DAI Assignment 1

Aayush Borkar, Sandeep Reddy Nallamilli, Yash Sabale

23B0944, 23B1006, 23B1043

Last updated August 26, 2024

Contents

§1. Let's Gamble

A can either win after both of them throwing n throws each, or A can win on the last throw by drawing in the first n throws and getting one more win in the last throw. In the first case, the last throw from A does not matter, and in the latter, A needs to win the last throw.

We define the following events:

$$E_A : \text{A has more wins after } n \text{ throws}$$
$$E_B : \text{B has more wins after } n \text{ throws}$$
$$E_D : \text{A and B have the same number of wins after } n \text{ throws}$$

The probability that A is leading after n throws must be equal to the probability that B is leading after n throws:

$$P(E_A) = P(E_B)$$

Also, we have:

$$P(E_A) + P(E_B) + P(E_D) = 1$$

Thus,

$$P(E_A) = \frac{1 - P(E_D)}{2}$$

To find the probability that A will have more wins, we consider both cases where A could win:
1. A wins after n throws: This is already covered by $P(E_A)$.
2. A wins on the last throw given that both A and B have the same number of wins after n throws: The probability of A winning the last throw is $\frac{1}{2}$, and the probability of ending in a draw after n throws is $P(E_D)$.
Combining these,

$$P(\text{A having more wins}) = P(E_A) + P(\text{A winning on the last throw} \mid E_D) \times P(E_D)$$

$$P(\text{A having more wins}) = \frac{1 - P(E_D)}{2} + \frac{1}{2} \times P(E_D)$$

$$P(\text{A having more wins}) = \frac{1 - P(E_D) + P(E_D)}{2}$$

$$P(\text{A having more wins}) = \frac{1}{2}$$

§2. Two Trading Teams

Before proceeding to solve the problem, we define the following events:

$$W_A : \text{We win when we play a game against team } A$$
$$W_B : \text{We win when we play a game against team } B$$

We win a sequence of games when we win two consecutive games in the sequence. Also, we can choose the sequence in which we play the games out of A-B-A and B-A-B. Given this info, we define three more events:

$$W : \text{We win two consecutive games when we play three games}$$
$$ABA : \text{We play in the order } A\text{-}B\text{-}A$$
$$BAB : \text{We play in the order } B\text{-}A\text{-}B$$

We're given that team B is better than team A, so we can say that we have a higher probability of winning against team A than against team B.

$$P(W_A) > P(W_B) \tag{2.1}$$

Let's analyse our winning probability when we play in the order A-B-A. We can win in two ways:

1. We win against team A and team B, and the third match doesn't matter since we've already gotten 2 consecutive wins. The probability of this happening is:
$$P(W_A)P(W_B) \tag{2.2}$$
2. We lose against team A, win against team B, and win against team A. The probability of this happening is:
$$P(W_A^c)P(W_B)P(W_A) = (1 - P(W_A))P(W_B)P(W_A) \tag{2.3}$$

The total probability of us winning when playing $A - B - A$ is the sum of the probabilities of the above two cases (2.2) and (2.3).

$$\begin{aligned} P(W \mid ABA) &= P(W_A)P(W_B) + (1 - P(W_A))P(W_B)P(W_A) \\ &= P(W_A)P(W_B)(2 - P(W_A)) \end{aligned} \tag{2.4}$$

Similarly, we can calculate the probability of us winning when playing in the order $B - A - B$.

$$\begin{aligned} P(W \mid BAB) &= P(W_B)P(W_A) + (1 - P(W_B))P(W_A)P(W_B) \\ &= P(W_B)P(W_A)(2 - P(W_B)) \end{aligned} \tag{2.5}$$

From (2.1), we know that

$$\begin{gathered} P(W_A) > P(W_B) \\ 2 - P(W_A) < 2 - P(W_B) \\ P(W_A)P(W_B)(2 - P(W_A)) < P(W_B)P(W_A)(2 - P(W_B)) \end{gathered} \tag{2.6}$$

From (2.4), (2.5), and (2.6), we can conclude that

$$P(W \mid ABA) < P(W \mid BAB)$$

$\therefore$ We should play in the order B-A-B as we stand a higher chance of winning that way.

§3. Random Variables

3.1.

We are given the following inequalities:

$$P(Q_1 < q_1) \geq 1 - p_1 \quad \text{and} \quad P(Q_2 < q_2) \geq 1 - p_2$$

We check the complementary probabilities for the events $Q_1 \geq q_1$ and $Q_2 \geq q_2$, we get the following inequalities:

$$P(Q_1 \geq q_1) = P((Q_1 < q_1)^c) = 1 - P(Q_1 < q_1)$$

Given that $P(Q_1 < q_1) \geq 1 - p_1$, we see that:

$$P(Q_1 \geq q_1) \leq p_1$$

Similarly, for Q_2, we have:

$$P(Q_2 \geq q_2) \leq p_2$$

We use the probability of the union of events $Q_1 \geq q_1$ and $Q_2 \geq q_2$:

$$P[(Q_1 \geq q_1) \cup (Q_2 \geq q_2)] + P[(Q_1 \geq q_1) \cap (Q_2 \geq q_2)] = P(Q_1 \geq q_1) + P(Q_2 \geq q_2)$$

Since $P(Q_1 \geq q_1) \leq p_1$ and $P(Q_2 \geq q_2) \leq p_2$, we obtain:

$$P(Q_1 \geq q_1) + P(Q_2 \geq q_2) \leq p_1 + p_2$$

$$P[(Q_1 \geq q_1) \cup (Q_2 \geq q_2)] + P[(Q_1 \geq q_1) \cap (Q_2 \geq q_2)] \leq p_1 + p_2$$

$$P[(Q_1 \geq q_1) \cup (Q_2 \geq q_2)] \leq p_1 + p_2 - P[(Q_1 \geq q_1) \cap (Q_2 \geq q_2)]$$

