Let's flip a coin in Python

FOUNDATIONS OF PROBABILITY IN PYTHON

Alexander A. Ramírez M.
CEO @ Synergy Vision

Probability

- Foundation of Data Science
- Allows to produce data from models
- Study regularities in random phenomena

Gain intuition

...with coin flips

Only two outcomes

Heads or Tails

Flipping a coin in Python

Bernoulli random experiment

```
from scipy.stats import bernoulli
bernoulli.rvs(p=0.5, size=1)
```

```
array([0])
```

Another draw

```
bernoulli.rvs(p=0.5, size=1)
```

```
array([1])
```

Flipping multiple coins

Change `size` parameter to flip more...

```
bernoulli.rvs(p=0.5, size=10)
```

```
array([0, 0, 0, 0, 0, 0, 1, 1, 0, 0])
```

How many heads?

```
sum(bernoulli.rvs(p=0.5, size=10))
```

```
5
```

Flipping multiple coins (Cont.)

Another draw...

```
sum(bernoulli.rvs(p=0.5, size=10))
```

```
2
```

Flipping multiple coins (Cont.)

Binomial random variable

```
from scipy.stats import binom
binom.rvs(n=10, p=0.5, size=1)
```

```
array([7])
```

Many draws

```
binom.rvs(n=10, p=0.5, size=10)
```

```
array([6, 2, 3, 5, 5, 5, 5, 4, 6, 6])
```

Flipping multiple coins (Cont.)

Biased coin draws

```
binom.rvs(n=10, p=0.3, size=10)
```

```
array([3, 4, 3, 3, 2, 2, 2, 2, 3, 6])
```

Random generator seed

- Use the `random_state` parameter of the `rvs()` function

```
from scipy.stats import binom
binom.rvs(n=10, p=0.5, size=1, random_state=42)
```

- Use `numpy.random.seed()`

```
import numpy as np
np.random.seed(42)
```

Random generator seed (Cont.)

Flipping 10 fair coins with a random seed

```
from scipy.stats import binom
import numpy as np


np.random.seed(42)
binom.rvs(n=10, p=0.5, size=1)
```

```
array([4])
```

Let's practice flipping coins in Python

FOUNDATIONS OF PROBABILITY IN PYTHON

Probability mass and distribution functions

FOUNDATIONS OF PROBABILITY IN PYTHON

Alexander A. Ramírez M.
CEO @ Synergy Vision

Probability mass function (pmf)

$$binomial.pmf(k, n, p) = \binom{n}{k} p^k (1-p)^{n-k}$$

Probability mass function (pmf)

