Monte Carlo Simulation

Python Programming Lab

05506231 Statistics and Probability

Asst. Prof. Dr.Anantaporn Hanskunatai

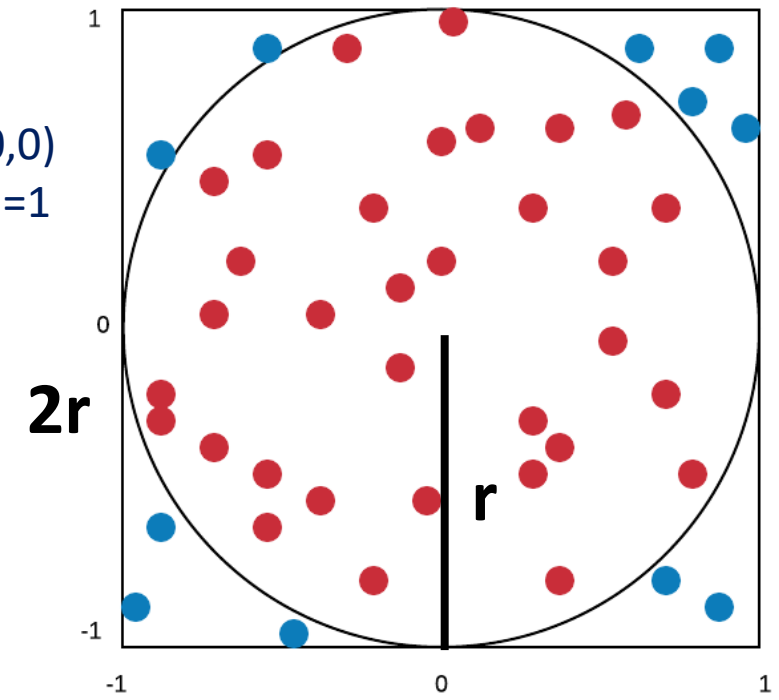
Outline

- Monte Carlo simulations case studies
  - Estimating the value of Pi
  - Predicting sales commission budget

Estimating the Value of Pi

- Monte Carlo simulations use random sampling to obtain numerical results

Center = (0,0)
Radius (r) =1



- a dart thrower who always manages to hit the board and is equally likely to hit any area of the board
- the probability that the dart will hit p(hit) is

$$p(hit) = \frac{\#\ of\ darts\ in\ the\ circle}{\#\ of\ darts\ thrown} = \frac{area\ of\ circle}{area\ of\ square} = \frac{\pi r^2}{4r^2}$$

$$p(hit) = \frac{\pi}{4}$$

$$4 * p(hit) = \pi$$

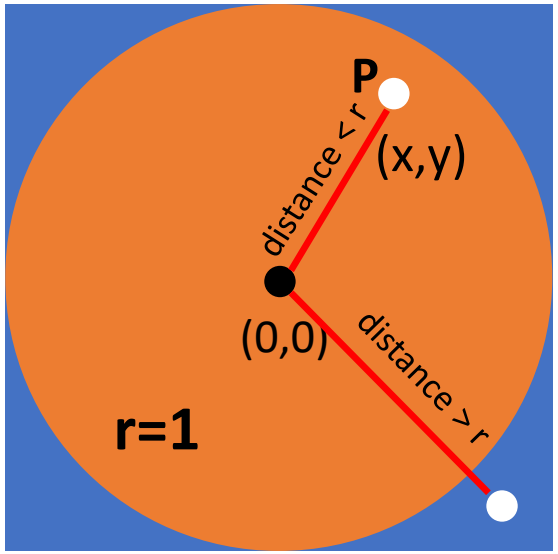
Image source: http://selkie-macalester.org/csinparallel/modules/CrossPlatformProgramming/build/html/MonteCarloPi/Pi.html

Estimating the Value of Pi

The Algorithm

1. Set the radius, sampling size to N (#iteration of random points)
2. circle_points=0
2. For i=1 to N
3. random x and y as a point p=(x,y)
4. If point p is inside the circle increment circle_points
5. End for
6. Calculate Pi = 4*(circle_points/N)
7 Return Pi

Checking the Position of Point p (inside or outside the circle)