Using *De Morgan's Law*, $(A \cup B)^c = A^c \cap B^c$, we work with the following complementary event:

$$P[(Q_1 < q_1) \cap (Q_2 < q_2)] = P[((Q_1 \geq q_1) \cup (Q_2 \geq q_2))^c] = 1 - P[(Q_1 \geq q_1) \cup (Q_2 \geq q_2)]$$

Since $P[(Q_1 \geq q_1) \cup (Q_2 \geq q_2)] \geq 0$, we conclude:

$$P[(Q_1 < q_1) \cap (Q_2 < q_2)] \geq 1 - (p_1 + p_2)$$

This gives a lower bound on the probability that both Q_1 and Q_2 are less than q_1 and q_2. Claim:

$$(Q_1 < q_1) \cap (Q_2 < q_2) \subseteq \{Q_1 Q_2 < q_1 q_2\}$$

Let $x \in (Q_1 < q_1) \cap (Q_2 < q_2)$. This means:

$$Q_1(x) < q_1 \quad \text{and} \quad Q_2(x) < q_2$$

Since both $Q_1(x)$ and $Q_2(x)$ are positive, their product is also positive:

$$0 < Q_1(x) Q_2(x) < q_1 q_2$$

Thus, $x \in \{Q_1 Q_2 < q_1 q_2\}$, proving the claim. Using the fact that $(Q_1 < q_1) \cap (Q_2 < q_2) \subseteq \{Q_1 Q_2 < q_1 q_2\}$, we conclude:

$$P(Q_1 Q_2 < q_1 q_2) \geq P[(Q_1 < q_1) \cap (Q_2 < q_2)] \geq 1 - (p_1 + p_2)$$

$$P(Q_1 Q_2 < q_1 q_2) \geq 1 - (p_1 + p_2)$$

3.2.

We start by using the definition for the variance of a sample.

$$\sigma^2 = \frac{\sum_{i=1}^{n} (x_i - \mu)^2}{n - 1}$$

Since squares are always positive, we can conclude the following inequality.

$$\forall i,\ (x_i - \mu)^2 \leq \sum_{i=1}^{n} (x_i - \mu)^2$$

From the definition, we see that the total sum of squared deviations is equal to $\sigma^2(n-1)$. Hence we can further refine our inequality for our singular data point.

$$(x_i - \mu)^2 \leq \sigma^2 (n - 1)$$

Since both sides of the inequality are positive, we can conclude that it must hold even after taking the square root since it is a monotonically increasing function.

$$|x_i - \mu| \leq \sigma\sqrt{n-1}$$

This gives us our required result.
Now we compare this with **Chebyshev's Inequality**, which is stated as follows:

$$P(|X-\mu| \geq k\sigma) \leq \frac{1}{k^2}$$

As n increases we see that the probability of the deviation being more than $\sigma\sqrt{n-1}$ decreases as $\frac{1}{n-1}$ which is true since the previous inequality implies that all deviations lie within $\sigma\sqrt{n-1}$. Hence Chebyshev's inequality is more accurate for larger sample sizes.

§4. Staff Assistant

(a)

The event E_i is defined as the event that the i^{th} candidate is the best and that we hire him. We define a few more events as follows.

$$\begin{aligned} B_i &: \text{The } i^{th} \text{ candidate is the best} \\ H_i &: \text{We hire the } i^{th} \text{ candidate} \\ E_i &: B_i \cap H_i \\ E &: \text{We hire the best candidate} \end{aligned}$$

Using the definition of conditional probability, we can write the probability of event E_i as follows.

$$\begin{aligned} \Pr(E_i) &= \Pr(B_i \cap H_i) \\ &= \Pr(H_i \mid B_i)\Pr(B_i) \end{aligned} \tag{4.1}$$

Since exactly one candidate can be the best, we can calculate the probability of event B_i as follows.

$$\Pr(B_i) = \frac{1}{n} \tag{4.2}$$

If $i \leq m$ then we know that i^{th} candidate cannot be hired. Otherwise, given B_i, we know that i^{th} candidate is the best, so he/she is automatically better than the first m candidates interviewed. The only extra condition, to have them hired is that the candidates $m+1$ to $i-1$ (both inclusive) are not better than the first m candidates. Or in other words, the best among the first $i-1$ candidates is in the first m candidates.

$$\Pr(H_i \mid B_i) = \begin{cases} 0 & i \leq m \\ \frac{m}{i-1} & i > m \end{cases} \tag{4.3}$$

From equations (4.1), (4.2), and (4.3), we get.

$$\Pr(E_i) = \begin{cases} 0 & i \leq m \\ \frac{m}{i-1} \cdot \frac{1}{n} & i > m \end{cases} \tag{4.4}$$

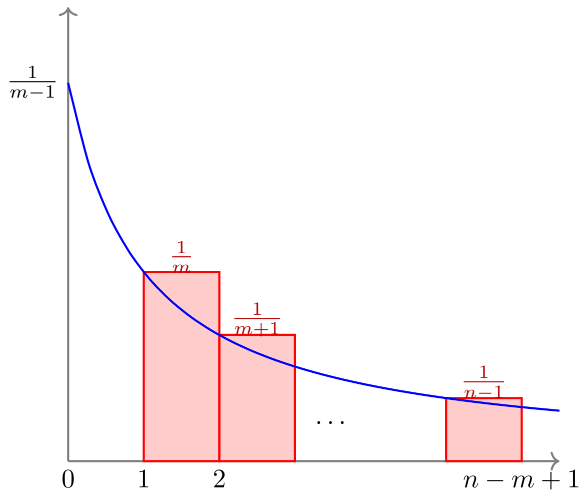
Some candidate i must be the best, so we can calculate the probability of event E as the sum of the probabilities of all the events E_i. From equation (4.4), we get.

$$\begin{aligned} \Pr(E) &= \sum_{j=1}^{n} \Pr(E_i) \\ &= \frac{m}{n} \sum_{j=m+1}^{n} \frac{1}{j-1} \end{aligned} \tag{4.5}$$

