What is a Monte Carlo simulation?

MONTE CARLO SIMULATIONS IN PYTHON

Izzy Weber
Curriculum Manager, DataCamp

Simulations and Monte Carlo simulations

Simulations:

- Experiments that attempt to imitate reality
- Often use computer programs

Monte Carlo simulations:

- Used to predict the probability of different outcomes impacted by the presence of *random variables*
- Rely on repeated random sampling to obtain numerical results
- Results are stochastic since the model relies on random sampling

Simulation example

Rolling six-sided die

- Tom rolls a fair six-sided die n times
- After each roll, he records the outcome
- He then places the die in a bag and selects a new one for the next roll

Questions

1. How many dice will Tom collect after n rolls?
2. What will be the mean outcome after n rolls?

Simulating Tom's outcomes

- `total_dice` : the number of dice in Tom's bag after n rolls
- `mean_point_dice` : the mean of all outcomes after n rolls

```
import random
import numpy as np

def roll_dice(n, seed):
    random.seed(seed)
    total_dice = 0
    point_dice = []
    for i in range(n):
        total_dice += 1
        point_dice.append(random.randint(1, 6))
    mean_point_dice = np.mean(point_dice)
    return([total_dice, mean_point_dice])
```

Simulation results

Simulation One:

```
seed=1231
print(roll_dice(10, seed))
print(roll_dice(100, seed))
print(roll_dice(1000, seed))
print(roll_dice(10000, seed))
```

Results:

```
[10, 3.6]
[100, 3.5]
[1000, 3.495]
[10000, 3.503]
```

Simulation Two:

```
seed=3124
print(roll_dice(10, seed))
print(roll_dice(100, seed))
print(roll_dice(1000, seed))
print(roll_dice(10000, seed))
```

Results:

```
[10, 3.8]
[100, 3.28]
[1000, 3.474]
[10000, 3.5508]
```

The Law of Large Numbers

As the number of identically distributed, randomly generated variables increases, their sample mean approaches their theoretical mean.

Simulation Three (`seed = 3124`):

```
print(roll_dice(100000, seed))
print(roll_dice(500000, seed))
print(roll_dice(1000000, seed))
```

Results:

```
[100000, 3.51344]
[500000, 3.50428]
[1000000, 3.501995]
```

Let's practice!

MONTE CARLO SIMULATIONS IN PYTHON

Resampling as a special type of Monte Carlo simulation

MONTE CARLO SIMULATIONS IN PYTHON

Izzy Weber
Curriculum Manager, DataCamp

Resampling as a special type of Monte Carlo simulation

Monte Carlo simulations

- Sample from probability distributions
- Distributions either known or assumed
- Rely on historical data or expertise to choose proper distributions

Resampling

- Sample randomly from existing data
- Existing data is implicit probability distribution
- Assume that data is representative

Resampling methods

1. **Sampling without replacement**
   - Used to draw a random sample
2. **Sampling with replacement (or bootstrapping)**
   - Use to estimate the sampling distribution of almost any statistic
3. **Permutation**
   - Often used to compare two groups

Sampling without replacement

Randomly draw two different states of the six states in New England

```python
import random
def two_random_ne_states():
    ne_states=["Maine",
               "Vermont",
               "New Hampshire",
               "Massachusetts",
               "Connecticut",
               "Rhode Island"]
    return(random.sample(ne_states, 2))
```

```python
two_random_ne_states()
two_random_ne_states()
```

```
['Massachusetts', 'Connecticut']
['New Hampshire', 'Maine']
```

Bootstrapping

Estimate the 95% confidence interval for the mean height of NBA players

```
import random
import numpy as np

nba_heights = [196, 191, 198, 216, 188, 185, 211, 201,
               188, 191, 201, 208, 191, 183, 196]
simu_heights = []
for i in range(1000):
    bootstrap_sample = random.choices(nba_heights, k=15)
    simu_heights.append(np.mean(bootstrap_sample))
upper = np.quantile(simu_heights, 0.975)
lower = np.quantile(simu_heights, 0.025)
print([np.mean(simu_heights), lower, upper])
```

```
[196.26666666666668, 191.8, 201.2]
```