Distance(P, center) = $\sqrt{(x-0)^2+(y-0)^2}$

$= \sqrt{x^2+y^2}$

If distance(P,center) ≤ r then the point P is inside the circle
otherwise the point P is outside the circle

Python Programming for Estimating the Value of Pi

- Import libraries and set the initial values

```
import random
import math
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
r = 1.0  # radius
N = 1001 # number of iteration
d = {"Trials":[],"Pi":[]}
```

Python Programming for Estimating the Value of Pi

- Monte Carlo Simulation

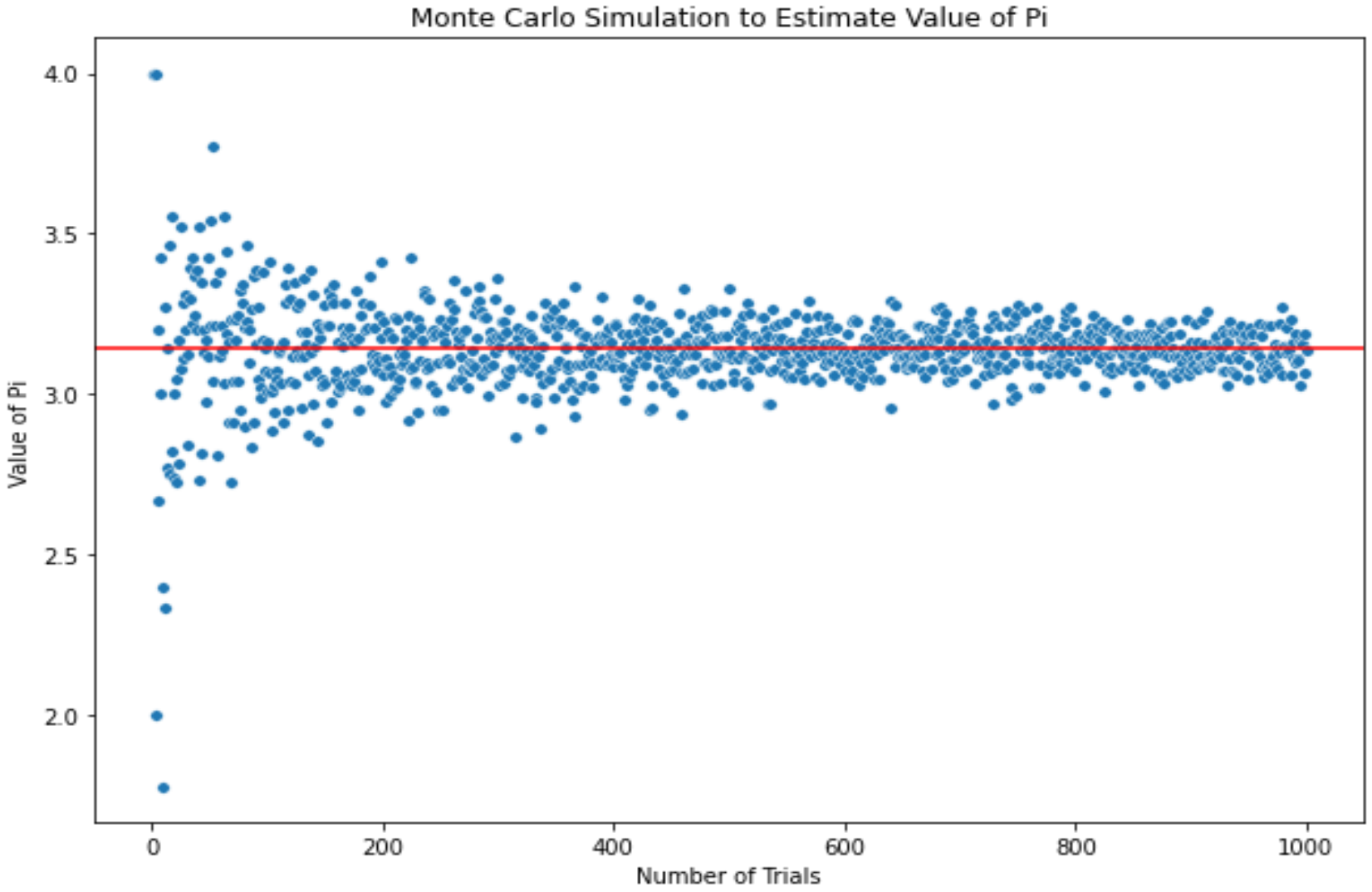
```
for T in range(1,N):
  circle_p=0
  for i in range(T):
    x = random.uniform(-1.0, 1.0)
    y = random.uniform(-1.0, 1.0)

    x2 = x ** 2
    y2 = y ** 2

    if math.sqrt(x2 + y2) <= r:
      circle_p+=1

  d["Trials"].append(T)
  d["Pi"].append((circle_p/T)*4)
```