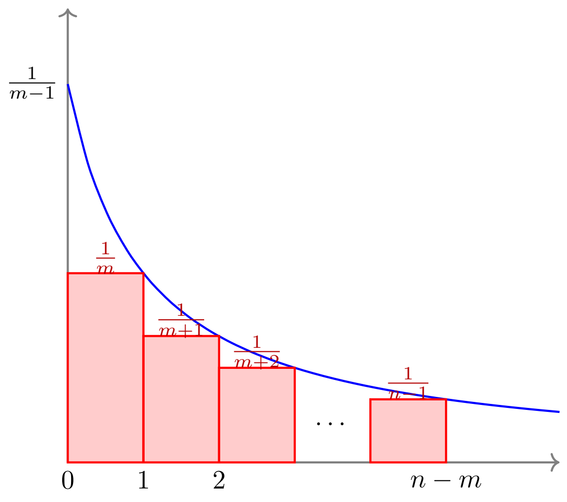
4.1. (b)

Consider the function

$$f(x) = \frac{1}{m+x-1}$$



(a) Graph of $f(x)$. The rectangles constitute the required summation, and is greater than the area under the curve from $x = 1$ to $x = n - m + 1$.



(b) Graph of $f(x)$. The rectangles constitute the required summation, and is less than the area under the curve from $x = 0$ to $x = n - m$.

Figure 1: $f(x)$ and Riemann Sums

From Figure 1a, we can see that the total area of the red rectangles is equal to the summation $\sum_{j=m+1}^{n} \frac{1}{j-1}$, and that area is greater than the area under the curve $f(x)$ from $x = 1$ to $x = n - m + 1$. So, we can obtain a lower bound for the summation.

$$\begin{aligned}\sum_{j=m+1}^{n} \frac{1}{j-1} &\geq \int_{1}^{n-m+1} f(x)\,dx \\ &= \int_{1}^{n-m+1} \frac{1}{m+x-1}\,dx \\ &= \ln(m+x-1)\Big|_{1}^{n-m+1} \\ &= \ln(n) - \ln(m) \qquad (4.6)\end{aligned}$$

Similarly, from Figure 1b, we can see that the total area of the red rectangles is equal to the summation $\sum_{j=1}^{n} \frac{1}{j-1}$, and that area is less than the area under the curve $f(x)$ from $x = 0$ to $x = n - m$. So, we can obtain an upper bound for the summation.

$$\begin{aligned}\sum_{j=m+1}^{n} \frac{1}{j-1} &\leq \int_{0}^{n-m} f(x)\,dx \\ &= \int_{0}^{n-m} \frac{1}{m+x-1}\,dx \\ &= \ln(m+x-1)\Big|_{0}^{n-m} \\ &= \ln(n-1) - \ln(m-1) \qquad (4.7)\end{aligned}$$

From equations (4.5), (4.6) and (4.7), we get

$$\frac{m}{n}\big(\ln(n) - \ln(m)\big) \leq \Pr(E) \leq \frac{m}{n}\big(\ln(n-1) - \ln(m-1)\big) \tag{4.8}$$

4.2. (c)

First, we define $x = \frac{m}{n}$, and since m and n positive integers where $n \geq m$, we have $0 < x \leq 1$. Then, the function that we wish to maximise becomes

$$\begin{aligned}\frac{m}{n}\big(\ln(n) - \ln(m)\big) &= \frac{m}{n}\ln\left(\frac{n}{m}\right)\\ &= -\frac{m}{n}\ln\left(\frac{m}{n}\right)\\ &= -x\ln(x) = g(x) \end{aligned}\tag{4.9}$$

Now, our goal is to maximise $g(x)$. We can do this by taking the derivative of $g(x)$ with respect to x and setting it to zero. First, we calculate the derivative of $g(x)$.

$$\begin{aligned}\frac{\mathrm{d}g(x)}{\mathrm{d}x} &= -\ln(x) - \frac{x}{x}\\ &= -\ln(x) - 1\end{aligned}$$

Next, we set the derivative to zero and solve for x.

$$\begin{aligned}\frac{\mathrm{d}g(x)}{\mathrm{d}x} &= 0\\ -\ln(x) - 1 &= 0\\ \ln(x) &= -1\\ x &= \frac{1}{e}\end{aligned}$$

To verify that this is a maxima, we take the second derivative of $g(x)$.

$$\begin{aligned}\frac{\mathrm{d}^2g(x)}{\mathrm{d}x^2} &= \frac{\mathrm{d}\big(-\ln(x) - 1\big)}{\mathrm{d}x}\\ &= -\frac{1}{x}\\ &< 0 \quad (\text{since } x > 0)\end{aligned}$$

The second derivative is negative, so the function has a maxima at $x = \frac{1}{e}$. Which means that $m = \frac{n}{e}$ is the optimal value of m. Next, we compute the value of $g(x)$ at $x = \frac{1}{e}$, which is the same as the value of $\frac{m}{n}\big(\ln(n) - \ln(m)\big)$ at $m = \frac{n}{e}$.

$$\begin{aligned}g\left(\frac{1}{e}\right) &= -\frac{1}{e}\ln\left(\frac{1}{e}\right)\\ &= \frac{\ln(e)}{e}\\ &= \frac{1}{e}\end{aligned}\tag{4.10}$$

From equations (4.8), (4.9) and (4.10), we get

$$\Pr(E) \geq \frac{1}{e}$$

§5. Free Trade

This question can be treated as a variation of the birthday paradox, The trader in n^{th} place gets the free trade if all the traders in front of him have unique ID numbers and he has an ID number equal to one of them.

