = 0
    while num_ones < n:
        y = np.random.choice([1,2,3,4,5,6])
        if y != 1:
            num_ones = 0
        else:
            num_ones = num_ones + 1
        total_rolls = total_rolls + 1
    return total_rolls

Roll_Count = roll_until_n(2)
print(Roll_Count)
```

```
72
```

Part B.

Make a DataFrame where each column represents a value of n from 1 to 6 and each row is a simulation from the model `Roll_Count(n)`. There should be 100 rows.

Solution

```
# Simulation
num_simulations = 100
n_values = [1,2,3,4,5,6]  # for n from 1 to 6

# Create an empty DataFrame with columns from 1 to 6
df = pd.DataFrame(columns=n_values)

# For 100 simulations
for i in range(num_simulations):
    # Calculate the number of rolls for each n
    row_data = [roll_until_n(n) for n in n_values]
    # Add the row to the DataFrame
    df.loc[i] = row_data

print(df)
```

```
     1    2    3     4      5       6
0    4   24  524  1081   4995   14963
1   10   11  297   520   6279   86559
2    1    3  137  3448   2665   55409
3    5   21   10  1492    711  169048
4    8   18   22  2807   9483  152176
..  ..  ...  ...   ...    ...     ...
95   2  101  222  1096  15624    8605
96   4   52  206   127    457   18044
97  21   41   78   370   4767   73735
98   2   57  635   336  24683   51014
99   8   35  275   638  12027     202

[100 rows x 6 columns]
```

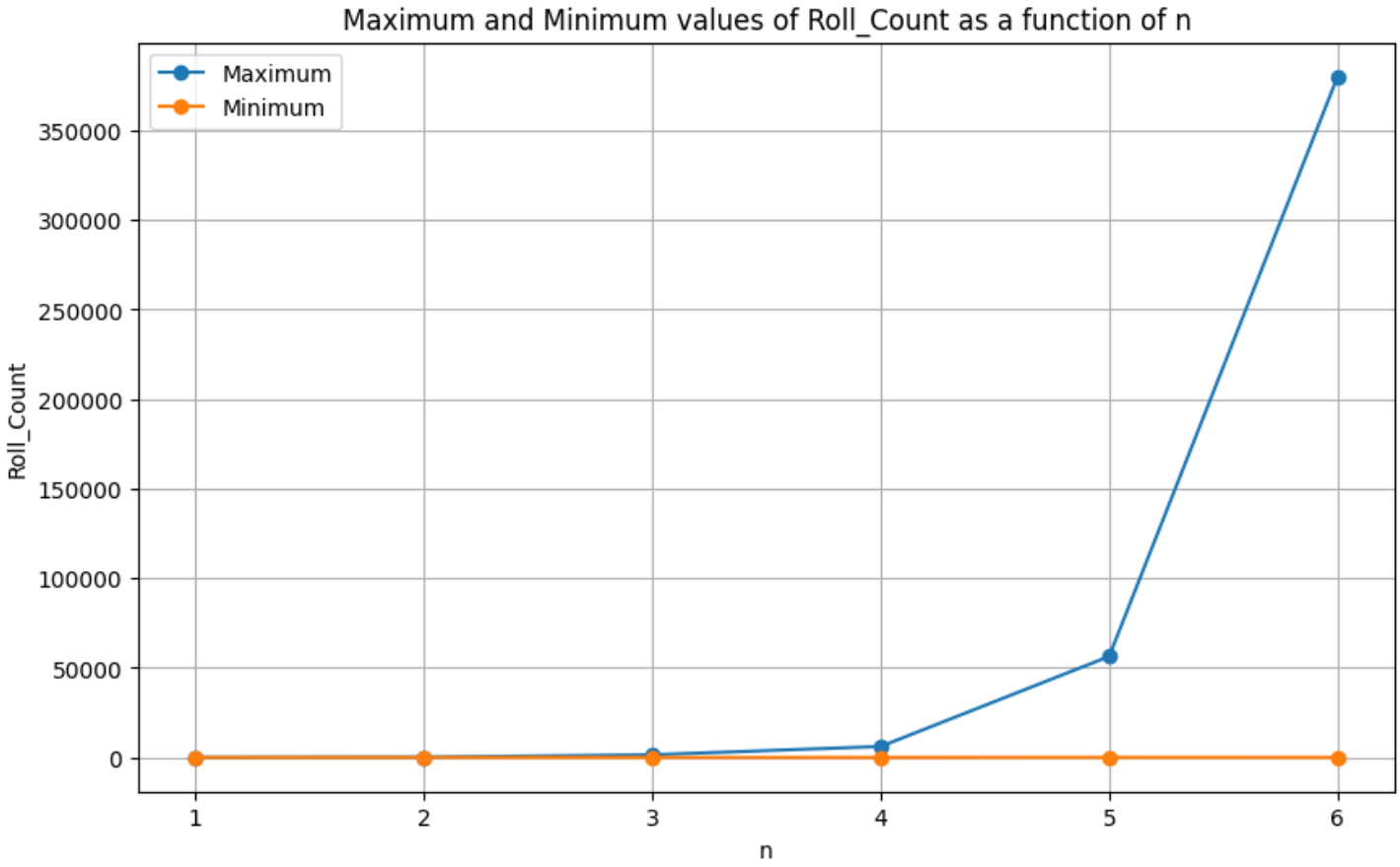
Part C.

Make a plot of the maximum and minimum values of J as a function of n on the same plot. You might notice one of these increases much faster than the other.

Solution