Python Programming for Estimating the Value of Pi

- Visualize Pi values calculated by Monte Carlo

```
df = pd.DataFrame(data=d)
plt.figure(figsize = (10,7))
plot = sns.scatterplot(x="Trials", y="Pi",  s=30, marker="o", data=df)
plot.set(title='Monte Carlo Simulation to Estimate Value of Pi', xlabel="
Number of Trials", ylabel="Value of Pi")
plt.axhline(y=3.14, color='r', linestyle='-')
plt.show()
```

Python Programming for Estimating the Value of Pi



Python Programming for Estimating the Value of Pi

```
df
```

	Trials	Pi
0	1	4.000000
1	2	4.000000
2	3	4.000000
3	4	2.000000
4	5	3.200000
...	...	...
995	996	3.152610
996	997	3.157472
997	998	3.186373
998	999	3.067067
999	1000	3.136000

1000 rows × 2 columns

```
df.head(10)
```

	Trials	Pi
0	1	4.000000
1	2	4.000000
2	3	4.000000
3	4	2.000000
4	5	3.200000
5	6	2.666667
6	7	3.428571
7	8	3.000000
8	9	1.777778
9	10	2.400000

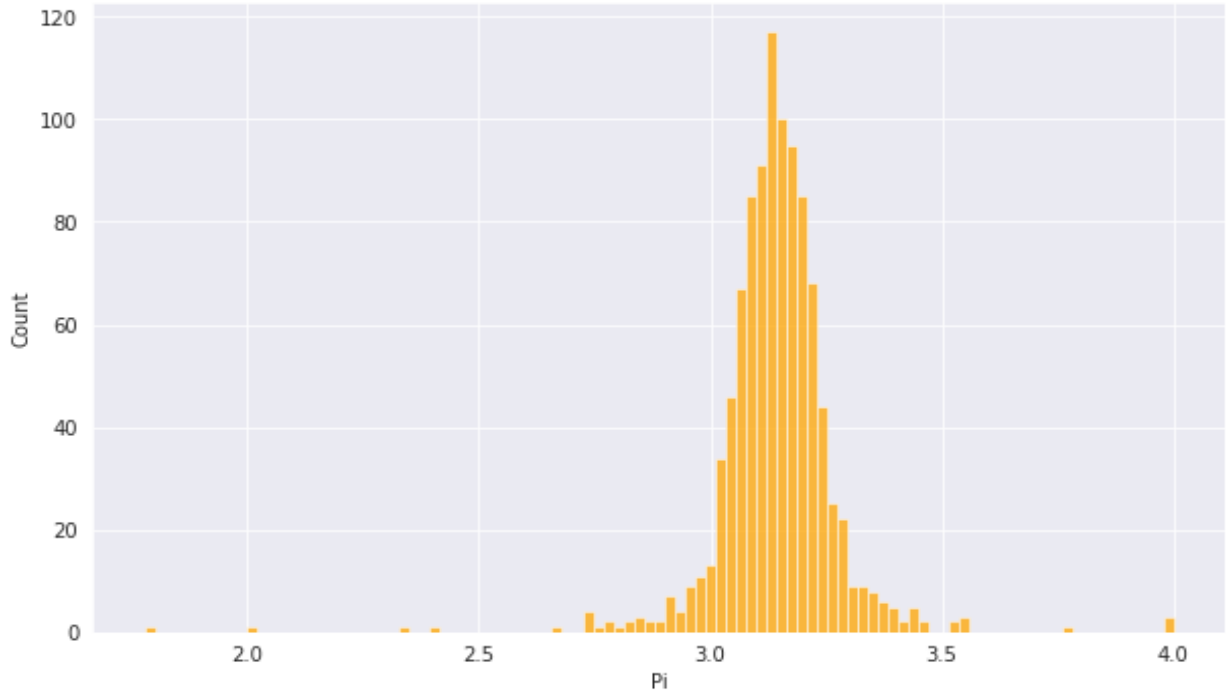
```
df.tail(10)
```

	Trials	Pi
990	991	3.188698
991	992	3.108871
992	993	3.101712
993	994	3.162978
994	995	3.027136
995	996	3.152610
996	997	3.157472
997	998	3.186373
998	999	3.067067
999	1000	3.136000

Python Programming for Estimating the Value of Pi

- Histogram of Pi values

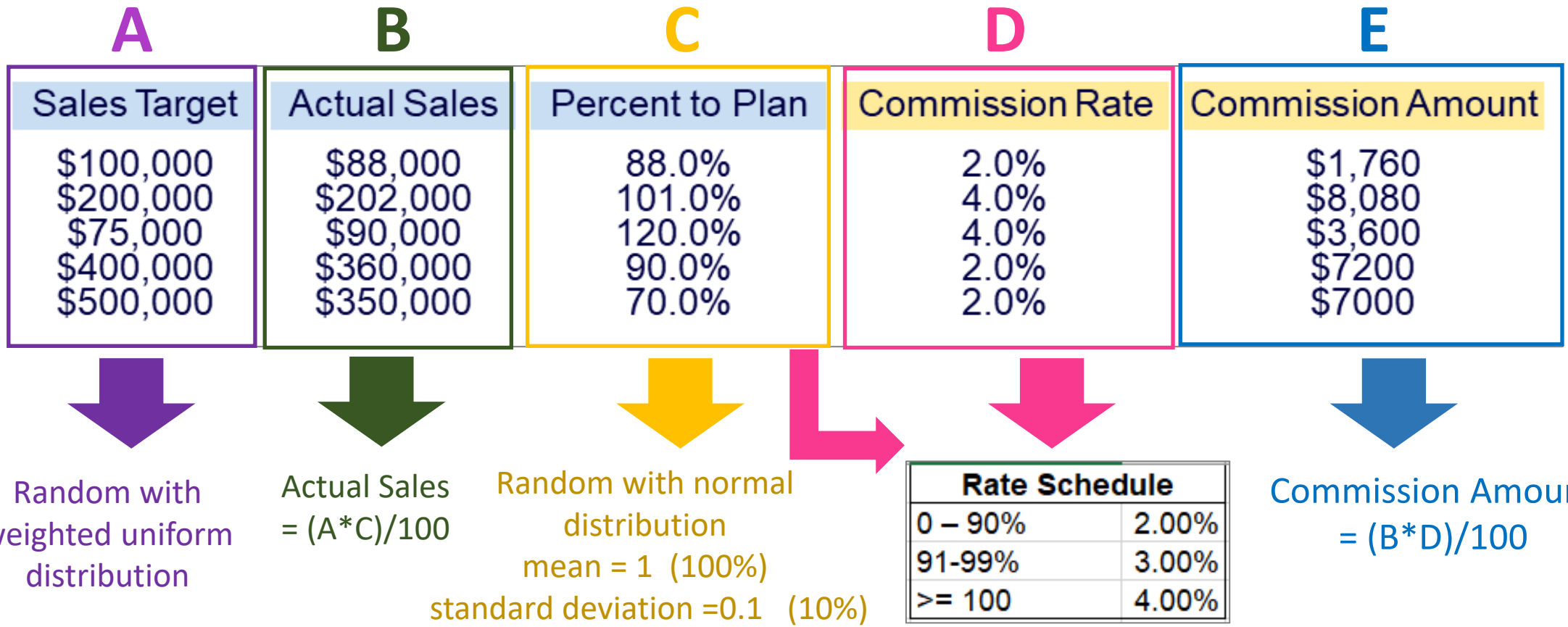
```
import seaborn as sns
sns.set_style('darkgrid')
fig = plt.figure(figsize = (10,6))
sns.histplot(df, x="Pi",color='orange');
```