This is given by:

Probablity that first n-1 traders have unique IDs $= \prod_{i=0}^{n-2} (\frac{200-i}{200})$

Probablity that n^{th} trader has matching ID with someone in front $= (\frac{n-1}{200})$

$$P(1^{st} \text{ trader gets the free trade}) = 0$$

$$P(2^{nd} \text{ trader gets the free trade}) = \frac{200}{200} \times \frac{1}{200}$$

$$P(3^{rd} \text{ trader gets the free trade}) = \frac{200}{200} \times \frac{199}{200} \times \frac{2}{200}$$

This can be generalised for the n^{th} trader ($n \leq 200$) as follows

$$P(n^{th} \text{ trader gets the free trade}) = \frac{\prod_{i=0}^{n-2}(200-i) \times (n-1)}{200^n}$$

$$P(n^{th} \text{ trader getting the free trade}) = P((n-1)^{th} \text{ trader getting the free trade}) \times \frac{202-n}{200} \times \frac{n-1}{n-2}$$

$$\frac{202-n}{200} \times \frac{n-1}{n-2} \geq 1 \ \forall n \leq 15$$
$$\frac{202-n}{200} \times \frac{n-1}{n-2} \leq 1 \ \forall n \geq 16$$

$\therefore$ P(n^{th} trader getting the free trade) will be maximum for $n = 15$
We can verify this by writing a python program to find these values and plot them

Listing 1: Python code to calculate the probability

```python
import matplotlib.pyplot as plt

placelist = []
probablitylist = []
maxplace,maxprobablity = 0,0

plt.figure(dpi=600)
for x in range(1,100):
    probablity = 1
    for i in range(x-1):
        probablity *= ((200-i)/200)
    probablity*=(x-1)/200
    placelist.append(x)
    probablitylist.append(probablity)
    if(probablity>maxprobablity):
        maxprobablity = probablity
        maxplace = x
    print(f"{x} th place: {probablity}")

plt.scatter(placelist,probablitylist)
plt.axvline(maxplace)
plt.xticks([maxplace])
plt.axhline(maxprobablity)
plt.yticks([maxprobablity])
plt.xlabel("Place ")
plt.ylabel("Probablity of getting the free trade")
print(f"Max value is {maxprobablity} for place = {maxplace}")

plt.savefig("./Q5GraphPlot.png",bbox_inches='tight')
```

The above code results in the following graph:

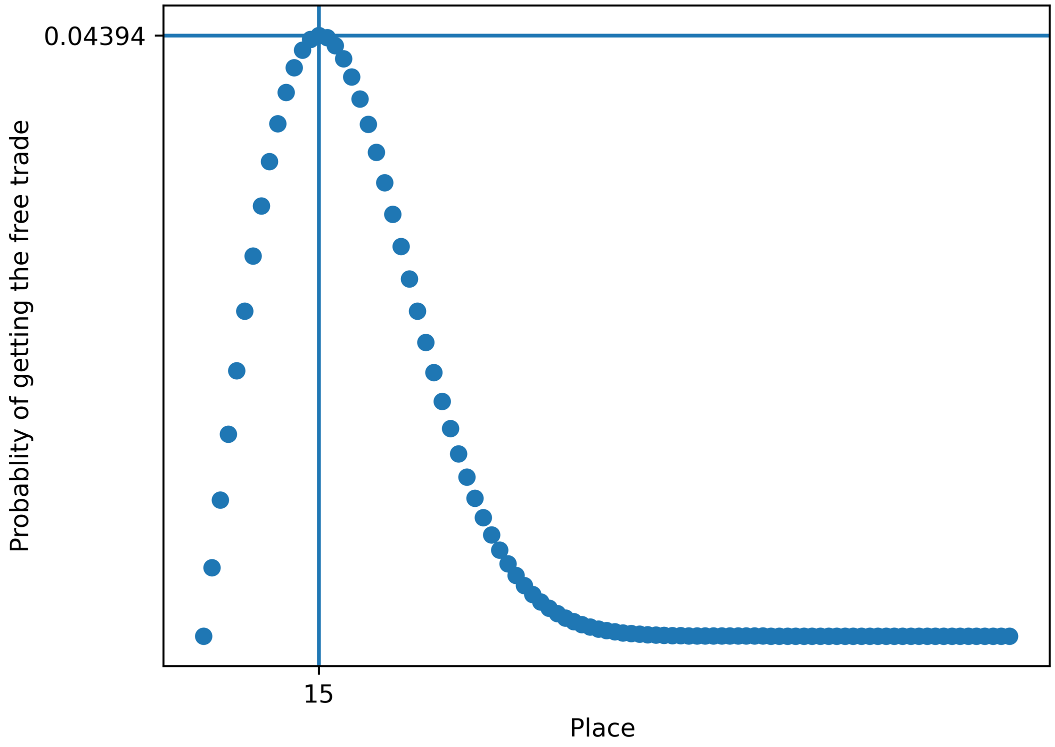


Figure 2: Graph of the probability of getting the free trade

$\therefore$ **We should position ourselves at the 15^{th} place to maximize the probability of getting the free trade.**

§6. Update Functions

All following results assume 1-based indexing. However, the code written below follows 0-based indexing.

- Old mean:

$$\mu = \frac{\sum_{i=1}^{n} x_i}{n}$$

  Updated mean:

$$\mu' = \frac{\sum_{i=1}^{n+1} x_i}{n+1}$$

We can split the term $\sum_{i=1}^{n+1} = \sum_{i=1}^{n} + x_{n+1} = n\mu + x_{n+1}$

$$\mu' = \frac{n\mu + x_{n+1}}{n+1}$$

- Old median:

$$\text{In sorted Array A of even length, median} = (A[n/2+1] + A[n/2])/2$$

$$\text{In sorted Array A of odd length, median} = A[(n+1)/2]$$

Updated median:

$$\text{In a sorted array of even length, median } = \begin{cases} \texttt{NewData} & \text{if } \texttt{NewData} \text{ lies between } A[n/2] \text{ and } A[n/2+1] \\ A[n/2] & \text{if } \texttt{NewData} \text{ is less than } A[n/2] \\ A[n/2+1] & \text{if } \texttt{NewData} \text{ is greater than } A[n/2+1] \end{cases}$$

$$\text{In a sorted array of odd length, median } = \begin{cases} \frac{A[(n+1)/2]+A[(n-1)/2]}{2} & \text{if } \texttt{NewData} \le A[(n-1)/2] \\ \frac{A[(n+1)/2]+A[(n+3)/2]}{2} & \text{if } \texttt{NewData} \ge A[(n+3)/2] \\ \frac{A[(n+1)/2]+\texttt{NewData}}{2} & \text{if } A[(n-1)/2] \le \texttt{NewData} \le A[(n+3)/2] \end{cases}$$