```
# Extracting max and min values for each n
max_vals = df.max()
```

```
min_vals = df.min()

# Plotting
plt.figure(figsize=(10,6))
plt.plot(n_values, max_vals, '-o', label='Maximum')
plt.plot(n_values, min_vals, '-o', label='Minimum')
plt.xlabel('n')
plt.ylabel('Roll_Count')
plt.title('Maximum and Minimum values of Roll_Count as a function of n')
plt.legend()
plt.grid(True)
plt.show()
```



Exercise 5.

(Two gene model): Consider the variant of the model discussed in class

$$\mathbb{P}(Y_A, Y_B) = \begin{cases} \frac{1}{3} \text{ if } Y_A = 0 \wedge Y_B = 0 \\ \frac{1}{3} \text{ if } Y_A = 0 \wedge Y_B = 1 \\ \frac{1}{6} \text{ if } Y_A = 1 \wedge Y_B = 0 \\ \frac{1}{6} \text{ if } Y_A = 1 \wedge Y_B = 1 \\ 0 \text{ o.w.} \end{cases}$$

Part A.

What are the marginal distributions of Y_A and Y_B?

Solution

We can find the marginal distribution of Y_A by summing over all possible values of Y_B:

Marginal distribution of Y_A:

$$\mathbb{P}(Y_A = 0) = \mathbb{P}(Y_A = 0 \mid Y_B = 0) + \mathbb{P}(Y_A = 0 \mid Y_B = 1) = \frac{1}{3} + \frac{1}{3} = \frac{2}{3}$$

$$\mathbb{P}(Y_A = 1) = \mathbb{P}(Y_A = 1 \mid Y_B = 0) + \mathbb{P}(Y_A = 1 \mid Y_B = 1) = \frac{1}{6} + \frac{1}{6} = \frac{1}{3}$$

Marginal distribution of Y_B:

$$\mathbb{P}(Y_B = 0) = \mathbb{P}(Y_B = 0 \mid Y_A = 0) + \mathbb{P}(Y_B = 0 \mid Y_A = 1) = \frac{1}{3} + \frac{1}{6} = \frac{1}{2}$$

$$\mathbb{P}(Y_B = 1) = \mathbb{P}(Y_B = 1 \mid Y_A = 0) + \mathbb{P}(Y_B = 1 \mid Y_A = 1) = \frac{1}{3} + \frac{1}{6} = \frac{1}{2}$$

Part B.

Are Y_A and Y_B independent?

Solution

If Y_A and Y_B are independent, then $\mathbb{P}(Y_A = y_A \mid Y_B = y_B) = \mathbb{P}(Y_A = y_A)$ and $\mathbb{P}(Y_B = y_B \mid Y_A = y_A) = \mathbb{P}(Y_B = y_B)$ for all values of y_A and y_B. We can check this for all possible values of y_A and y_B:

Checking that information about Y_B does not affect the distribution of Y_A:

$$\mathbb{P}(Y_A = 0 \mid Y_B = 0) = \frac{\mathbb{P}(Y_A = 0 \cap Y_B = 0)}{\mathbb{P}(Y_B = 0)} = \frac{\frac{1}{3}}{\frac{1}{2}} = \frac{2}{3} = \mathbb{P}(Y_A = 0) \quad \checkmark$$