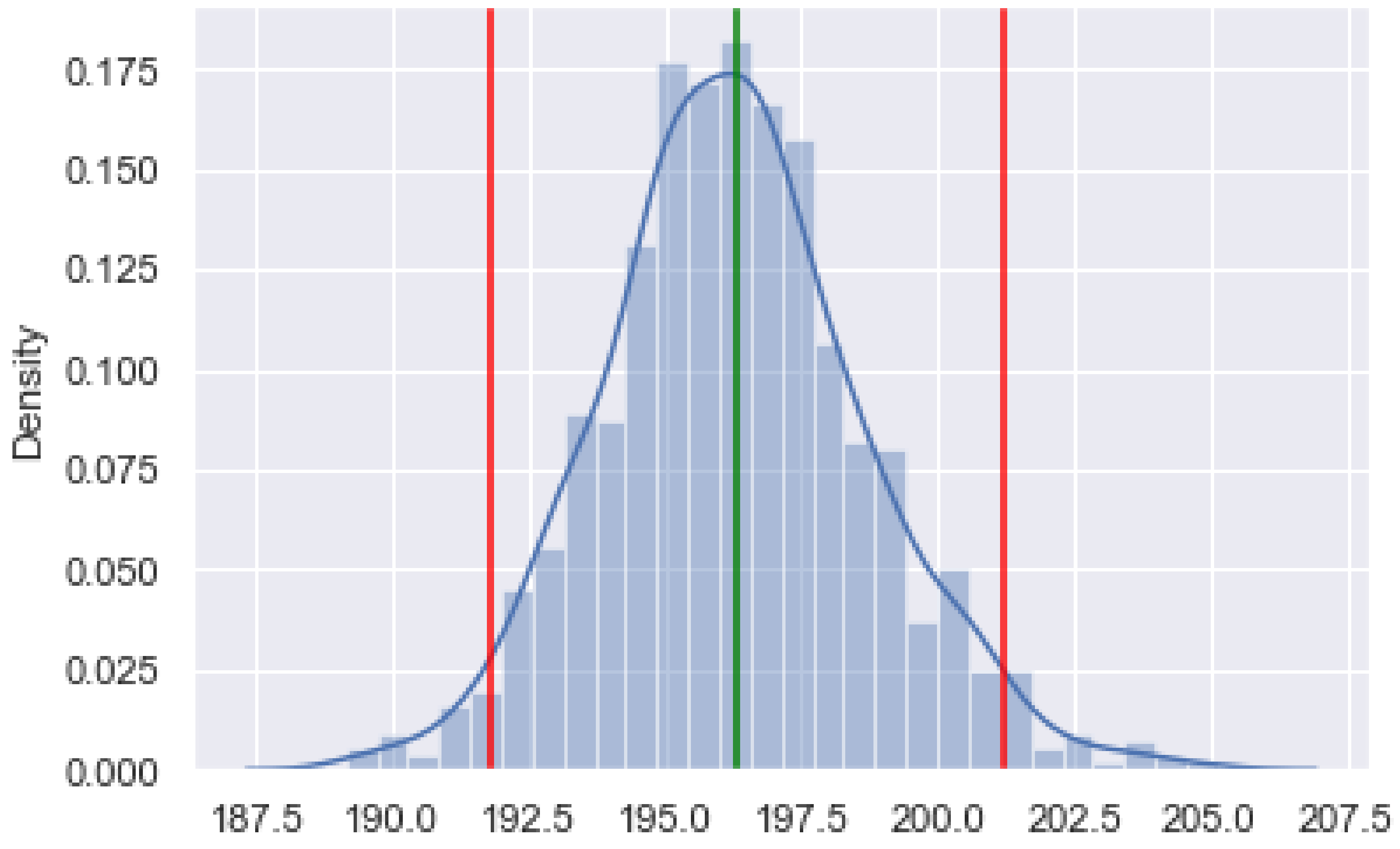
Visualization of bootstrap results

Plotting libraries:

- `seaborn`
- `matplotlib`

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

sns.displot(simu_heights)
plt.axvline(191.8, color="red")
plt.axvline(201.2, color="red")
plt.axvline(196.3, color="green")
```



Permutation

Estimate 95% confidence interval of the mean difference between heights of NBA players and US males

```
us_heights = [165, 185, 179, 187, 193, 180, 178, 179, 171, 176,
              169, 160, 140, 199, 176, 185, 175, 196, 190, 176]
nba_heights = [196, 191, 198, 216, 188, 185, 211, 201, 188, 191, 201, 208, 191, 183, 196]
all_heights = us_heights + nba_heights

simu_diff = []
for i in range(1000):
    perm_sample = np.random.permutation(all_heights)
    perm_nba, perm_adult = perm_sample[0:15], perm_sample[15:35]
    perm_diff = np.mean(perm_nba) - np.mean(perm_adult)
    simu_diff.append(perm_diff)
```

Permutation results

Difference in mean of NBA heights and adult American male heights:

```
np.mean(nba_heights) - np.mean(us_adult_height)
```

```
18.316666666666669
```

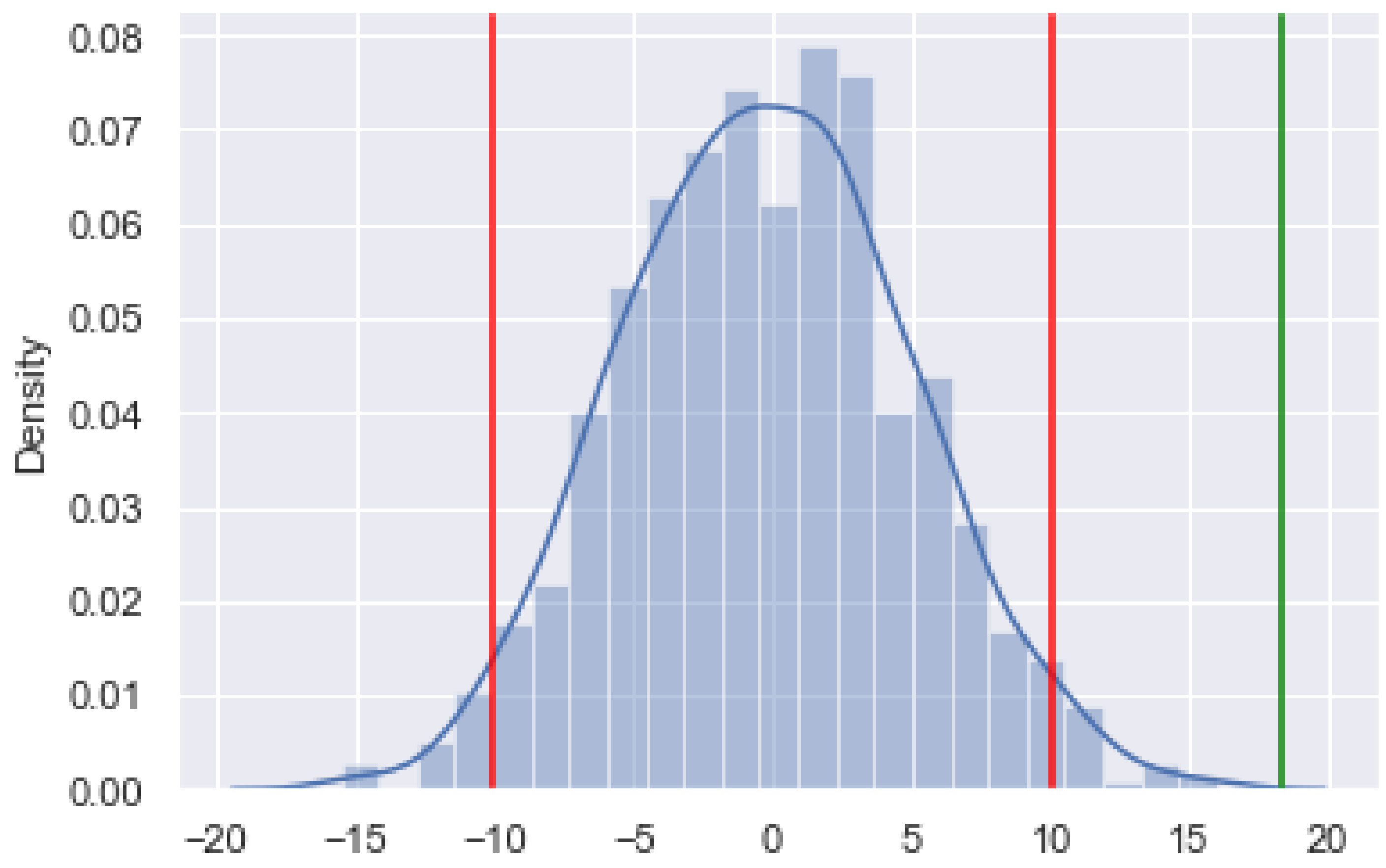
95% confidence interval for permutation of two random lists:

```
upper = np.quantile(simu_diff, 0.975)
lower = np.quantile(simu_diff, 0.025)
print([lower, upper])
```

```
[-10.033333333333331, 10.033333333333331]
```

Visualizing permutation results