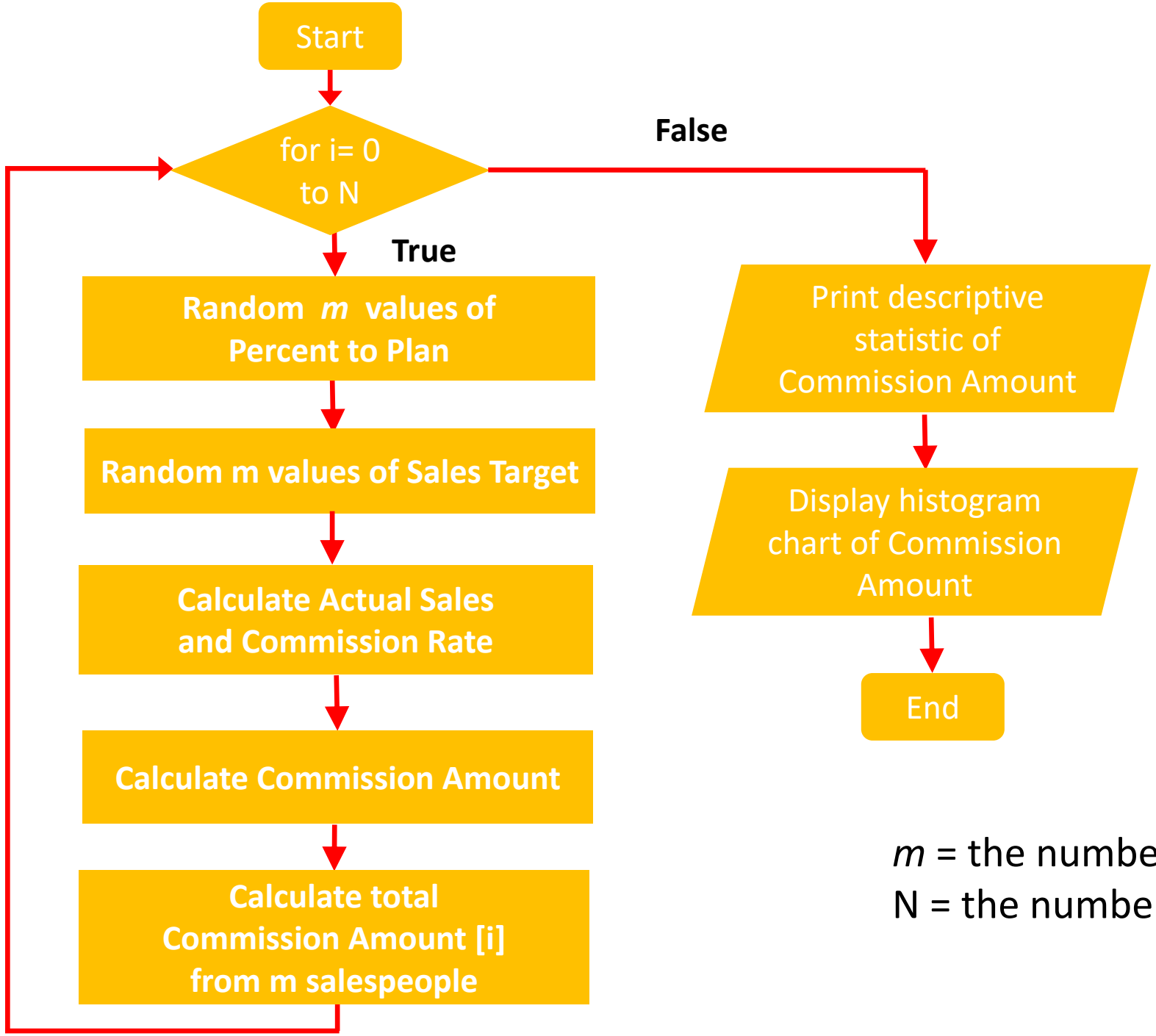



```
df['Pi'].mean()
```

```
df['Pi'].mean()

3.1408978362378472
```