$$\mathbb{P}(Y_A = 0 \mid Y_B = 1) = \frac{\mathbb{P}(Y_A = 0 \cap Y_B = 1)}{\mathbb{P}(Y_B = 1)} = \frac{\frac{1}{3}}{\frac{1}{2}} = \frac{2}{3} = \mathbb{P}(Y_A = 0) \quad \checkmark$$

$$\mathbb{P}(Y_A = 1 \mid Y_B = 0) = \frac{\mathbb{P}(Y_A = 1 \cap Y_B = 0)}{\mathbb{P}(Y_B = 0)} = \frac{\frac{1}{6}}{\frac{1}{2}} = \frac{1}{3} = \mathbb{P}(Y_A = 1) \quad \checkmark$$

$$\mathbb{P}(Y_A = 1 \mid Y_B = 1) = \frac{\mathbb{P}(Y_A = 1 \cap Y_B = 1)}{\mathbb{P}(Y_B = 1)} = \frac{\frac{1}{6}}{\frac{1}{2}} = \frac{1}{3} = \mathbb{P}(Y_A = 1) \quad \checkmark$$

Checking that information about Y_A does not affect the distribution of Y_B:

$$\mathbb{P}(Y_B = 0 \mid Y_A = 0) = \frac{\mathbb{P}(Y_B = 0 \cap Y_A = 0)}{\mathbb{P}(Y_A = 0)} = \frac{\frac{1}{3}}{\frac{2}{3}} = \frac{1}{2} = \mathbb{P}(Y_B = 0) \quad \checkmark$$

$$\mathbb{P}(Y_B = 0 \mid Y_A = 1) = \frac{\mathbb{P}(Y_B = 0 \cap Y_A = 1)}{\mathbb{P}(Y_A = 1)} = \frac{\frac{1}{6}}{\frac{1}{3}} = \frac{1}{2} = \mathbb{P}(Y_B = 0) \quad \checkmark$$

$$\mathbb{P}(Y_B = 1 \mid Y_A = 0) = \frac{\mathbb{P}(Y_B = 1 \cap Y_A = 0)}{\mathbb{P}(Y_A = 0)} = \frac{\frac{1}{3}}{\frac{2}{3}} = \frac{1}{2} = \mathbb{P}(Y_B = 1) \quad \checkmark$$

$$\mathbb{P}(Y_B = 1 \mid Y_A = 1) = \frac{\mathbb{P}(Y_B = 1 \cap Y_A = 1)}{\mathbb{P}(Y_A = 1)} = \frac{\frac{1}{6}}{\frac{1}{3}} = \frac{1}{2} = \mathbb{P}(Y_B = 1) \quad \checkmark$$

Because $\mathbb{P}(Y_A = y_A \mid Y_B = y_B) = \mathbb{P}(Y_A = y_A)$ and $\mathbb{P}(Y_B = y_B \mid Y_A = y_A) = \mathbb{P}(Y_B = y_B)$ for all values of y_A and y_B, Y_A and Y_B are independent.

Part C.

Confirm your answer with simulations.

Solution

```
# Number of samples
n_samples = 1000000

# Simulating from the joint distribution
samples = np.random.choice(['00', '01', '10', '11'], size=n_samples, p=[1/3, 1/3, 1/6, 1/6])
y_a_samples = [int(sample[0]) for sample in samples]
y_b_samples = [int(sample[1]) for sample in samples]

# Estimating the marginal distributions
p_y_a_0 = y_a_samples.count(0) / n_samples
p_y_a_1 = y_a_samples.count(1) / n_samples
p_y_b_0 = y_b_samples.count(0) / n_samples
p_y_b_1 = y_b_samples.count(1) / n_samples