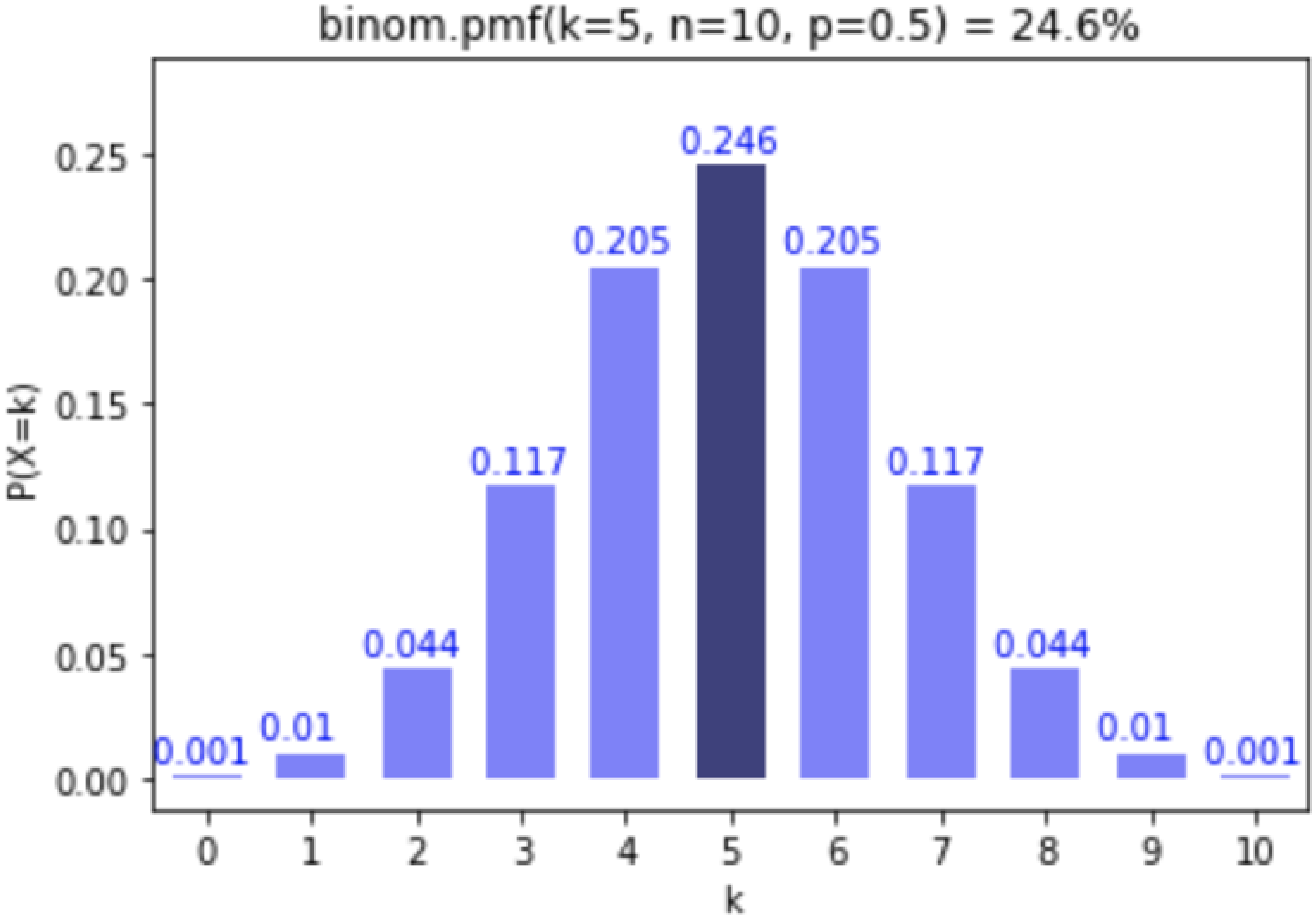
$$binomial.pmf(k, n, p) = \binom{n}{k} p^k (1-p)^{n-k}$$

Probability mass function (pmf) (Cont.)

$$binomial.pmf(k,n,p) = \binom{n}{k} {\color{red}p^k} (1-p)^{n-k}$$

Probability mass function (pmf) (Cont.)

$$binomial.pmf(k, n, p) = \binom{n}{k} p^k (1-p)^{n-k}$$


binom.pmf(k=5, n=10, p=0.5) = 24.6%
0.25
0.20
0.15
0.10
0.05
0.00
P(X=k)
0.001
0.01
0.044
0.117
0.205
0.246
0.205
0.117
0.044
0.01
0.001
0
1
2
3
4
5
6
7
8
9
10
k

Probability mass function (pmf) (Cont.)

$$binomial.pmf(k, n, p) = \binom{n}{k} p^k (1-p)^{n-k}$$

In **Python**:

```
binom.pmf(k, n, p)
```

Calculating probabilities with `binom.pmf()`

```
# Probability of 2 heads after 10 throws with a fair coin
binom.pmf(k=2, n=10, p=0.5)
```

```
0.04394531249999999
```

```
# Probability of 5 heads after 10 throws with a fair coin
binom.pmf(k=5, n=10, p=0.5)
```

```
0.24609375000000025
```

Calculating probabilities with binom.pmf() (Cont.)

```
# Probability of 50 heads after 100 throws with p=0.3
binom.pmf(k=50, n=100, p=0.3)
```

```
1.3026227131445298e-05
```

```
# Probability of 65 heads after 100 throws with p=0.7
binom.pmf(k=65, n=100, p=0.7)
```

```
0.0467796823527298
```

Probability distribution function (cdf)

$$binomial.cdf(k,n,p) = \binom{n}{0}p^0(1-p)^n + \binom{n}{1}p(1-p)^{n-1} + ... + \binom{n}{k}p^k(1-p)^{n-k}$$

Probability distribution function (cdf) (Cont.)

