experimenting_fair_coin_31.05.2025

May 31, 2025

1 Experimenting & Plotting Relative Frequencies of Heads & Tails with respect to Number of Tosses for a Fair Coin with help of pseudo-random choosing.

[ ]:
```
# installing required modules

%pip install matplotlib
%pip install seaborn
```

[1]:
```
#importing required modules

import matplotlib.pyplot as plt
import random
import seaborn as sns
```

[2]:
```
def take_input():
    # function to take input the number of tosses

    n = int(input("Enter number of tosses: "))
    return n
```

[3]:
```
def calculate():

    no_of_tosses = take_input()
    print("Number of tosses:", no_of_tosses)

    toss_result = None # will store H or T
    tosses = [] # stores the results as list
    possible_outcomes = ['H', 'T'] # possible outcomes

    count_H = 0 # counter for H
    count_T = 0 # counter for T

    rel_freq_H = 0.0 # relative frequency for H
    rel_freq_T = 0.0 # relative frequency for T

    rel_freq_list_H = [] # relative frequency list for H
    rel_freq_list_T = [] # relative frequency list for T

    estimated_rel_freq = 0.500
    deviation_percent_list_H = []
    deviation_percent_list_T = []

    for toss_number in range(1, (no_of_tosses + 1)): # range: [1, total number␣
↪of tosses entered]

        toss_result = random.choice(possible_outcomes) # randomly selects from␣
↪H or T
        tosses.append(toss_result) # stores the result into list

        # obvious logic implementation below (^_^)

        if (toss_result == 'H'):
            count_H += 1

        else:
            count_T += 1

        rel_freq_H = round(float(count_H/toss_number), 3)
        rel_freq_T = round(float(count_T/toss_number), 3)

        rel_freq_list_H.append(rel_freq_H)
        rel_freq_list_T.append(rel_freq_T)

        deviation_percent_list_H.append(round((abs(estimated_rel_freq -␣
↪rel_freq_H) * 100/estimated_rel_freq), 3))
        deviation_percent_list_T.append(round((abs(estimated_rel_freq -␣
↪rel_freq_T) * 100/estimated_rel_freq), 3))

    # printing to see the values

    # print("Tosses list:")
    # print(tosses)

    # print("Relative frequency list of Heads H:")
    # print(rel_freq_list_H)

    # print("Relative frequency list of Tails T:")
    # print(rel_freq_list_T)

    return(no_of_tosses, rel_freq_list_H, rel_freq_list_T,␣
↪deviation_percent_list_H, deviation_percent_list_T)
```

```
[4]: def plotting():

         # receive the values from the function
         no_of_tosses, rel_freq_list_H, rel_freq_list_T, deviation_percent_list_H,␣
     ↪deviation_percent_list_T = calculate()
         x_values = list(range(1, no_of_tosses+1))

         # set universal font size and Seaborn style
         plt.rcParams.update({'font.size': 10})  # setting font size
         sns.set(style="whitegrid")

         # side by side subplots
         fig, axes = plt.subplots(1, 2, figsize=(16, 6), dpi=200)

         # customizing plots

         # Plot 1: Relative Frequencies
         axes[0].plot(x_values, rel_freq_list_H, linestyle='--', color='blue',␣
     ↪linewidth=1, alpha=0.8, label='Heads')
         axes[0].plot(x_values, rel_freq_list_T, linestyle='--', color='red',␣
     ↪linewidth=1, alpha=0.8, label='Tails')
         axes[0].grid(True, linestyle='--', alpha=0.4)
         axes[0].set_yticks([0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0])
         axes[0].set_xlim(-0.5, no_of_tosses)
         axes[0].set_ylim(-0.2, 1.2)
         axes[0].set_xlabel("Number of tosses")
         axes[0].set_ylabel("Relative frequencies")
         axes[0].set_title("Relative Frequencies of Heads & Tails")
         axes[0].legend(fancybox=True)

         # Plot 2: Deviation Percentages
         axes[1].plot(x_values, deviation_percent_list_H, linestyle='-',␣
     ↪color='blue', linewidth=0.5, alpha=0.8, label='Heads')
         axes[1].plot(x_values, deviation_percent_list_T, linestyle='-',␣
     ↪color='red', linewidth=0.5, alpha=0.8, label='Tails')
         axes[1].grid(True, linestyle='--', alpha=0.4)
         axes[1].set_yticks([0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100])
         axes[1].set_xlim(-0.5, no_of_tosses)
         axes[1].set_ylim(-5, 105)
         axes[1].set_xlabel("Number of tosses")
         axes[1].set_ylabel("Deviation Percentage from 0.5")
         axes[1].set_title("Deviation Percentage of Relative Frequency")
         axes[1].legend(fancybox=True)

         plt.tight_layout()
         plt.show()
```