# Estimating the conditional distributions
p_y_a_0_given_y_b_0 = sum([y_a == 0 for y_a, y_b in
                           zip(y_a_samples, y_b_samples) if y_b == 0]) / y_b_samples.count(0)
p_y_a_0_given_y_b_1 = sum([y_a == 0 for y_a, y_b in
                           zip(y_a_samples, y_b_samples) if y_b == 1]) / y_b_samples.count(1)
p_y_a_1_given_y_b_0 = sum([y_a == 1 for y_a, y_b in
                           zip(y_a_samples, y_b_samples) if y_b == 0]) / y_b_samples.count(0)
p_y_a_1_given_y_b_1 = sum([y_a == 1 for y_a, y_b in
                           zip(y_a_samples, y_b_samples) if y_b == 1]) / y_b_samples.count(1)

p_y_b_0_given_y_a_0 = sum([y_b == 0 for y_a, y_b in
                           zip(y_a_samples, y_b_samples) if y_a == 0]) / y_a_samples.count(0)
p_y_b_0_given_y_a_1 = sum([y_b == 0 for y_a, y_b in
                           zip(y_a_samples, y_b_samples) if y_a == 1]) / y_a_samples.count(1)
p_y_b_1_given_y_a_0 = sum([y_b == 1 for y_a, y_b in
                           zip(y_a_samples, y_b_samples) if y_a == 0]) / y_a_samples.count(0)
p_y_b_1_given_y_a_1 = sum([y_b == 1 for y_a, y_b in
                           zip(y_a_samples, y_b_samples) if y_a == 1]) / y_a_samples.count(1)

# Displaying the results
print(f"P(Y_A = 0) = {p_y_a_0:.4f}")
print(f"\tP(Y_A = 0 | Y_B = 0) = {p_y_a_0_given_y_b_0:.4f}")
print(f"\tP(Y_A = 0 | Y_B = 1) = {p_y_a_0_given_y_b_1:.4f}")

print(f"\nP(Y_A = 1) = {p_y_a_1:.4f}")
print(f"\tP(Y_A = 1 | Y_B = 0) = {p_y_a_1_given_y_b_0:.4f}")
print(f"\tP(Y_A = 1 | Y_B = 1) = {p_y_a_1_given_y_b_1:.4f}")

print(f"\nP(Y_B = 0) = {p_y_b_0:.4f}")
print(f"\tP(Y_B = 0 | Y_A = 0) = {p_y_b_0_given_y_a_0:.4f}")
print(f"\tP(Y_B = 0 | Y_A = 1) = {p_y_b_0_given_y_a_1:.4f}")

print(f"\nP(Y_B = 1) = {p_y_b_1:.4f}")
print(f"\tP(Y_B = 1 | Y_A = 0) = {p_y_b_1_given_y_a_0:.4f}")
print(f"\tP(Y_B = 1 | Y_A = 1) = {p_y_b_1_given_y_a_1:.4f}")
```