$$binomial.cdf(k, n, p) = \binom{n}{0} p^0 (1-p)^n + \binom{n}{1} p (1-p)^{n-1} + ... + \binom{n}{k} p^k (1-p)^{n-k}$$

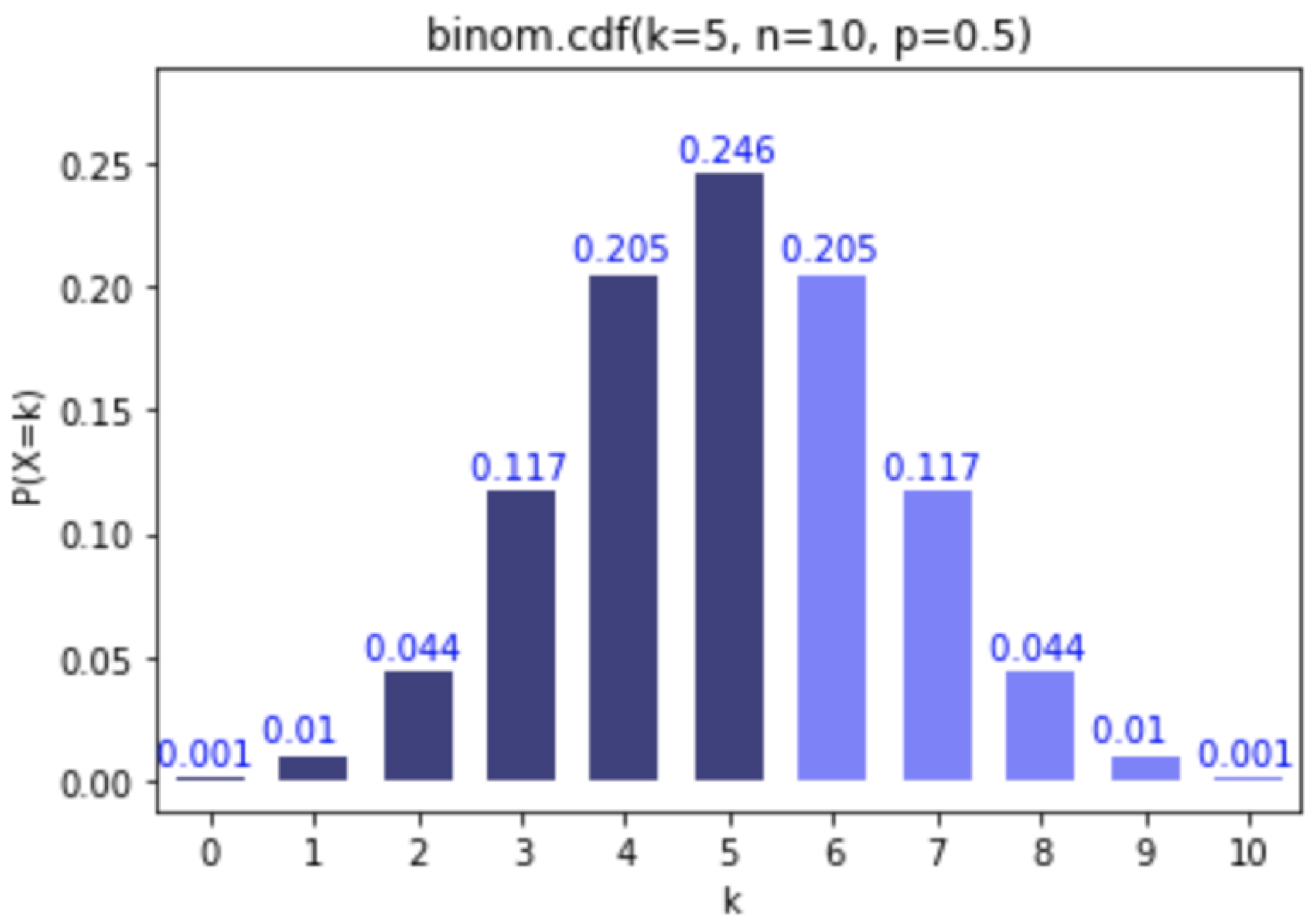
Probability distribution function (cdf) (Cont.)