Predicting Sales Commission Budget





m = the number of salespeople
N = the number of iterations

Python Programming for Predicting Sales Commission Budget

- Random Percent to Plan (normal distribution)

```
import pandas as pd
import numpy as np
import seaborn as sns
sns.set_style('whitegrid')
```

```
avg = 1
std_dev = .1
num_reps = 500
pct_to_target = np.random.normal(avg, std_dev, num_reps).round(2)
```

```
pct_to_target[0:10]
```

```
array([1.1 , 1.03, 0.91, 0.88, 0.94, 1.1 , 0.93, 0.9 , 0.98, 0.78])
```

Python Programming for Predicting Sales Commission Budget

- Random Sales Target (weighted uniform distribution)

```
sales_target_values = [75_000, 100_000, 200_000, 300_000, 400_000, 500_000]
sales_target_prob = [.3, .3, .2, .1, .05, .05]
sales_target = np.random.choice(sales_target_values, num_reps, p=sales_target_prob)
```

- Create DataFrame

```
df = pd.DataFrame(index=range(num_reps), data={'Pct_To_Target': pct_to_target,
'Sales_Target': sales_target})
```

- Compute Actual Sales

```
df['Sales'] = df['Pct_To_Target'] * df['Sales_Target']
```

Python Programming for Predicting Sales Commission Budget

```
df
```

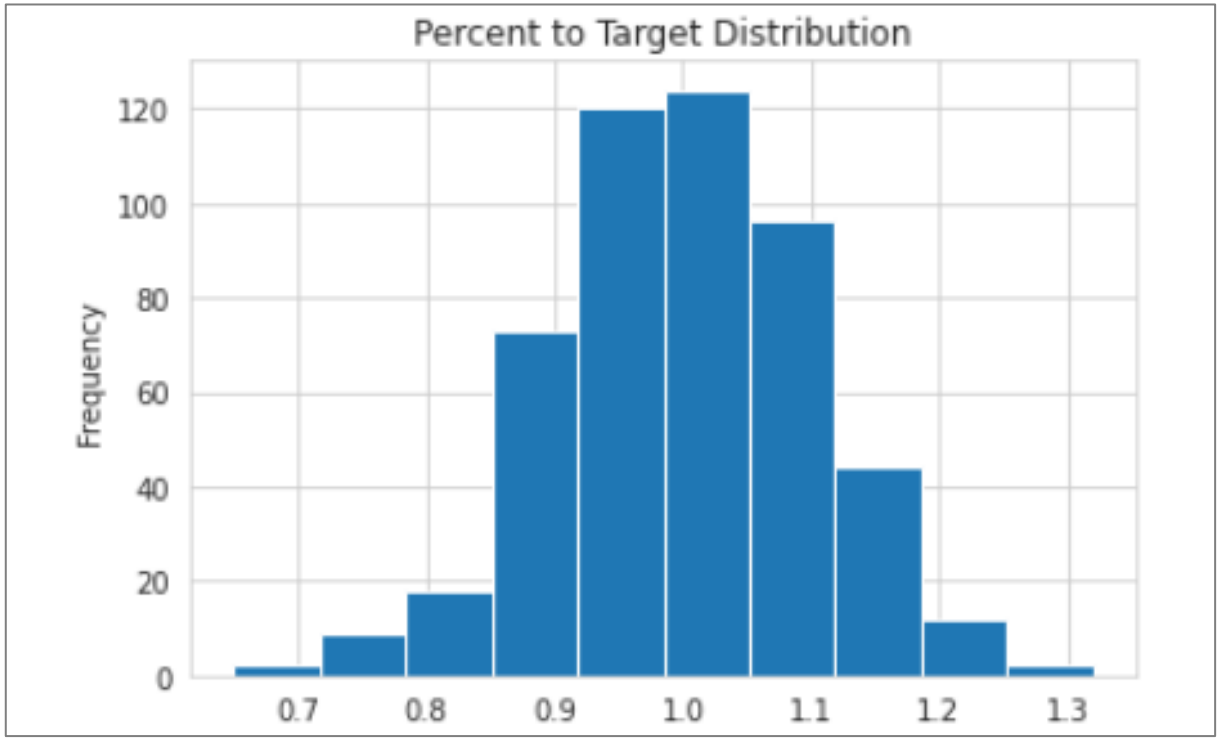
	Pct_To_Target	Sales_Target	Sales
0	1.10	100000	110000.0
1	1.03	100000	103000.0
2	0.91	75000	68250.0
3	0.88	100000	88000.0
4	0.94	100000	94000.0
...	...	...	...
495	1.00	400000	400000.0
496	1.17	100000	117000.0
497	0.98	75000	73500.0
498	1.01	100000	101000.0
499	1.05	100000	105000.0