```
sns.distplot(simu_diff)
plt.axvline(-10.03, color="red")
plt.axvline(10.03, color="red")
plt.axvline(18.32, color="green")
```



Let's practice!

MONTE CARLO SIMULATIONS IN PYTHON

Leveraging Monte Carlo simulations

MONTE CARLO SIMULATIONS IN PYTHON

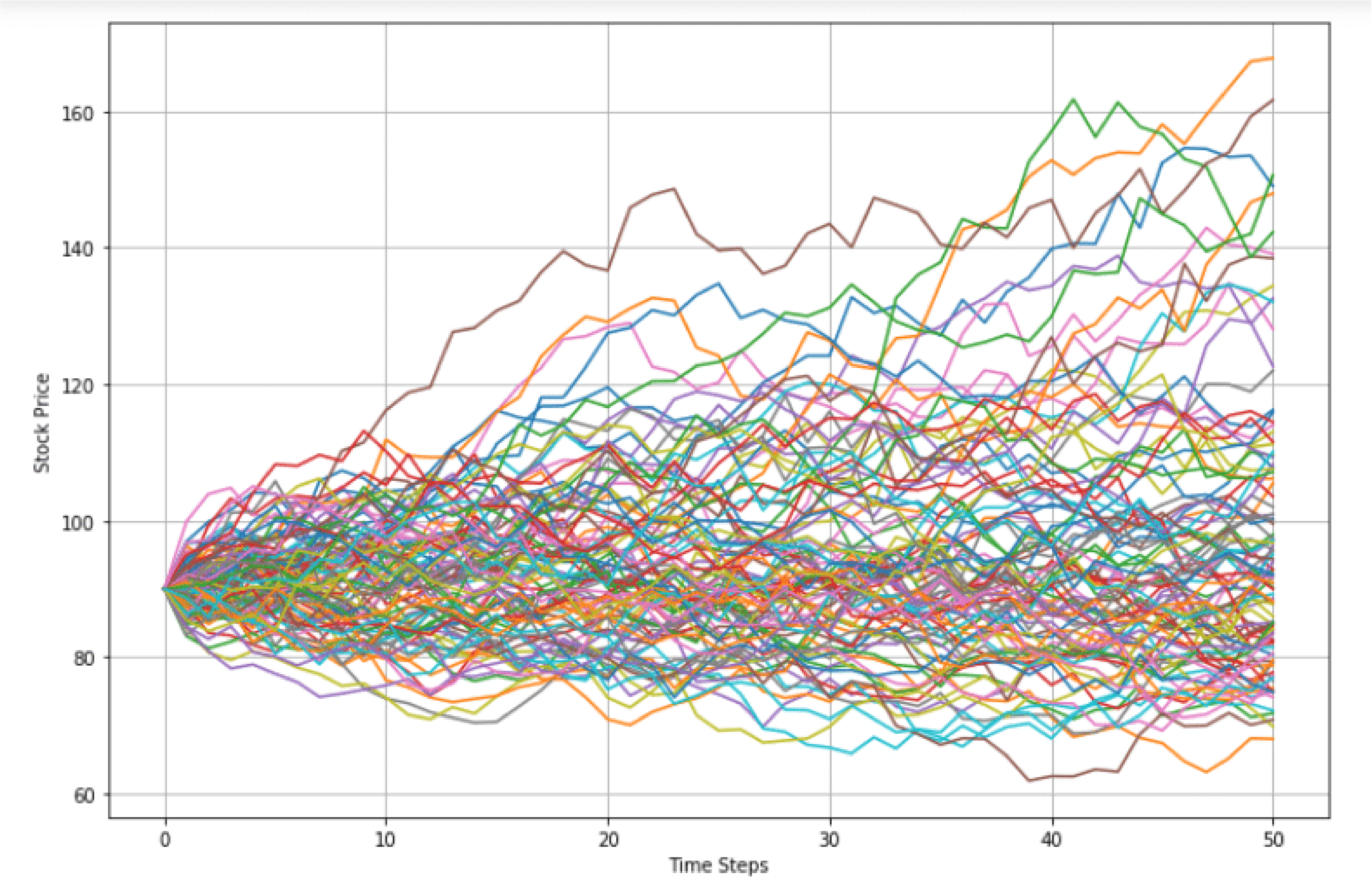
Izzy Weber
Curriculum Manager, DataCamp

Wide applicability

Monte Carlo simulations are used in fields such as...

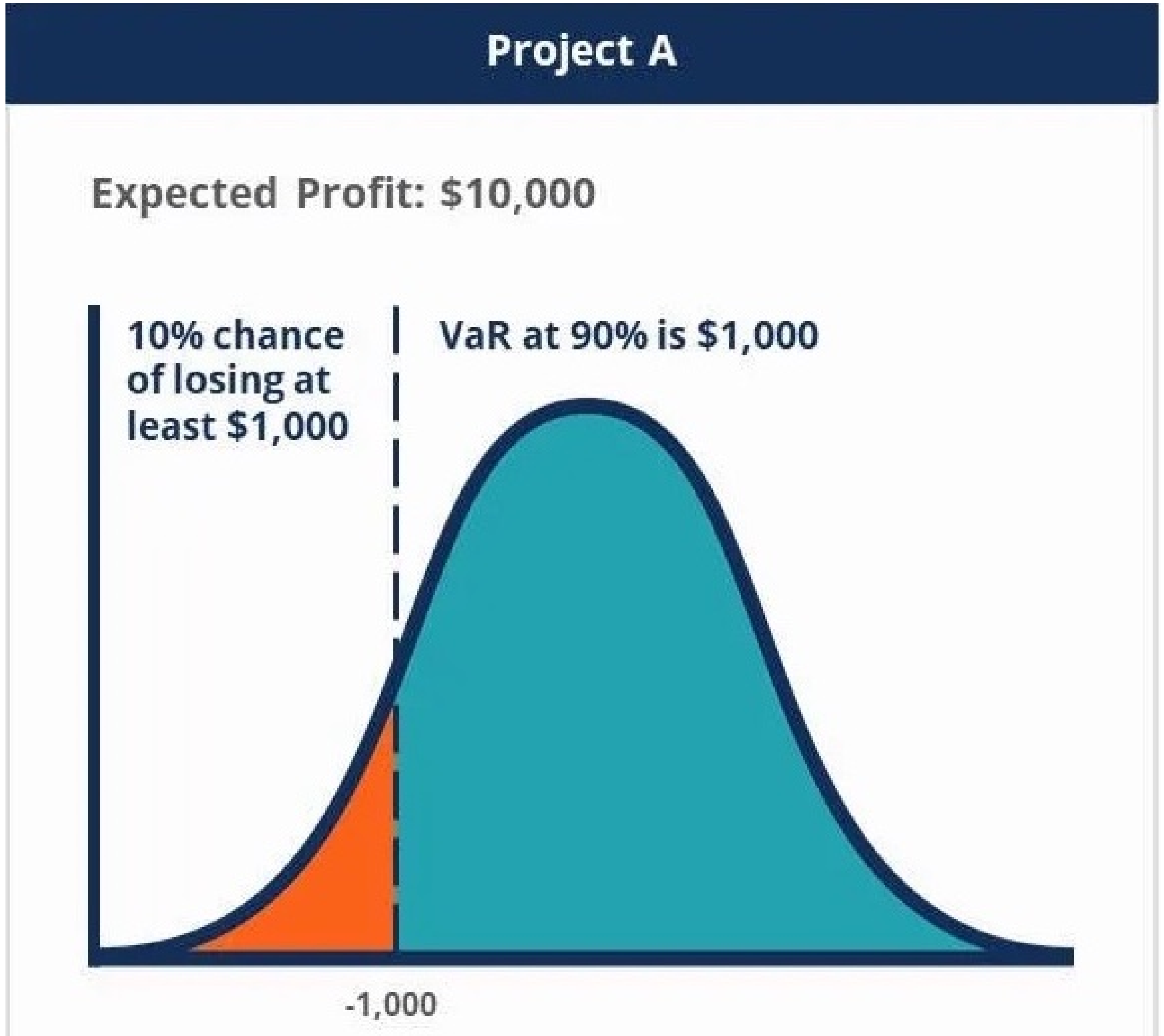
- Finance and business
- Engineering