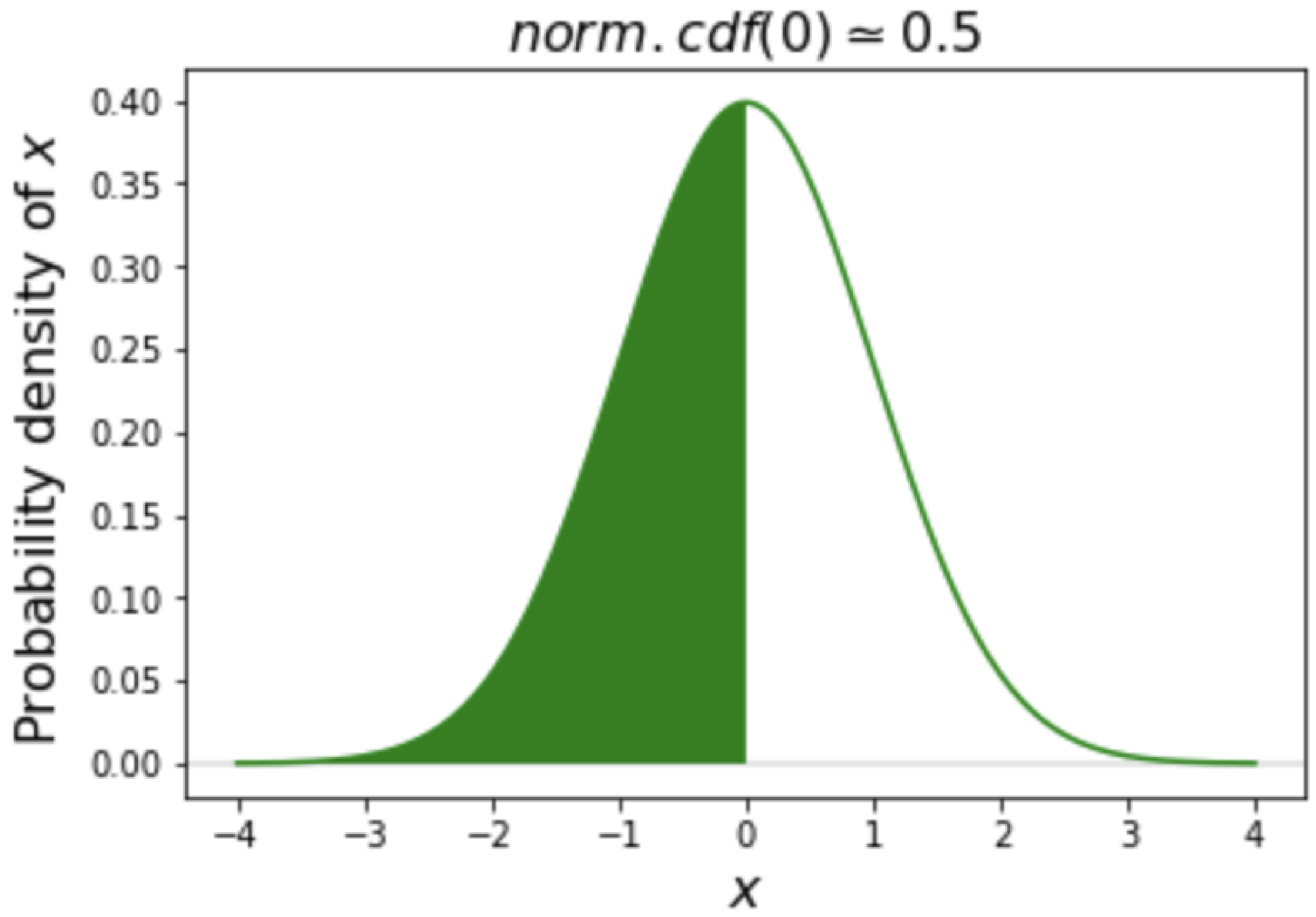
$$binomial.cdf(k,n,p) = \binom{n}{0} p^0 (1-p)^n + \binom{n}{1} p(1-p)^{n-1} + ... + \binom{n}{k} p^k (1-p)^{n-k}$$

Probability distribution function (cdf) (Cont.)

