From sample mean to population mean

FOUNDATIONS OF PROBABILITY IN PYTHON

Alexander A. Ramírez M.
CEO @ Synergy Vision

Sample mean review

LAW OF LARGE NUMBERS

> The sample mean approaches the expected value as the sample size increases.

Sample mean review (Cont.)

$$\text{Sample mean} = \bar{X}_2 = \frac{x_1 + x_2}{2}$$

Sample mean review (Cont.)

$$\text{Sample mean} = \bar{X}_3 = \frac{x_1 + x_2 + x_3}{3}$$

Sample mean review (Cont.)

$$\text{Sample mean} = \bar{X}_n = \frac{x_1 + x_2 + \cdots + x_n}{n}$$

Sample mean review (Cont.)

$$\text{Sample mean} = \bar{X}_n = \frac{x_1 + x_2 + \cdots + x_n}{n} \to \mathbb{E}(\mathbb{X})$$

Generating the sample

```
# Import binom and describe
from scipy.stats import binom
from scipy.stats import describe

# Sample of 250 fair coin flips
samples = binom.rvs(n=1, p=0.5, size=250, random_state=42)
```

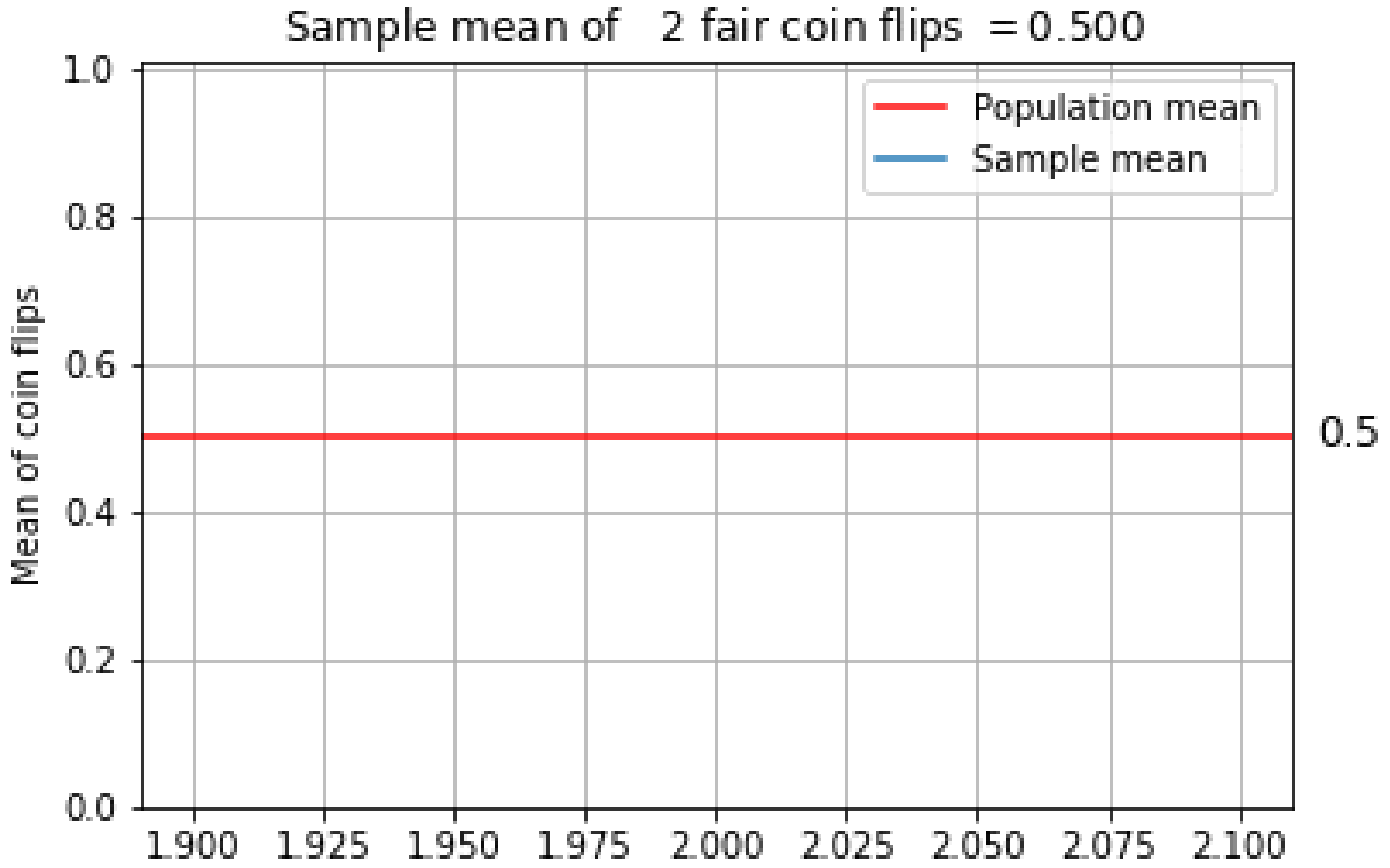
```
# Print first 100 values from the sample
print(samples[0:100])
```