- Physical sciences

Stock price prediction



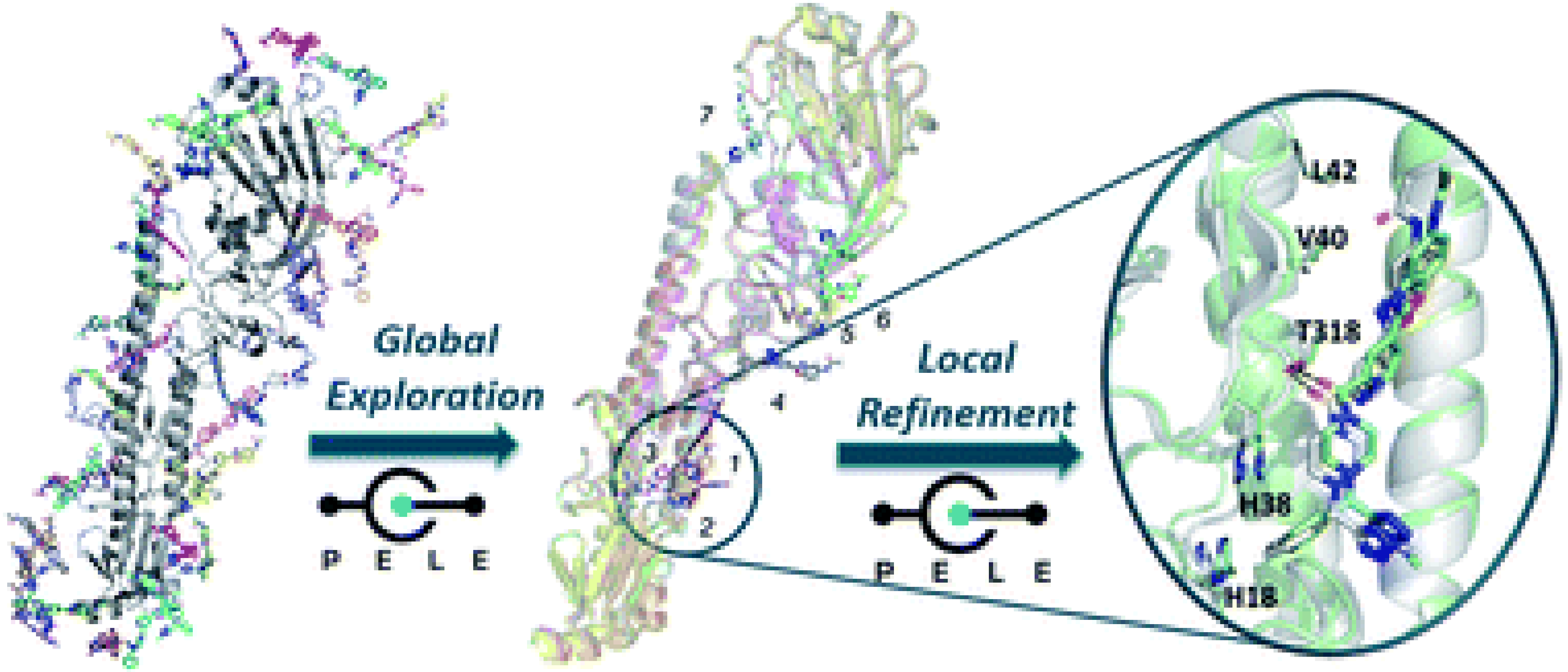
[1] https://marketxls.com/monte-carlo-simulation-excel

Risk management



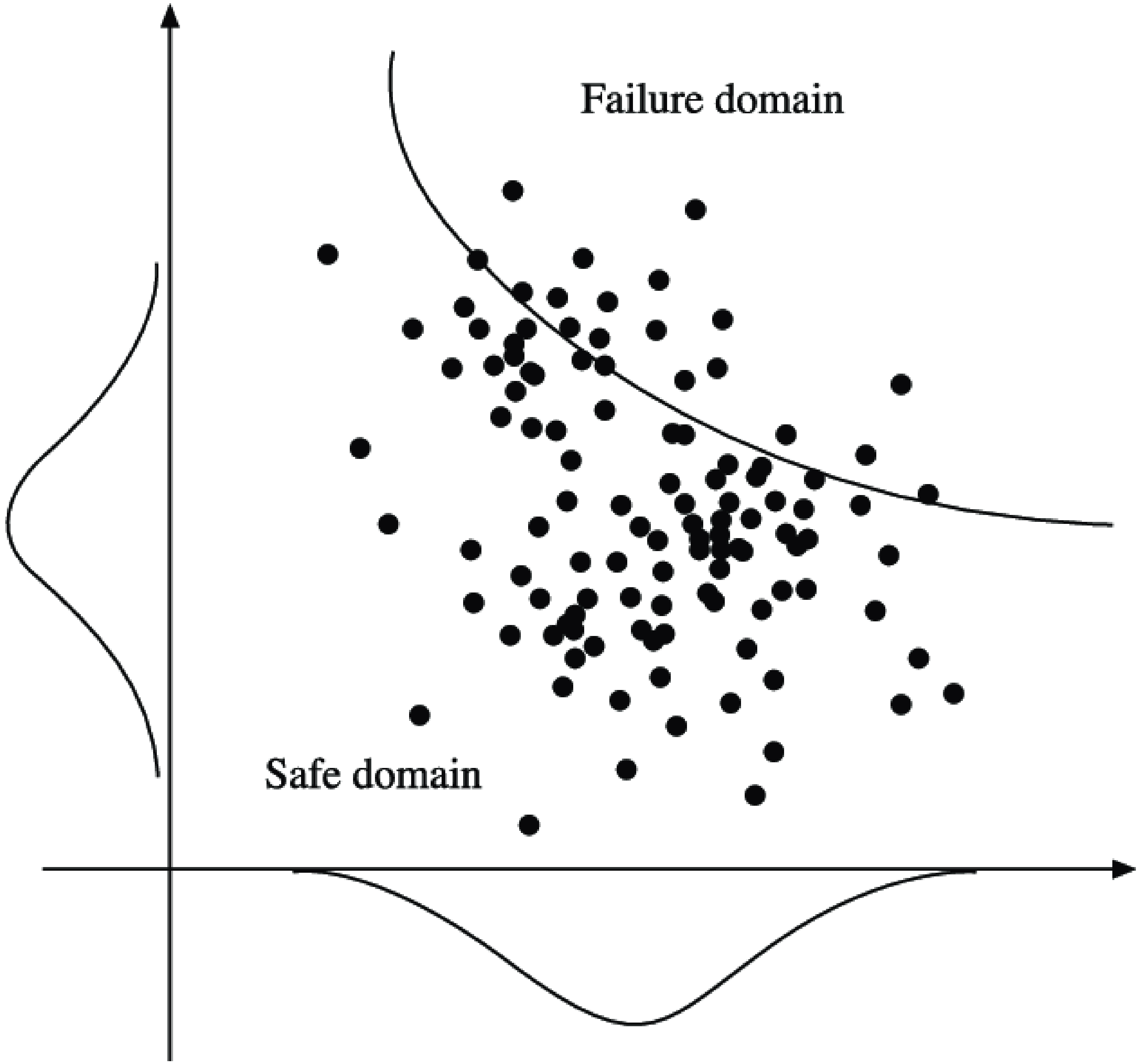
[1] https://www.investopedia.com/articles/04/092904.asp https://corporatefinanceinstitute.com/course/modeling-risk-monte-carlo-simulation/

Binding site identification



[1] https://pubs.rsc.org/en/content/articlelanding/2020/ra/d0ra01127d

Reliability analysis in engineering



[1] www.researchgate.net/publication/228814883_Probabilistic_Transformation_Method_in_Reliability_Analysis

Benefits of Monte Carlo Simulations

- Take into consideration a range of values for various inputs
- Show not only what could happen, but how likely each outcome is
- Make it easy to visualize the range of possible outcomes
- Can examine what would have happened under different circumstances

Bags of biased dice

Roll two separate dice from two bags, each containing three biased dice:

```
bag1 = [[1, 2, 3, 6, 6, 6], [1, 2, 3, 4, 4, 6], [1, 2, 3, 3, 3, 5]]
bag2 = [[2, 2, 3, 4, 5, 6], [3, 3, 3, 4, 4, 5], [1, 1, 2, 4, 5, 5]]
```

Simulation:

- Pick one die from each bag randomly; roll both dice
- Success if the dice outcomes add up to eight; otherwise, failure
- We want to calculate the probability of success for each unique combination of dice

Biased dice simulation

```
def roll_biased_dice(n):
    results = {}
    for i in range(n):
        bag_index1 = random.randint(0, 2)
        die_index1 = random.randint(0, 5)
        bag_index2 = random.randint(0, 2)
        die_index2 = random.randint(0, 5)
        point1 = bag1[bag_index1][die_index1]
        point2 = bag2[bag_index2][die_index2]
        key = "%s_%s" % (point1, point2)
        if point1 + point2 == 8:
            if key not in results:
                results[key] = 1
            else:
                results[key] += 1
```

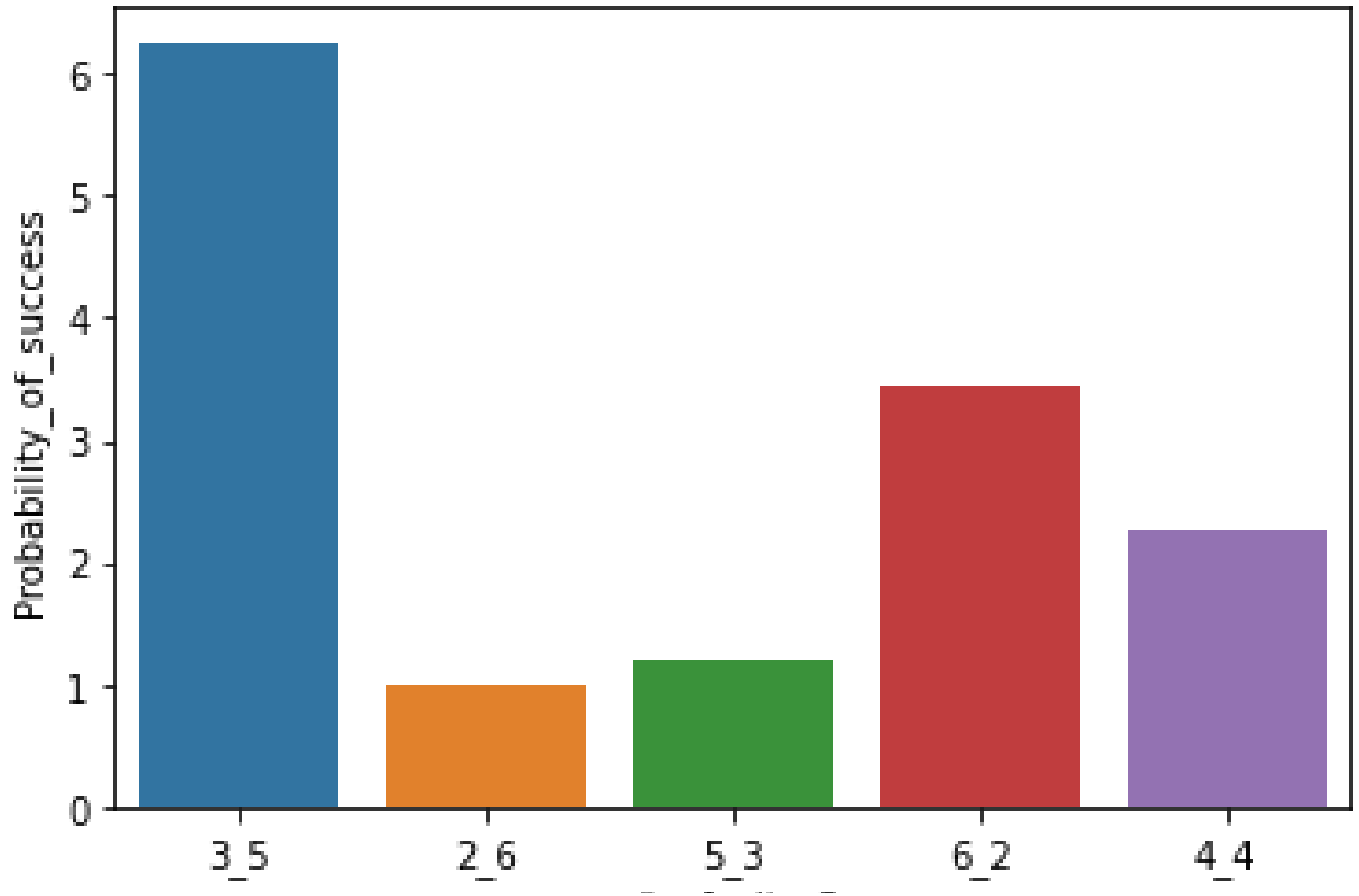
Biased dice results

dice1_dice2	probability_of_success
6_2	5.54
3_5	2.67
2_6	1.45
4_4	4

Biased dice results