- Old Standard Deviation:

$$\sigma^2 = \frac{(\sum_{i=1}^{n} x_i^2) - n\mu^2}{n-1}$$

New Standard Deviation:

$$\sigma'^2 = \frac{(\sum_{i=1}^{n+1} x_i^2) - (n+1)\mu'^2}{n}$$

We can split the term $\sum_{i=1}^{n+1} x_i^2 = \sum_{i=1}^{n} x_i^2 + x_n^2$

$$\sigma'^2 = \frac{(n-1)\sigma^2 + n\mu^2 + x_n^2 - (n+1)\mu'^2}{n}$$

Listing 2: Python code to calculate new Mean, Median and Standard deviation

```python
def newMean (OldMean: float, NewDataValue: float, n: int, A: list[float]) -> float:
    oldSum = OldMean * n
    newSum = oldSum + NewDataValue
    return newSum / (n + 1)

def newMedian (OldMedian: float, NewDataValue: float, n: int, A: list[float]) ->
   float:
    # Assuming sorted list A
    if n % 2 == 0:
        if A[n // 2 - 1] <= NewDataValue <= A[n // 2]:
            return NewDataValue
        elif NewDataValue < A[n // 2 - 1]:
            return A[n // 2 - 1]
        else:
            return A[n // 2]
    else:
        if NewDataValue <= A[n // 2 - 1]:
            return (A[n // 2 - 1] + A[n // 2]) / 2
        elif NewDataValue >= A[n//2 + 1]:
            return (A[n // 2] + A[n // 2 + 1]) / 2
        else:
            return (A[n // 2] + NewDataValue) / 2
```

```

def newStd (OldMean: float, OldStd: float, NewMean: float, NewDataValue: float, n:
    int, A: list[float]) -> float:
    oldSquareSum = ((OldStd ** 2) * (n - 1)) + (n * (OldMean ** 2))
    newSquareSum = oldSquareSum + (NewDataValue ** 2)
    return ((newSquareSum - ((n + 1) * (NewMean) ** 2)) / n) ** 0.5
```

To update the histogram, we check which bin `NewData` lies in. If no such bin exists, we create a new one with just `NewData`.

§7. Plots

- **Violin Plot:** These plots represent the density of data as the width at that section. The mean and median of the dataset can be seen clearly. It also shows the symmetry/skewness of the dataset based on the wideness on either side. It shows if the dataset has multiple peaks. It also allows us to compare groups by showing them side by side. It can be used for pharmaceutical trials, ecological studies, financial analysis and quality control.

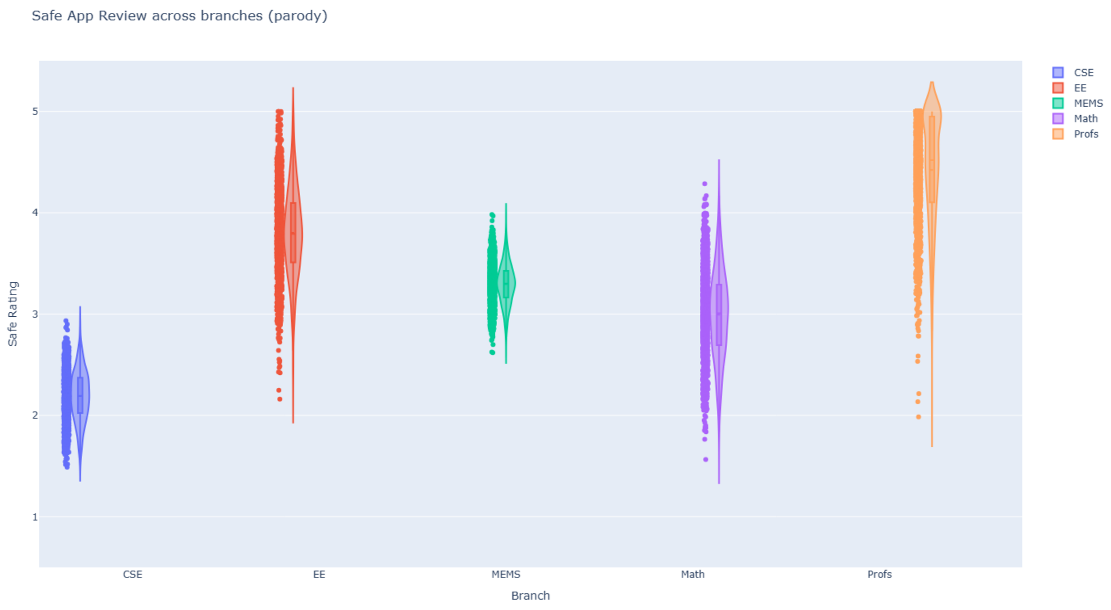


Figure 3: Violin Plot

- **Pareto Plot:** A Pareto chart is a type of bar chart where each category is represented by a vertical bar. The bars are arranged in descending order of frequency or magnitude from left to right. A cumulative line is often superimposed on the chart to show the total impact of all categories combined. Pareto charts are used to identify the most significant factors in a dataset, typically following the 80/20 rule, where 80% of the effects come from 20% of the causes. They are commonly used in quality control, project management, and process improvement to prioritize issues or root causes.