```
P(Y_A = 0) = 0.6667
P(Y_A = 0 | Y_B = 0) = 0.6662
P(Y_A = 0 | Y_B = 1) = 0.6673

P(Y_A = 1) = 0.3333
P(Y_A = 1 | Y_B = 0) = 0.3338
P(Y_A = 1 | Y_B = 1) = 0.3327

P(Y_B = 0) = 0.5001
```

```
P(Y_B = 0 | Y_A = 0) = 0.4997
P(Y_B = 0 | Y_A = 1) = 0.5010

P(Y_B = 1) = 0.4999
P(Y_B = 1 | Y_A = 0) = 0.5003
P(Y_B = 1 | Y_A = 1) = 0.4990
```

Exercise 6.

(Verifying variance formula for Bernoulli variable): Verify the formula for the variance:

$$Var(Y) = q(1-q) : Y \sim Bernoulli(q)$$

Remember, you can do this you can use the fact that pointwise arithmetic between numpy arrays can be performed directly on the ways, e.g. `q_range * q_range` makes a list where every element is the corresponding element of `qrange` squared. You should experiment to ensure you are using enough samples.

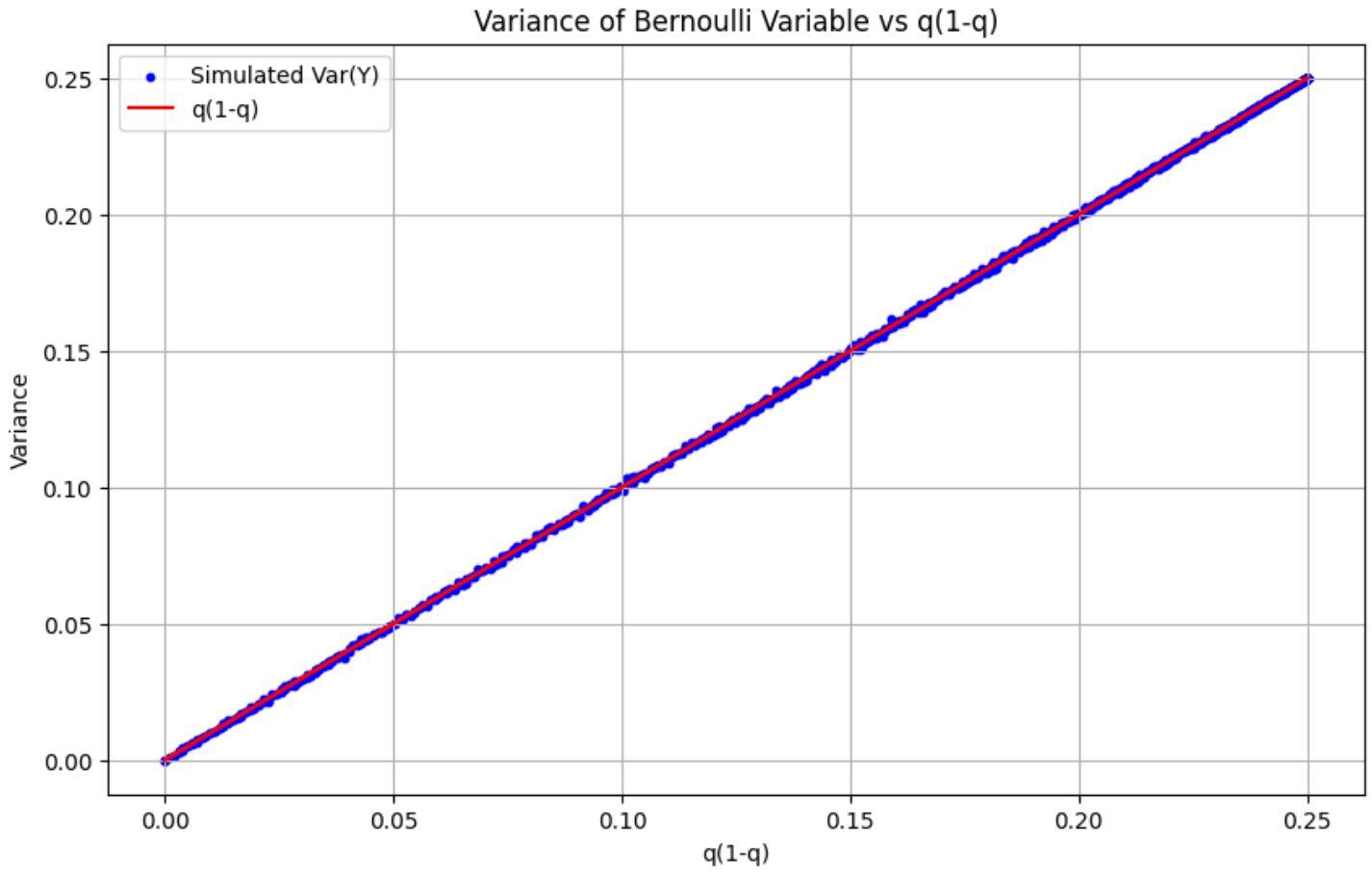
Solution

Remember that the variance of a random variable Y is defined as:

$$Var(Y) = \mathbb{E}[(Y - \mathbb{E}[Y])^2] = \mathbb{E}[Y^2] - \mathbb{E}[Y]^2$$

Let us simulate $Var(Y)$ for 100000 trials for each value of q in the range $[0, 1]$ with steps of 0.01 and plot the results:

```python
num_simulations = 100000

results = np.zeros((1001, 2))
# Epsilon to fix floating point errors
epsilon = 1e-10

for q in np.arange(0, 1.01, 0.001):
    q = min(max(q, epsilon), 1 - epsilon)
    y_sim = np.random.choice([0, 1], size=num_simulations, p=[1-q, q])
    y_sim_squared = y_sim ** 2

    y_mean = np.mean(y_sim)
    y_squared_mean = np.mean(y_sim_squared)

    var_y = y_squared_mean - y_mean ** 2

    results[int(q * 1000), 0] = q * (1 - q)  # Store q(1-q) here
    results[int(q * 1000), 1] = var_y