```
bag1 = [[1, 2, 3, 6, 6, 6], [1, 2, 3, 4, 4, 6], [1, 2, 3, 3, 3, 5]]
bag2 = [[2, 2, 3, 4, 5, 6], [3, 3, 3, 4, 4, 5], [1, 1, 2, 4, 5, 5]]
```

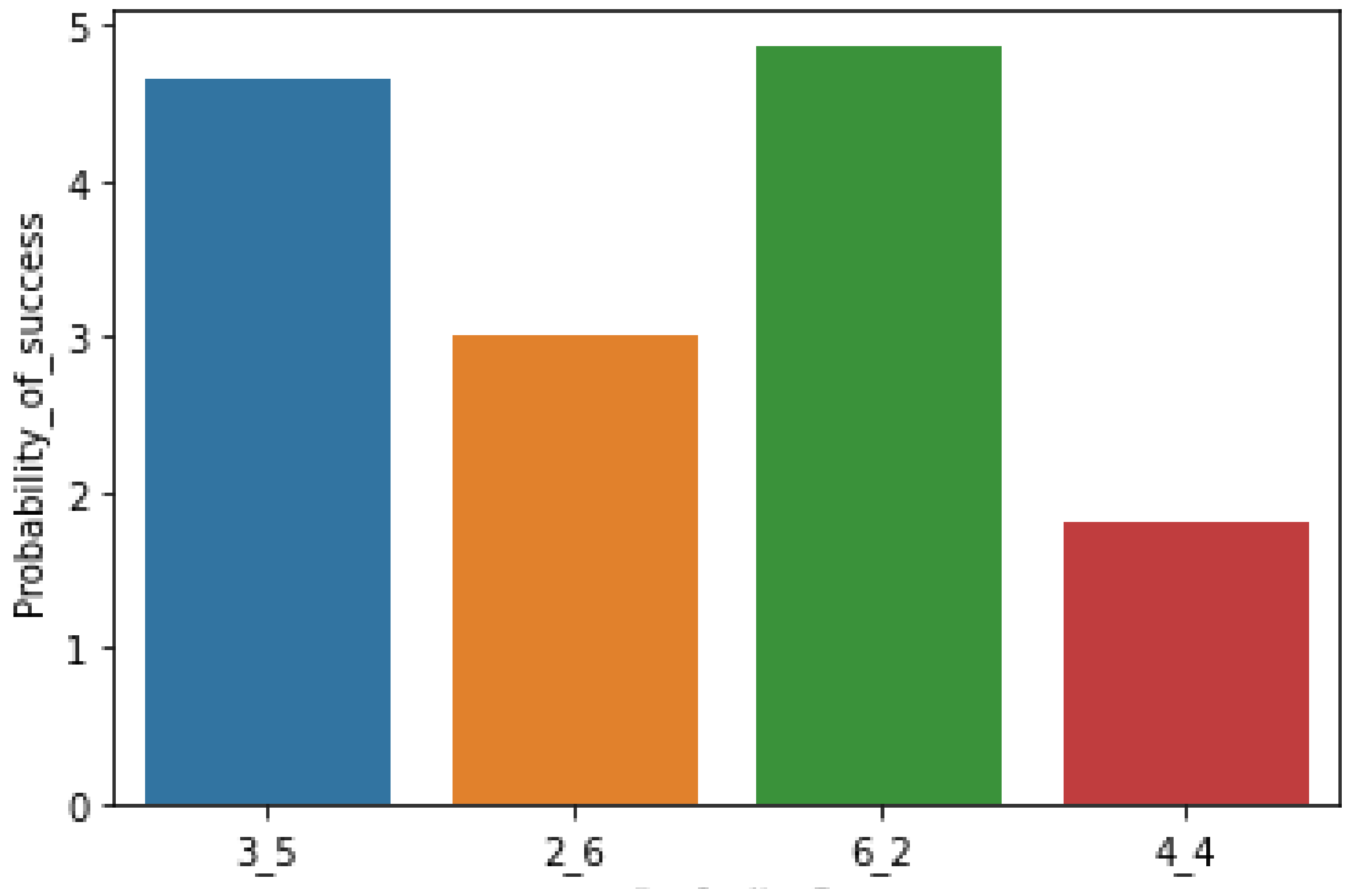
Simulation results of 10,000 trials:



Biased dice results

```
bag1 = [[2, 2, 3, 4, 6, 6], [1, 2, 2, 4, 6, 6], [1, 2, 3, 3, 3, 3]]
bag2 = [[1, 2, 3, 4, 5, 6], [1, 3, 3, 4, 4, 6], [2, 2, 2, 3, 5, 5]]
```

Simulation results of 10,000 trials:



Limitations of Monte Carlo simulations

- Model output is only as good as model input
- Probability of extreme events is often underestimated

Let's practice!

MONTE CARLO SIMULATIONS IN PYTHON