$$binomial.cdf(k,n,p) = \binom{n}{0}p^0(1-p)^n + \binom{n}{1}p(1-p)^{n-1} + ... + \binom{n}{k}p^k(1-p)^{n-k}$$

Cumulative distribution function (cdf)





Cumulative distribution function (cdf) (Cont.)

$$binomial.cdf(k,n,p) = \binom{n}{0}p^0(1-p)^n + \binom{n}{1}p(1-p)^{n-1} + ... + \binom{n}{k}p^k(1-p)^{n-k}$$

In **Python:**

```
binom.cdf(k=1, n=3, p=0.5)
```

Calculating cumulative probabilities

```
# Probability of 5 heads or less after 10 throws with a fair coin
binom.cdf(k=5, n=10, p=0.5)
```

```
0.6230468749999999
```

```
# Probability of 50 heads or less after 100 throws with p=0.3
binom.cdf(k=50, n=100, p=0.3)
```

```
0.9999909653138043
```

Calculating cumulative probabilities (Cont.)

```
# Probability of more than 59 heads after 100 throws with p=0.7
1-binom.cdf(k=59, n=100, p=0.7)
```

```
0.9875015928335618
```

```
# Probability of more than 59 heads after 100 throws with p=0.7
binom.sf(k=59, n=100, p=0.7)
```

```
0.9875015928335618
```

Let's calculate some probabilities

FOUNDATIONS OF PROBABILITY IN PYTHON

Expected value, mean, and variance

FOUNDATIONS OF PROBABILITY IN PYTHON

Alexander A. Ramírez M.
CEO @ Synergy Vision

Expected value

Expected value: sum of possible outcomes weighted by it's probability.

$$E(X) = \sum_{i=1}^{k} x_i p_i = x_1 p_1 + x_2 p_2 + \cdots + x_k p_k$$

Expected value

The expected value of a discrete random variable is the sum of the possible outcomes weighted by their probability.

$$E(X) = \sum_{i=1}^{k} x_i p_i = x_1 p_1 + x_2 p_2 + \cdots + x_k p_k$$

In our case, for the coin flip we get:

$$E(X) = \sum_{i=1}^{2} x_i p_i = x_1 p_1 + x_2 p_2 = 0 \times (1 - p) + 1 \times p = p$$

Expected value (Cont.)

The expected value of a discrete random variable is the sum of the possible outcomes weighted by their probability.

$$E(X) = \sum_{i=1}^{k} x_i p_i = x_1 p_1 + x_2 p_2 + \cdots + x_k p_k$$

In our case, for the coin flip we get:

$$E(X) = \sum_{i=1}^{2} x_i p_i = x_1 p_1 + x_2 p_2 = 0 \times (1 - p) + \color{red}{1 \times p} = p$$