# Plotting the results:
plt.figure(figsize=(10, 6))
plt.scatter(results[:, 0], results[:, 1], c='blue', label='Simulated Var(Y)', s=10)
plt.plot(results[:, 0], results[:, 0], 'r-', label='q(1-q)')
plt.title('Variance of Bernoulli Variable vs q(1-q)')
plt.xlabel('q(1-q)')
plt.ylabel('Variance')
plt.legend()
plt.grid(True)
plt.show()
```



Exercise 7.

(Working with Washington Post Data): This a continuation of Exercise 3. Consider the quantities:

$$\mathbb{P}(\text{age} < z)$$

$$\mathbb{P}(\text{age} < z \mid \text{white})$$

$$\mathbb{P}(\text{age} < z \mid \text{not white})$$

Part A.

Explain who each of these are related to the plot you made in Exercise 3.

Solution

$\mathbb{P}(\text{age} < z)$ is related to the first plot where we simply plot the number of victims with an age less than a, given by $n(a)$. $\mathbb{P}(\text{age} < z)$ is simply specifying to divide $n(a)$ by the total number of victims in order to give a proportion of victims with an age less than a or in this case z.

$\mathbb{P}(\text{age} < z \mid \text{white})$ is related to the second plot where we plot the number of white victims with an age less than a, given by $n_{white}(a)$. $\mathbb{P}(\text{age} < z \mid \text{white})$ is simply specifying to divide $n_{white}(a)$ by the total number of white victims in order to give a proportion of white victims with an age less than a or in this case z.

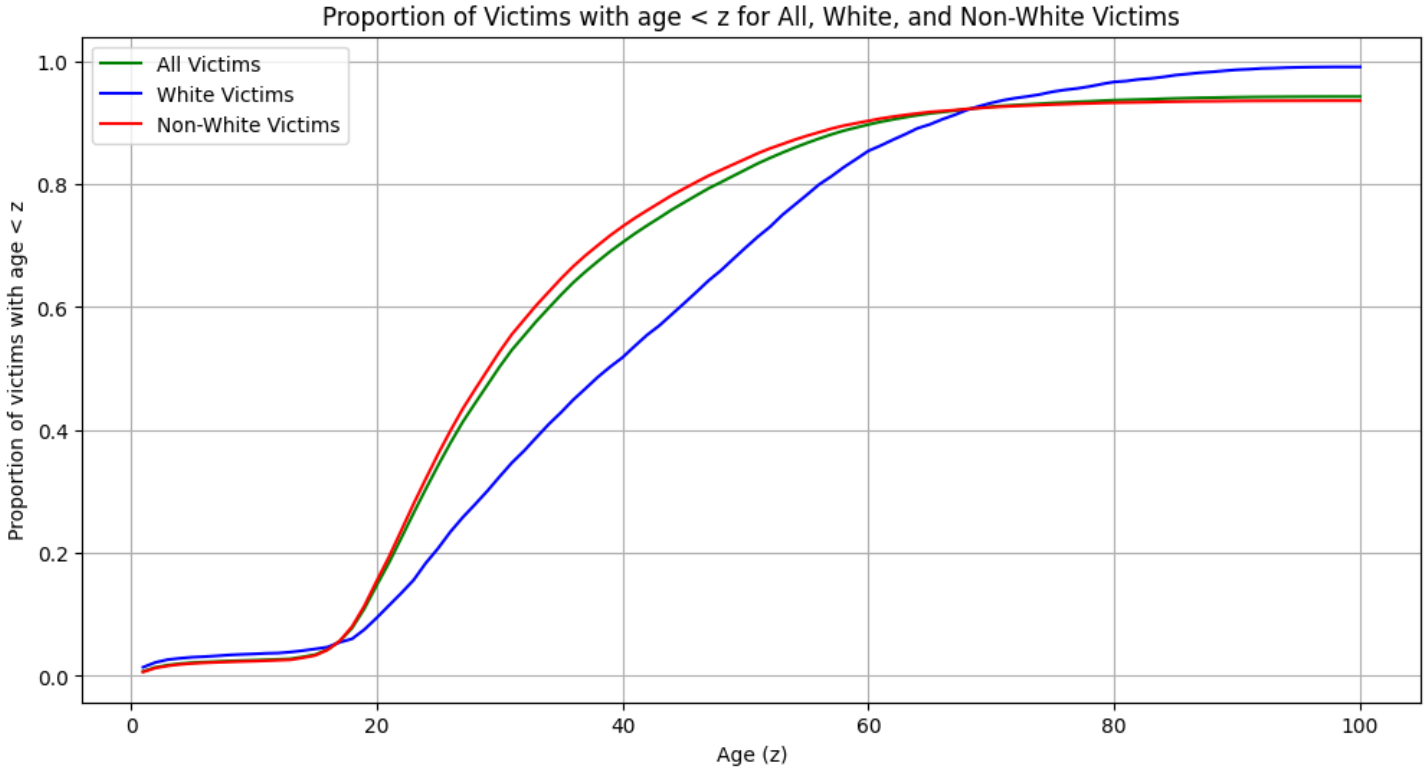
$\mathbb{P}(\text{age} < z \mid \text{not white})$ is related to the third plot where we plot the number of non-white victims with an age less than a, given by $n_{non-white}(a)$. $\mathbb{P}(\text{age} < z \mid \text{not white})$ is simply specifying to divide $n_{non-white}(a)$ by the total number of non-white victims in order to give a proportion of non-white victims with an age less than a or in this case z.

Part B.

Make plots of them and comment of the difference between the plot in Exercise 3. Do you think age and race are independent based on these plots?

Solution