500 rows × 3 columns

Python Programming for Predicting Sales Commission Budget

- Visualize the distribution of Percent to Plan (Pct_To_Target)

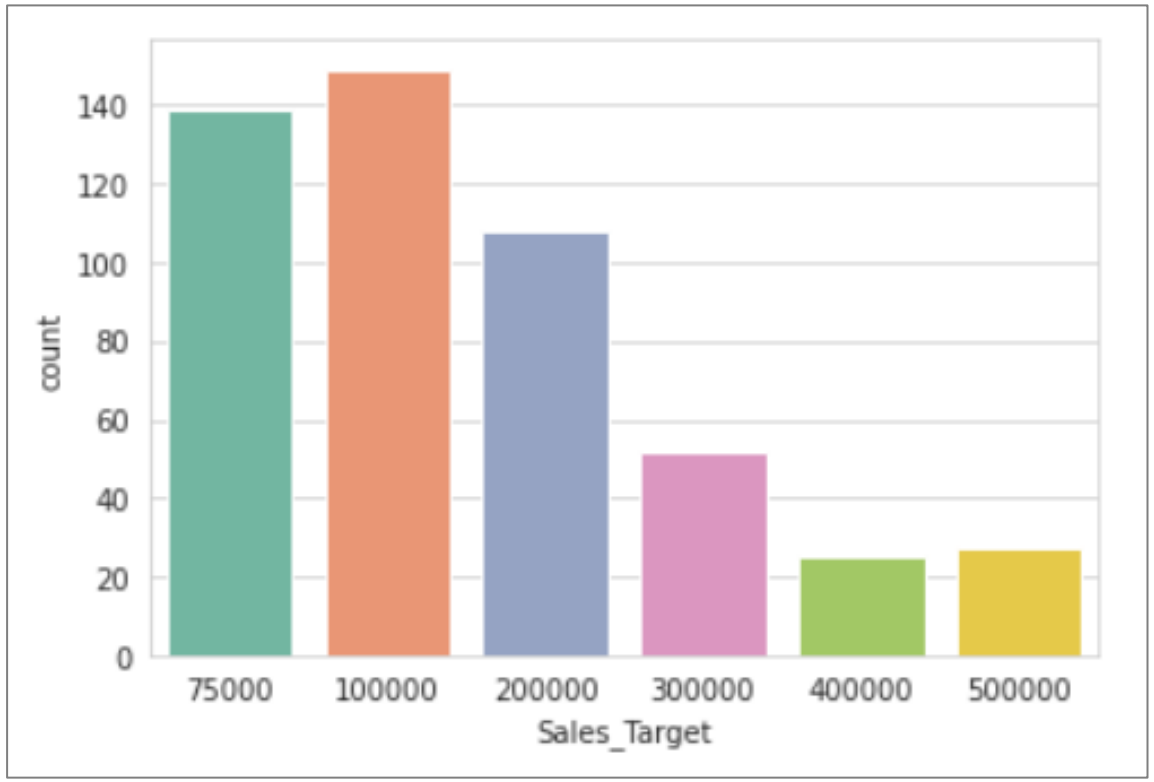
```
df['Pct_To_Target'].plot(kind='hist', title='Percent to Target Distribution')
```



Python Programming for Predicting Sales Commission Budget

- Visualize the distribution of Sales Target

```
sns.countplot(x = 'Sales_Target', data = df, palette = 'Set2')
```



Python Programming for Predicting Sales Commission Budget

- Calculate Commission Rate

```
def calc_commission_rate(x):
    if x <= .90:
        return .02
    if x <= .99:
        return .03
    else:
        return .04
```