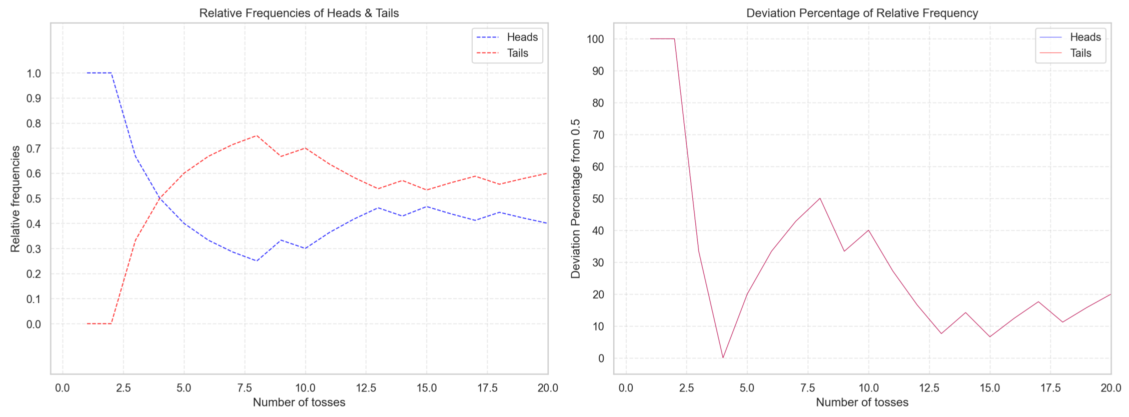
```

```

1.0.1 Plotting with three different values for `total number of tosses`.

In the absolute deviation plot for a fair coin, the lines for heads and tails perfectly overlap because, at every toss, the deviation of each from the expected probability (0.5) is always equal in magnitude. Since the relative frequencies of heads and tails always sum to 1, their absolute deviations from 0.5 are identical, resulting in a single overlapping curve on the graph.

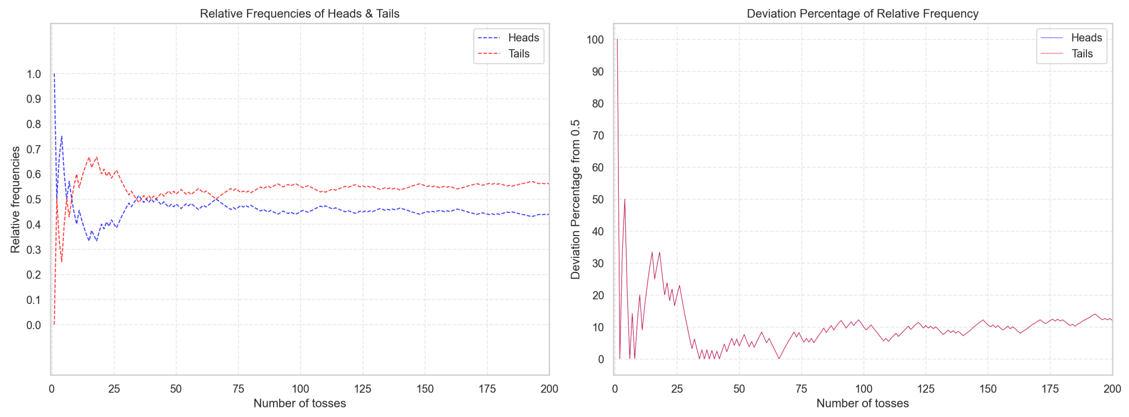
[5]:
```
plotting()
```

```
Number of tosses: 20
```



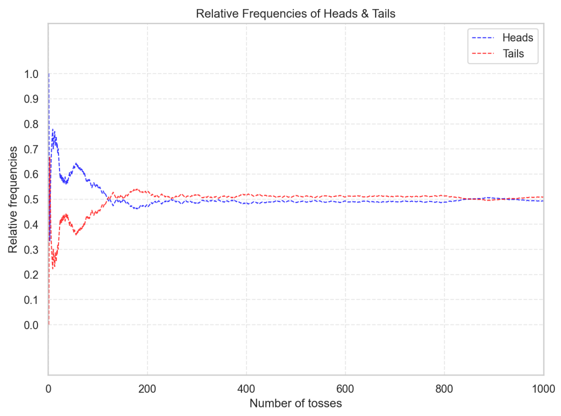
[6]:
```
plotting()
```

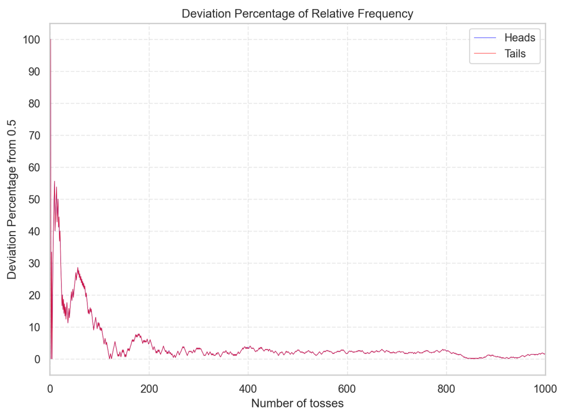
```
Number of tosses: 200
```



[7]:
```
plotting()
```

```
Number of tosses: 1000
```


Relative Frequencies of Heads & Tails
Heads
Tails
Relative frequencies
Number of tosses


Deviation Percentage of Relative Frequency
Heads
Tails
Deviation Percentage from 0.5
Number of tosses