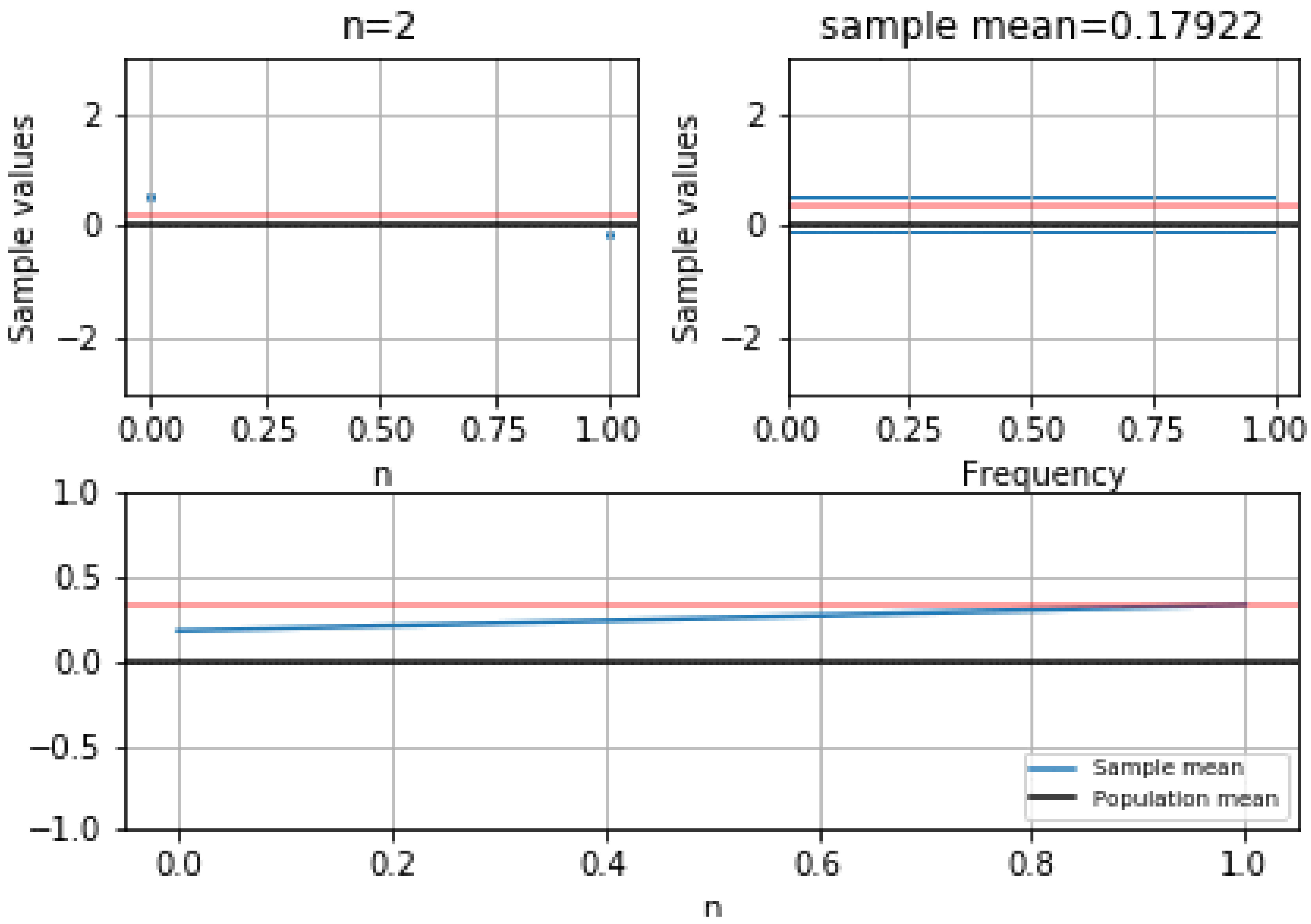
```
[0 1 1 1 0 0 0 1 1 1 0 1 1 0 0 0 0 1 0 0 1 0 0 0 0 1 0 1 1 0 1 0 0 1 1 1 0
 0 1 0 0 0 0 1 0 1 0 1 1 0 1 1 1 1 1 1 0 0 0 0 0 0 1 0 0 1 0 1 0 1 1 0 0 1
 1 1 1 0 0 0 1 1 0 0 0 0 1 1 1 0 0 1 1 1 1 0 1 0 0 0]
```