Rate Schedule	
0 – 90%	2.00%
91-99%	3.00%
>= 100	4.00%

```
df['Commission_Rate'] = df['Pct_To_Target'].apply(calc_commission_rate)
```

- Calculate Commission Amount

```
df['Commission_Amount'] = df['Commission_Rate'] * df['Sales']
```

Python Programming for Predicting Sales Commission Budget

```
df.head(10)
```

	Pct_To_Target	Sales_Target	Sales	Commission_Rate	Commission_Amount
0	1.10	100000	110000.0	0.04	4400.0
1	1.03	100000	103000.0	0.04	4120.0
2	0.91	75000	68250.0	0.03	2047.5
3	0.88	100000	88000.0	0.02	1760.0
4	0.94	100000	94000.0	0.03	2820.0
5	1.10	200000	220000.0	0.04	8800.0
6	0.93	75000	69750.0	0.03	2092.5
7	0.90	100000	90000.0	0.02	1800.0
8	0.98	75000	73500.0	0.03	2205.0
9	0.78	75000	58500.0	0.02	1170.0

Python Programming for Predicting Sales Commission Budget

- Simulation with 1,000 iterations

```
num_simulations = 1000
all_stats = []
# Loop through many simulations
for i in range(num_simulations):
    # Choose random inputs for the sales targets and percent to target
    sales_target = np.random.choice(sales_target_values, num_reps, p=sales_target_prob)
    pct_to_target = np.random.normal(avg, std_dev, num_reps).round(2)
    # Build the dataframe based on the inputs and number of reps
    df = pd.DataFrame(index=range(num_reps), data={'Pct_To_Target': pct_to_target,
                                                   'Sales_Target': sales_target})
    # Back into the sales number using the percent to target rate
    df['Sales'] = df['Pct_To_Target'] * df['Sales_Target']
    # Determine the commissions rate and calculate it
    df['Commission_Rate'] = df['Pct_To_Target'].apply(calc_commission_rate)
    df['Commission_Amount'] = df['Commission_Rate'] * df['Sales']
    # We want to track sales,commission amounts and sales targets over all the simulations
    all_stats.append([df['Sales'].sum().round(0),
                      df['Commission_Amount'].sum().round(0),
                      df['Sales_Target'].sum().round(0)])
```

Python Programming for Predicting Sales Commission Budget

- Create DataFrame of Actual Sales, Commission Amount, and Sales Target from 1,000 iterations

```
results_df = pd.DataFrame.from_records(all_stats, columns=['Sales',
'Commission_Amount','Sales_Target'])
```

```
results_df
```

	Sales	Commission_Amount	Sales_Target
0	79842750.0	2693095.0	80375000
1	82048750.0	2866695.0	80900000
2	86590750.0	2914702.0	87100000
3	88511750.0	3020305.0	88675000
4	83995250.0	2829298.0	84300000
...	...	...	...
995	85594250.0	2968085.0	84825000
996	84027000.0	2813100.0	84875000
997	80785500.0	2817742.0	79975000
998	86000250.0	2946770.0	86000000
999	86224500.0	2972592.0	85825000

1000 rows × 3 columns

Python Programming for Predicting Sales Commission Budget

- View descriptive statistics of Actual Sales, Commission Amount, and Sales Target

```
results_df.describe().style.format('{:,.2f}')
```

	Sales	Commission_Amount	Sales_Target
count	1,000.00	1,000.00	1,000.00
mean	83,802,774.50	2,860,189.98	83,809,275.00
std	2,678,024.82	100,923.70	2,657,210.07
min	74,394,750.00	2,547,040.00	74,250,000.00
25%	81,906,937.50	2,791,457.00	81,900,000.00
50%	83,657,125.00	2,853,421.00	83,712,500.00
75%	85,628,812.50	2,927,949.00	85,675,000.00
max	93,971,500.00	3,191,832.00	94,125,000.00

Python Programming for Predicting Sales Commission Budget

- Visualize the distribution of Total Commission Amount (1,000 iterations)

```
results_df['Commission_Amount'].plot(kind='hist', title="Total Commission Amount")
```