Arithmetic mean

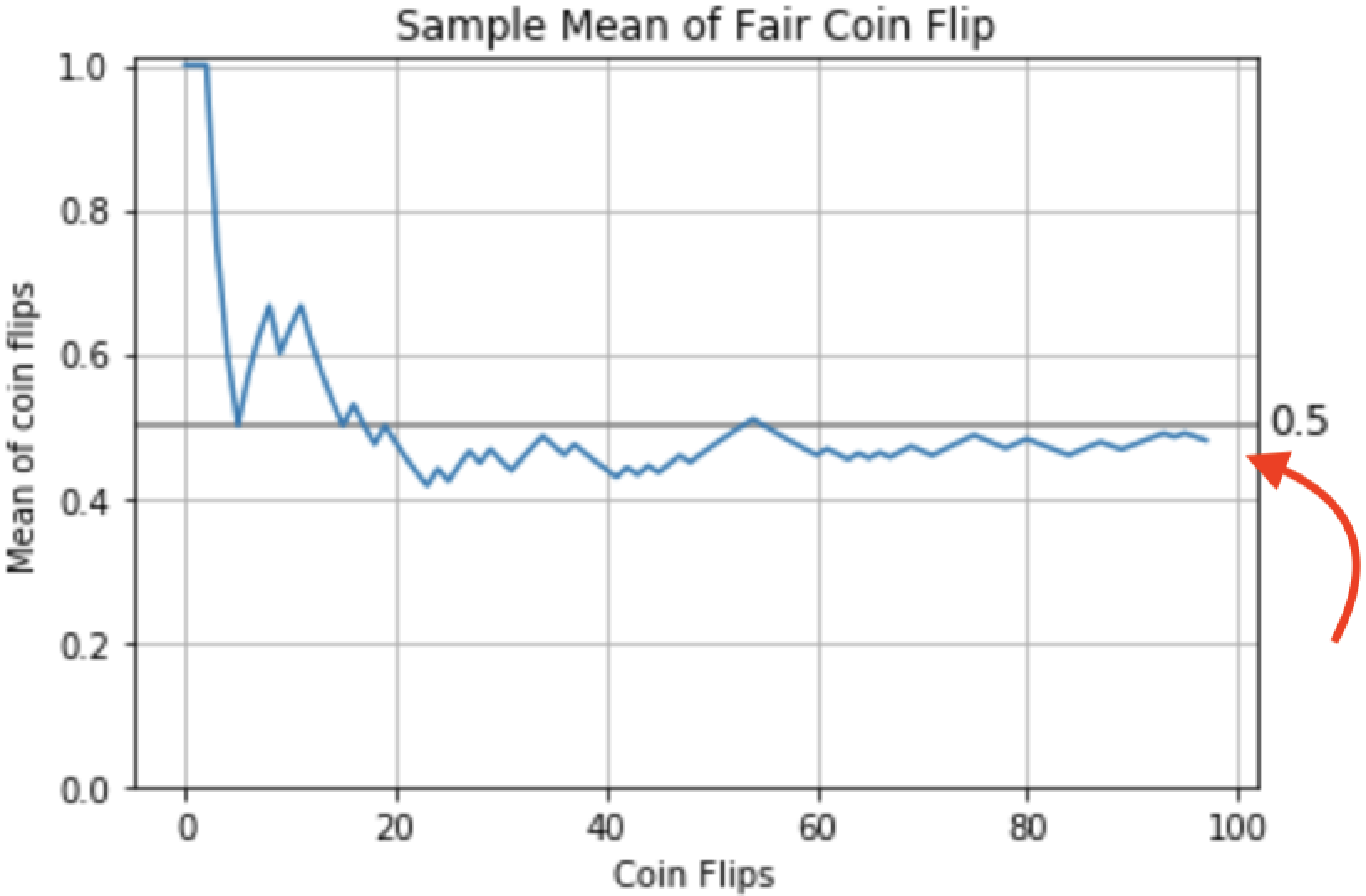
Each x_i is the outcome from one experiment (i.e., a coin flip, either 0 or 1).