Calculating the sample mean

```
# Calculate the sample mean
print(describe(samples[0:10]).mean)
```

```
0.6
```


Sample mean of 2 fair coin flips = 0.500
Population mean
Sample mean
Mean of coin flips
1.0
0.8
0.6
0.4
0.2
0.0
1.900
1.925
1.950
1.975
2.000
2.025
2.050
2.075
2.100
0.5


n=2
sample mean=0.17922
Sample values
Sample values
n
Frequency
n
Sample mean
Population mean

Plotting the sample mean

```
from scipy.stats import binom
from scipy.stats import describe
import matplotlib.pyplot as plt

# Define our variables
coin_flips, p, sample_size , averages = 1, 0.5, 1000, []
```

```
# Generate the sample
samples = binom.rvs(n=coin_flips, p=p, size=sample_size, random_state=42)
```

Plotting the sample mean (Cont.)

```
# Calculate the sample mean
for i in range(2,sample_size+1):
    averages.append(describe(samples[0:i]).mean)

# Print the first values of averages
print(averages[0:10])
```

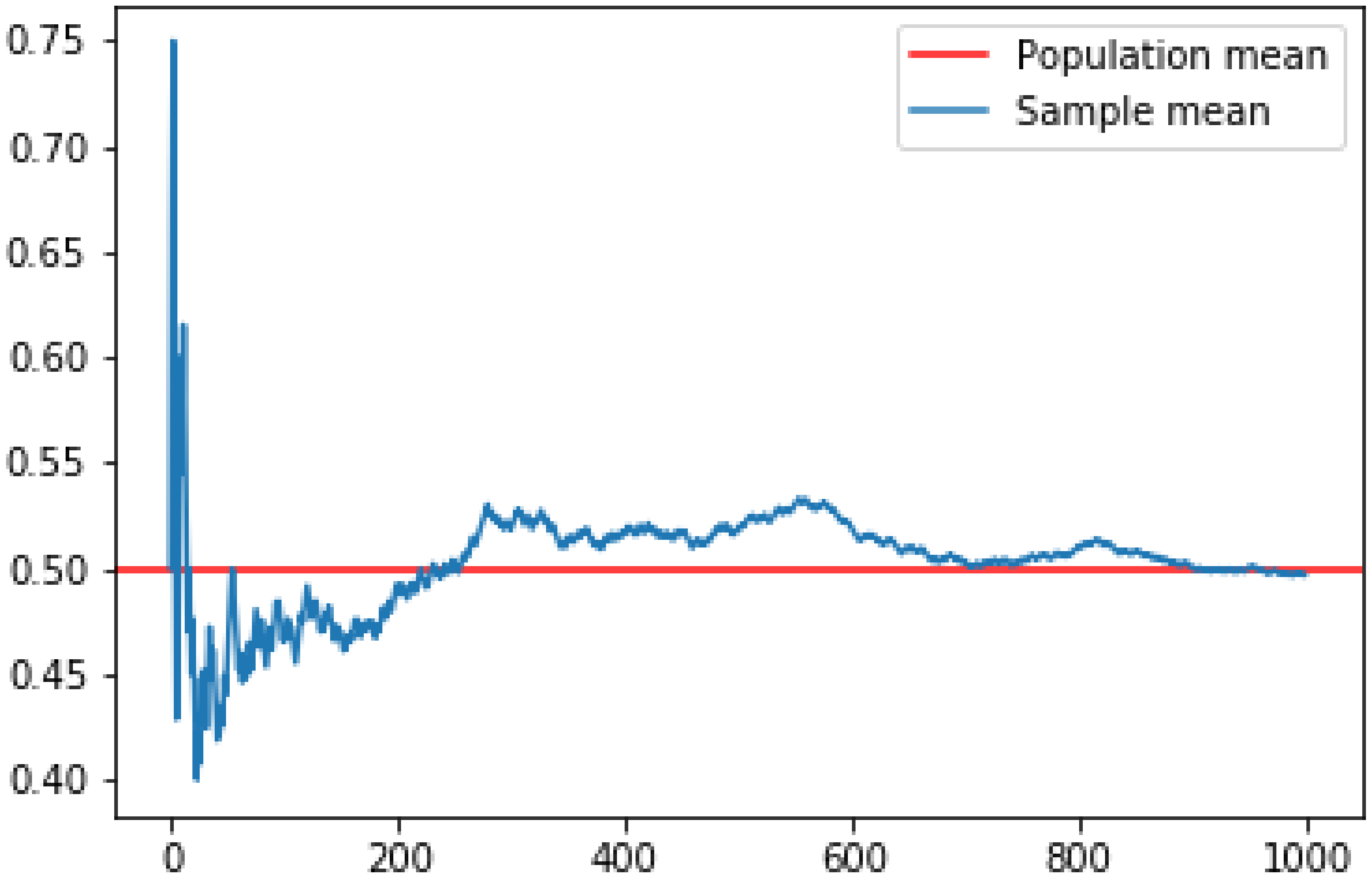
```
[0.5, 0.6666666666666666, 0.75, 0.6, 0.5, 0.42857142857142855, 0.5, 0.5555555555555555
```

Plotting the sample mean (Cont.)

```
# Add population mean line and sample mean plot
plt.axhline(binom.mean(n=coin_flips, p=p), color='red')
plt.plot(averages, '-')
```

```
# Add legend
plt.legend(("Population mean","Sample mean"), loc='upper right')
plt.show()
```

Sample mean plot



Let's practice!