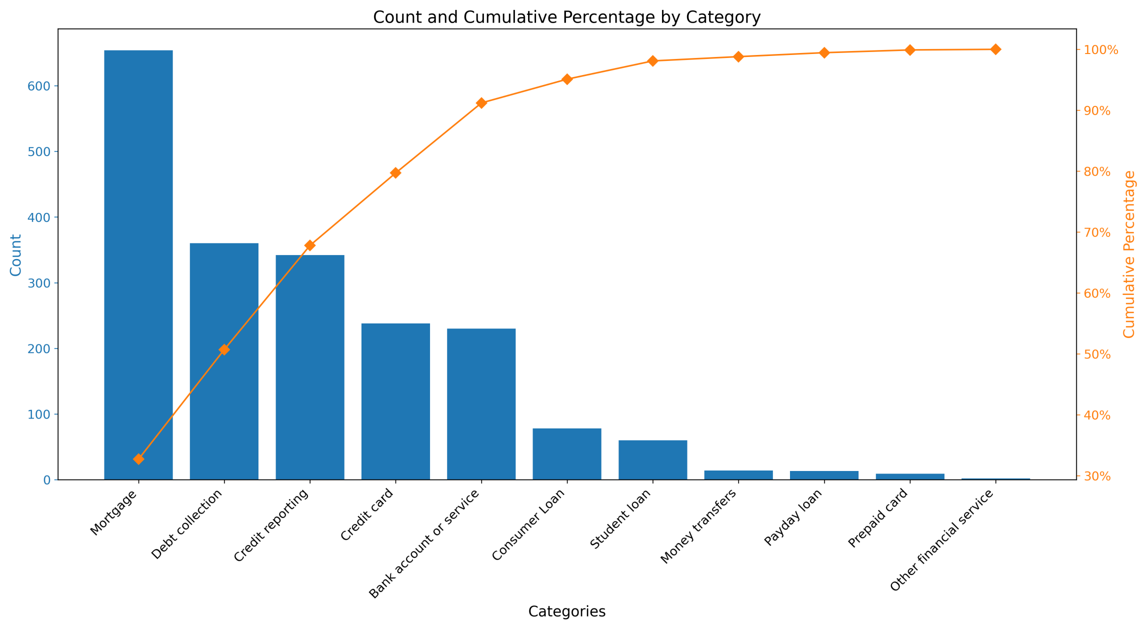


Figure 4: Pareto Plot, data from [3]

- **Waterfall Plot:** A waterfall plot is a three-dimensional visualization technique commonly used to display changes in signal intensity, frequency, or other continuous variables over time. Each point on the plot represents data at a given time, with the x-axis typically showing time or another independent variable, the y-axis displaying the frequency or another dependent variable, and color or intensity representing the magnitude of the data. Waterfall plots are frequently used in fields such as signal processing, spectroscopy, and audio analysis to monitor how data evolves over time, offering a clear view of trends, patterns, and fluctuations.

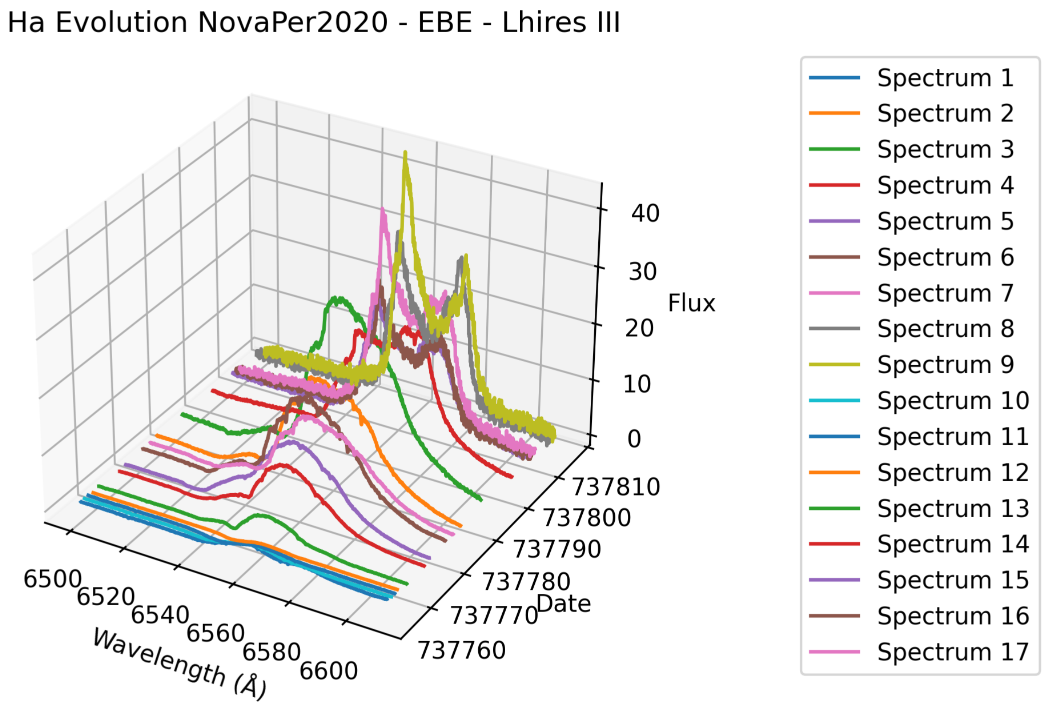


Figure 5: Waterfall Plot, data from [2]

- **Coxcomb Plot:** These plots represent data in a circular layout where each category is represented by a wedge. All wedges have the same angular width. Larger areas and longer radii represent larger values and colours are used to represent different time periods. They are used to represent data that changes cyclically like financial performance over the quarters, monthly rainfall or performance of sports players over a season.