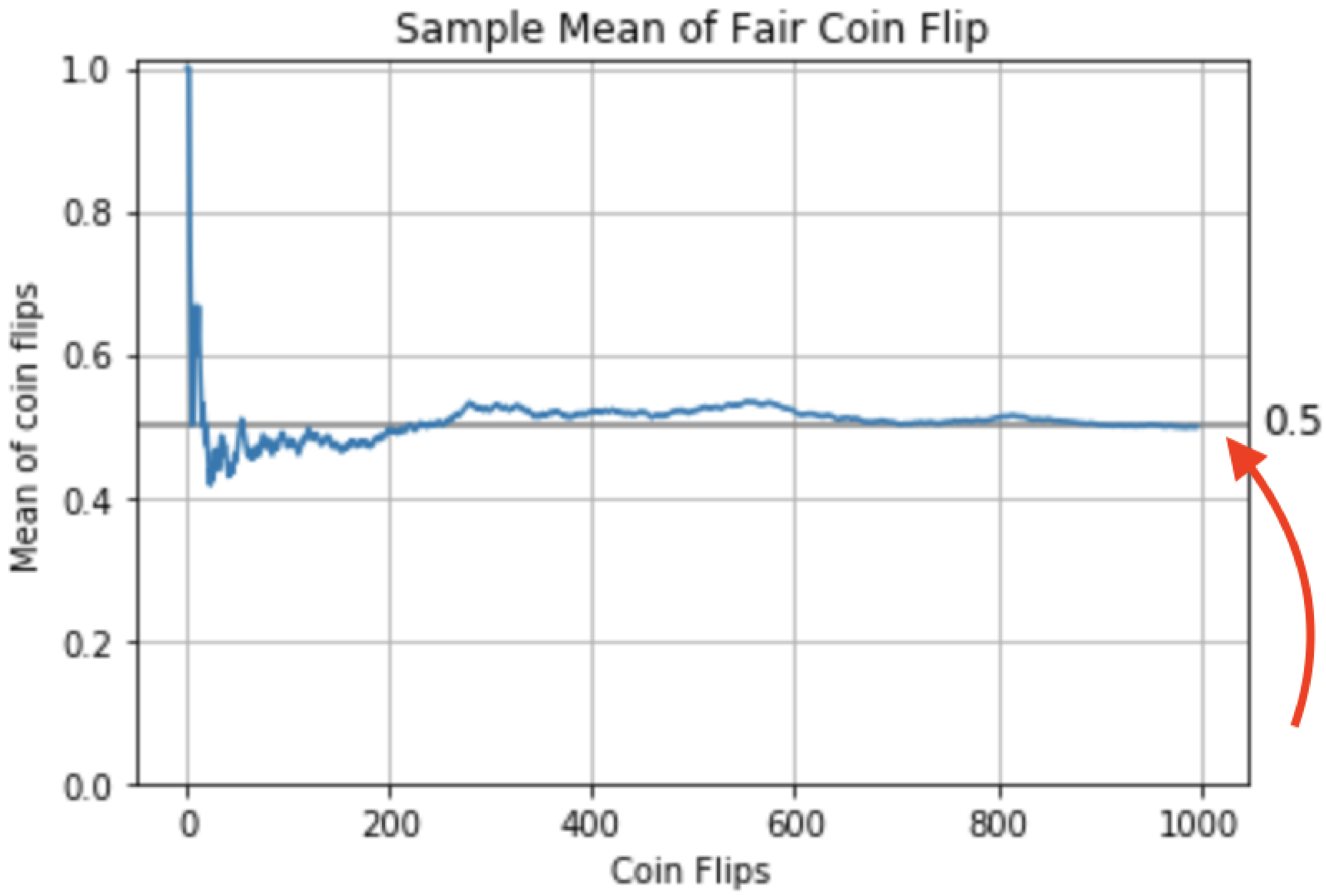
$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{1}{n}(x_1 + x_2 + \cdots + x_n)$$