```
# Convert lists to numpy arrays
all_n_values_array = np.array(all_n_values)
white_n_values_array = np.array(white_n_values)
non_white_n_values_array = np.array(non_white_n_values)

# Calculate proportions for all, white, and non-white victims
proportion_values = all_n_values_array / washington_data.shape[0]
white_proportion_values = white_n_values_array / white_victims.shape[0]
non_white_proportion_values = non_white_n_values_array / non_white_victims.shape[0]

# Plotting the proportions
plt.figure(figsize=(12,6))

plt.plot(range(1, 101), proportion_values, label='All Victims', color='green')
plt.plot(range(1, 101), white_proportion_values, label='White Victims', color='blue')
plt.plot(range(1, 101), non_white_proportion_values, label='Non-White Victims', color='red')
plt.xlabel('Age (z)')
plt.ylabel('Proportion of victims with age < z')
plt.title('Proportion of Victims with age < z for All, White, and Non-White Victims')
plt.grid(True)
plt.legend()
plt.show()
```



Age and race do not appear to be independent based on this plot, due to the fact that the cumulative density of age for white victims is quite different than both all victims and non-white victims. These means that, given that the victim is white or non-white, the distributions of age are different and so the variables are not independent.

Part C.

Using the data, approximate:

$$\mathbb{P}(\text{white} \mid \text{age} < 60)$$

Hint: One way to do this is to use Bayes' rule

Solution

Remember that Bayes' Theorem states that:

$$\mathbb{P}(A \mid B) = \frac{\mathbb{P}(B \mid A) \times \mathbb{P}(A)}{\mathbb{P}(B)}$$

And through the law of total probability, we know that:

$$\mathbb{P}(B) = \mathbb{P}(B \mid A) \times \mathbb{P}(A) + \mathbb{P}(B \mid A^c) \times \mathbb{P}(A^c)$$

Therefore, we may state that:

$$\mathbb{P}(\text{white} \mid \text{age} < 60) = \frac{\mathbb{P}(\text{age} < 60 \mid \text{white}) \times \mathbb{P}(\text{white})}{\mathbb{P}(\text{age} < 60)}$$

Let us now calculate each of these terms in the dataset and use them to print out the conditional probability.

```
p_age_60_give_white = white_proportion_values[60]
p_age_60 = proportion_values[60]
p_white = white_victims.shape[0] / washington_data.shape[0]

p_white_given_age_60 = (p_age_60_give_white * p_white) / p_age_60
print(f"P(White | Age = 60) = {p_white_given_age_60:.4f}")
```

```
P(White | Age = 60) = 0.1162
```