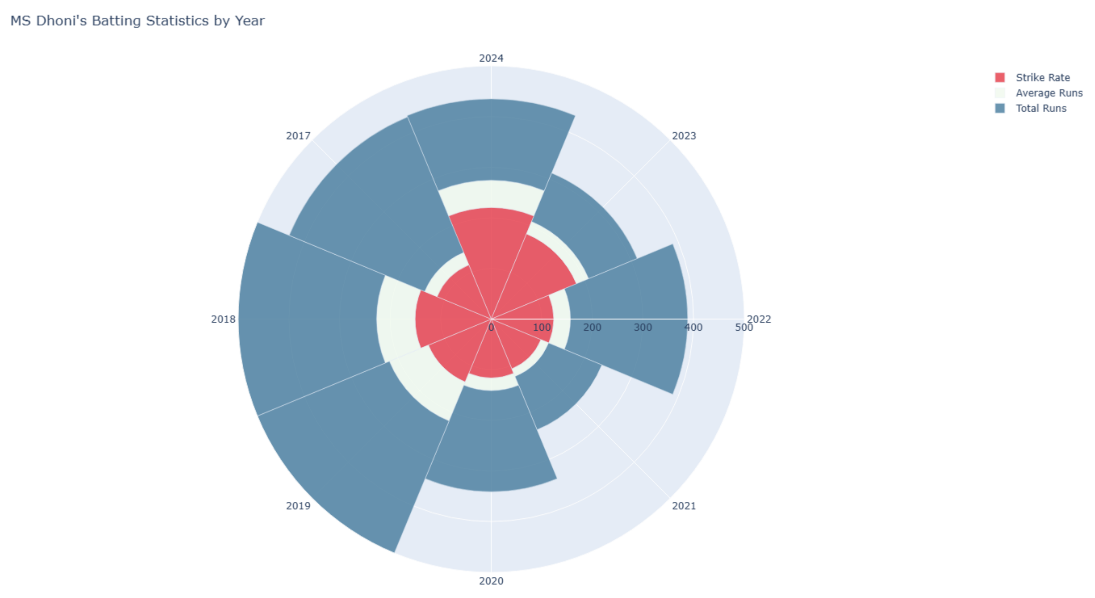


Figure 6: Coxcomb Plot, data from [1]

§8. Monalisa

First, the image is read using `matplotlib`'s `imread()` method. Then we run a for loop for the shift amount and use `numpy`'s slicing to shift the image. After, the shifted image is obtained, it's correlation with the original image is calculated using `numpy`'s `corrcoef()` method. The shift amount is stored in `correlations_x` and the correlation value is stored in `correlations_y`. Finally, a plot is made using `matplotlib`'s `plot()` method.

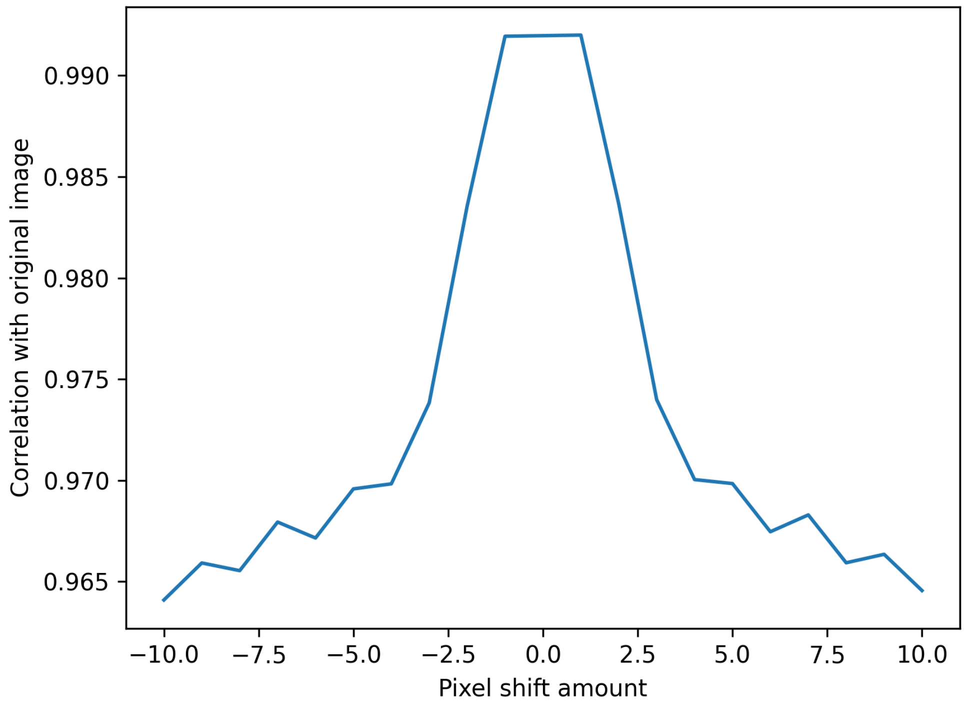


Figure 7: Plot of correlation values against the pixel shift amounts

Listing 3: Python code to compute and plot the correlations of the shifted images

```python
# imports
import numpy as np
import matplotlib.pyplot as plt

plt.rcParams["figure.dpi"] = 300  # higher resolution output images

img = plt.imread("Mona_Lisa.jpg")  # read the image

plt.imshow(img)
plt.gca().set_xlabel("x")  # set the x-label of the current Axes (returned by the gca)
plt.gca().set_ylabel("y")  # set the y-label of the current Axes

correlations_x = np.array(
    list(range(-10, 0)) + list(range(1, 11))
)  # x axis values for correlation plot
correlations_y = np.zeros(20)  # y axis values for correlation plot

for k in range(20):
    tx = correlations_x[k]  # shift amount
    img_new = np.zeros_like(img, dtype=np.uint8)  # create a new image
    if tx >= 0:
        img_new[:, tx:] = img[:, : img.shape[1] - tx]
    else:
        img_new[:, : img.shape[1] + tx] = img[:, -tx:]
    correlations_y[k] = np.corrcoef(img.flatten(), img_new.flatten())[
        0, 1
    ]  # store correlation

# plot the correlations
plt.figure()
plt.plot(correlations_x, correlations_y)
plt.xlabel("Pixel shift amount")
plt.ylabel("Correlation with original image")
plt.savefig("correlation.png")
```

The plot obtained is shown in Figure 7. We can see that the correlation between the original image and the shifted image decreases as the shift amount increases. This is because adjacent pixels in the image have higher correlation as compared to pixels that are apart. In high resolution images, colors transition smoothly (without an abrupt change) between adjacent pixels. This is easy to observe from Figure 8.