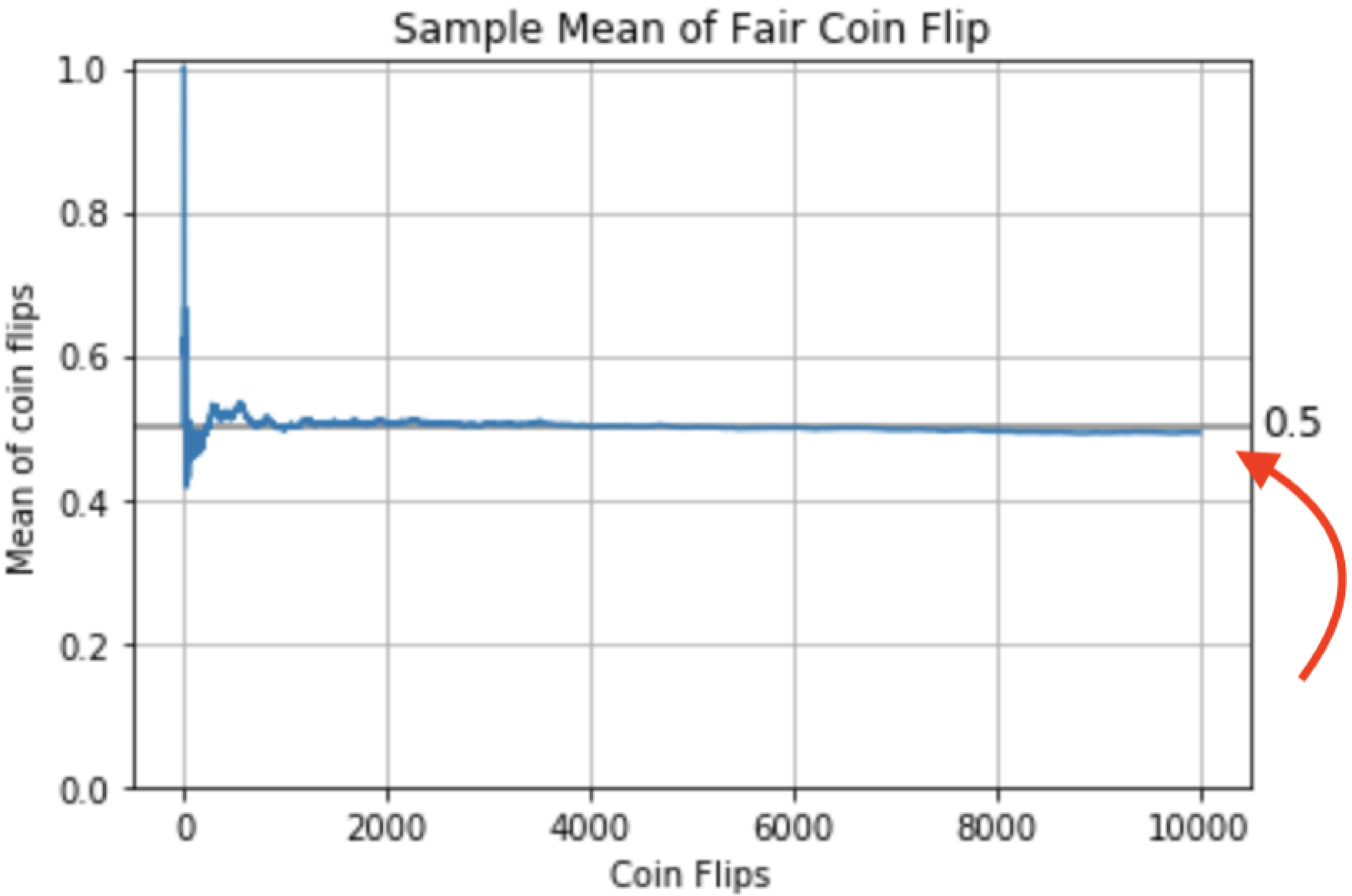
In **Python** we will use the `scipy.stats.describe()` function to get the arithmetic mean.

```
from scipy.stats import describe
describe([0,1]).mean
```

```
0.5
```


Sample Mean of Fair Coin Flip
1.0
0.8
0.6
0.4
0.2
0.0
Mean of coin flips
0
20
40
60
80
100
Coin Flips
0.5


Sample Mean of Fair Coin Flip
1.0
0.8
0.6
0.4
0.2
0.0
Mean of coin flips
0
200
400
600
800
1000
Coin Flips
0.5


Sample Mean of Fair Coin Flip
1.0
0.8
0.6
0.4
0.2
0.0
Mean of coin flips
0
2000
4000
6000
8000
10000
Coin Flips
0.5

Variance

Variance is a measure of dispersion.

It's the expected value of the squared deviation from its expected value.

$$Var(X) = E[(X - E(X))^2] = \sum_{i=1}^{n} p_i \times (x_i - E(X))^2$$

In **Python**, we will use the `scipy.stats.describe()` function to get the sample variance.

```
describe([0,1]).variance
```

```
0.5
```

Binomial distribution expected value and variance

For $X \sim Binomial(n, p)$

$$E(X) = n \times p$$

$$Var(X) = n \times p \times (1 - p)$$

Example: $n = 10$ and $p = 0.5$

- $E(X) = 10 \times 0.5 = 5$
- $Var(X) = 10 \times 0.5 \times 0.5 = 2.5$

Binomial distribution expected value and variance (Cont.)

In **Python** we will use the `binom.stats()` method to get the expected value and variance.

```
binom.stats(n=10, p=0.5)
```

```
(array(5.), array(2.5))
```

Binomial distribution expected value and variance (Cont.)

What are the expected value and variance for one fair coin flip?

```
binom.stats(n=1, p=0.5)
```

```
(array(0.5), array(0.25))
```

What are the expected value and variance for one biased coin flip, with 30% probability of success?

```
binom.stats(n=1, p=0.3)
```

```
(array(0.3), array(0.21))
```

Binomial distribution expected value and variance (Cont.)

What are the expected value and variance for 10 fair coin flips?

```
binom.stats(n=10, p=0.5)
```

```
(array(5.), array(2.5))
```

Let's calculate expected values and variance from data

FOUNDATIONS OF PROBABILITY IN PYTHON