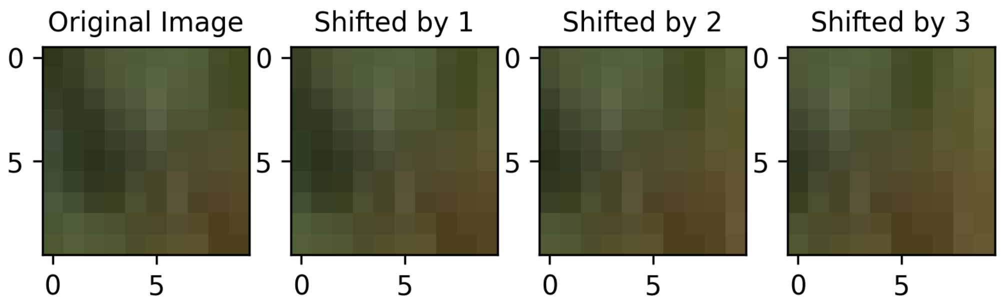


Figure 8: Top left 10x10 pixels of the original and shifted images

To obtain the histogram of the image, we first convert the image to grayscale to get pixel intensities. This is done using the formula

$$I = 0.2989 \times R + 0.5870 \times G + 0.1140 \times B$$

Then we use two for loops to iterate over all the pixels in the image, and calculate the histogram (frequency of each pixel intensity), which is stoed in variable `h`. The histogram is then normalised by dividing by the total number of pixels in the image, this is soted in the variable `p`. Finally, the histogram is plotted using `matplotlib`'s `hist()` method, but by using the `weights` parameter as we've already computed the histogram. The output is shown in Figure 9.

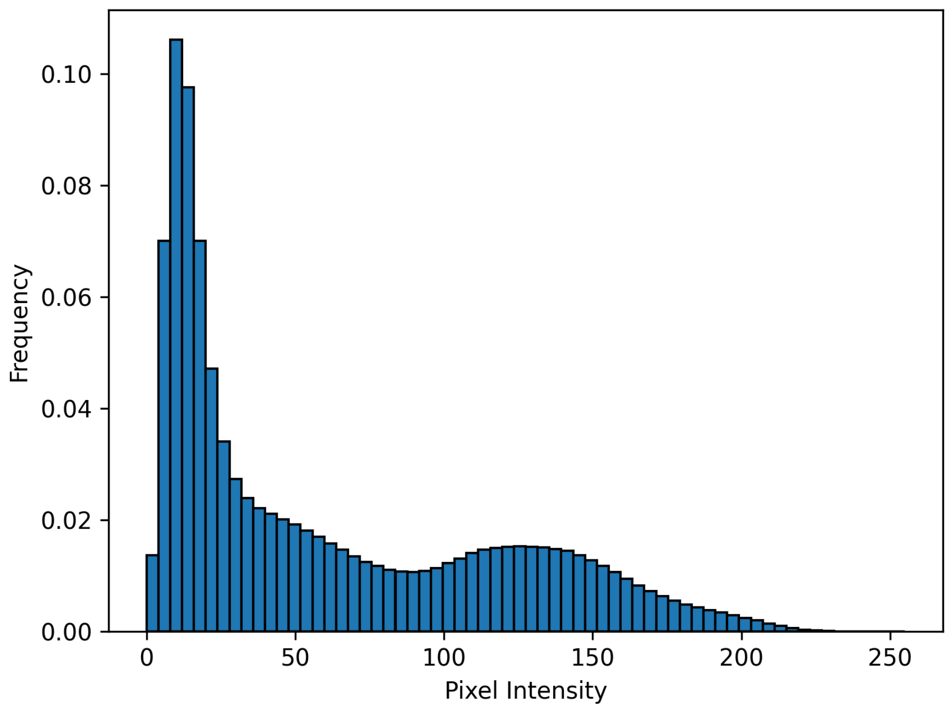


Figure 9: Normalized histogram of the image

Listing 4: Continuation of Listing 3, to compute and plot the normalised histogram

```python
# grayscale image for pixel intensities
img_gray = img[:, :, 0] * 0.2989 + img[:, :, 1] * 0.5870 + img[:, :, 2] * 0.1140
plt.imshow(img_gray, cmap="gray")

# histogram
h = np.zeros(256)
for i in range(img.shape[0]):
    for j in range(img.shape[1]):
        h[int(img_gray[i, j])] += 1
# normalized histogram
p = h / (img_gray.shape[0] * img_gray.shape[1])

# plot the normalized histogram
plt.figure()
_ = plt.hist(np.arange(256), weights=p, bins=64, edgecolor="black")
plt.xlabel("Pixel Intensity")
plt.ylabel("Frequency")
plt.savefig("normalized_histogram.png")
```

[0]Running Instructions: `python ./file_name.py`

References

[1] IPL T20. Chennai super kings squad details, 2024. Data for coxcomb plots. URL: https://www.iplt20.com/teams/chennai-super-kings/squad-details/1.

[2] Matthieu Lel. Spec3d data, 2024. Accessed: 2024-08-25. URL: https://gitlab.com/matthieulel/aspyt/-/tree/master/notebooks/spec3d_data.

[3] Daniel Selener. Consumer complaint database, 2024. Accessed: 2024-08-25. URL: https://www.kaggle.com/datasets/selener/consumer-complaint-database.