FOUNDATIONS OF PROBABILITY IN PYTHON

Adding random variables

FOUNDATIONS OF PROBABILITY IN PYTHON

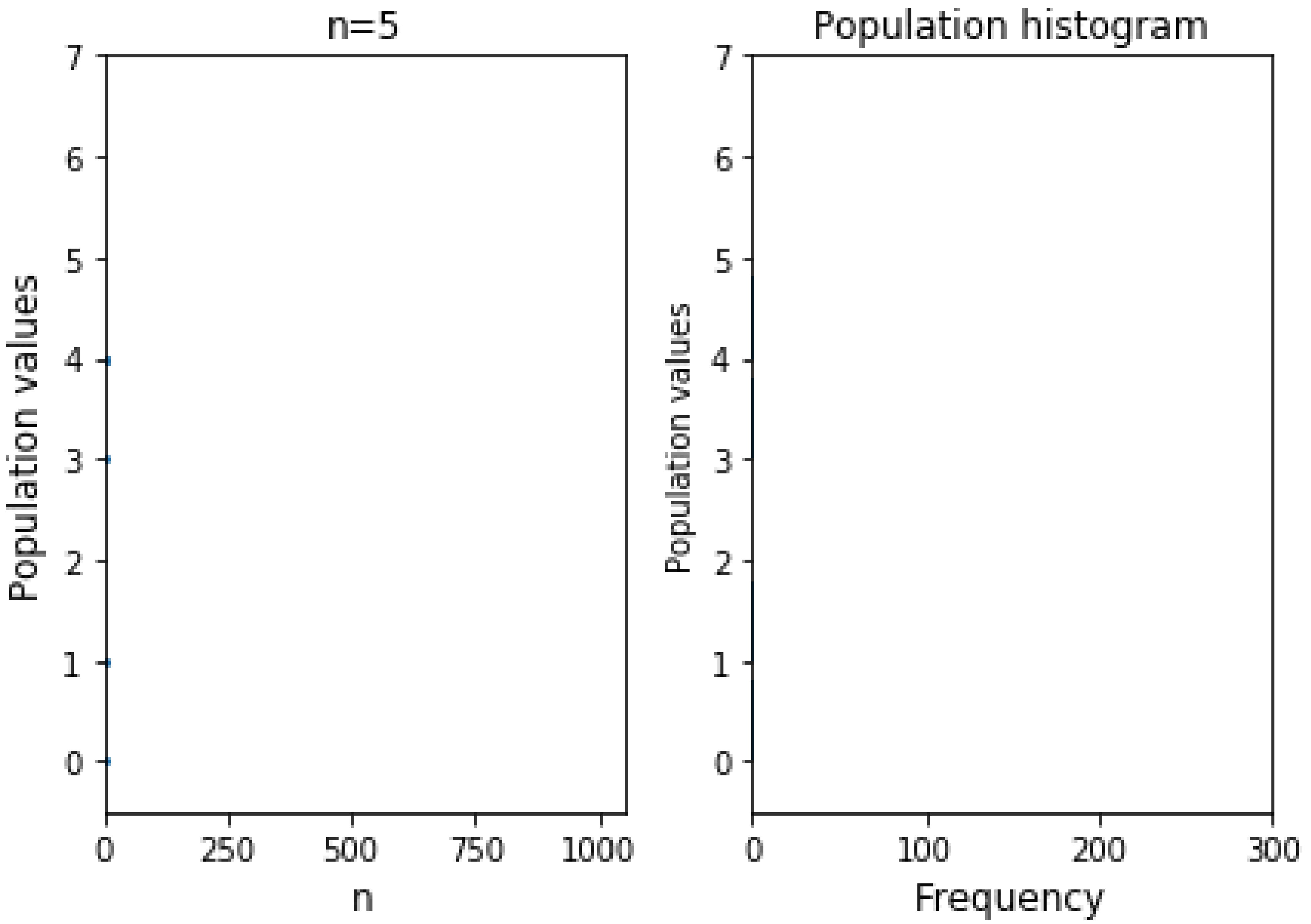
Alexander A. Ramírez M.
CEO @ Synergy Vision

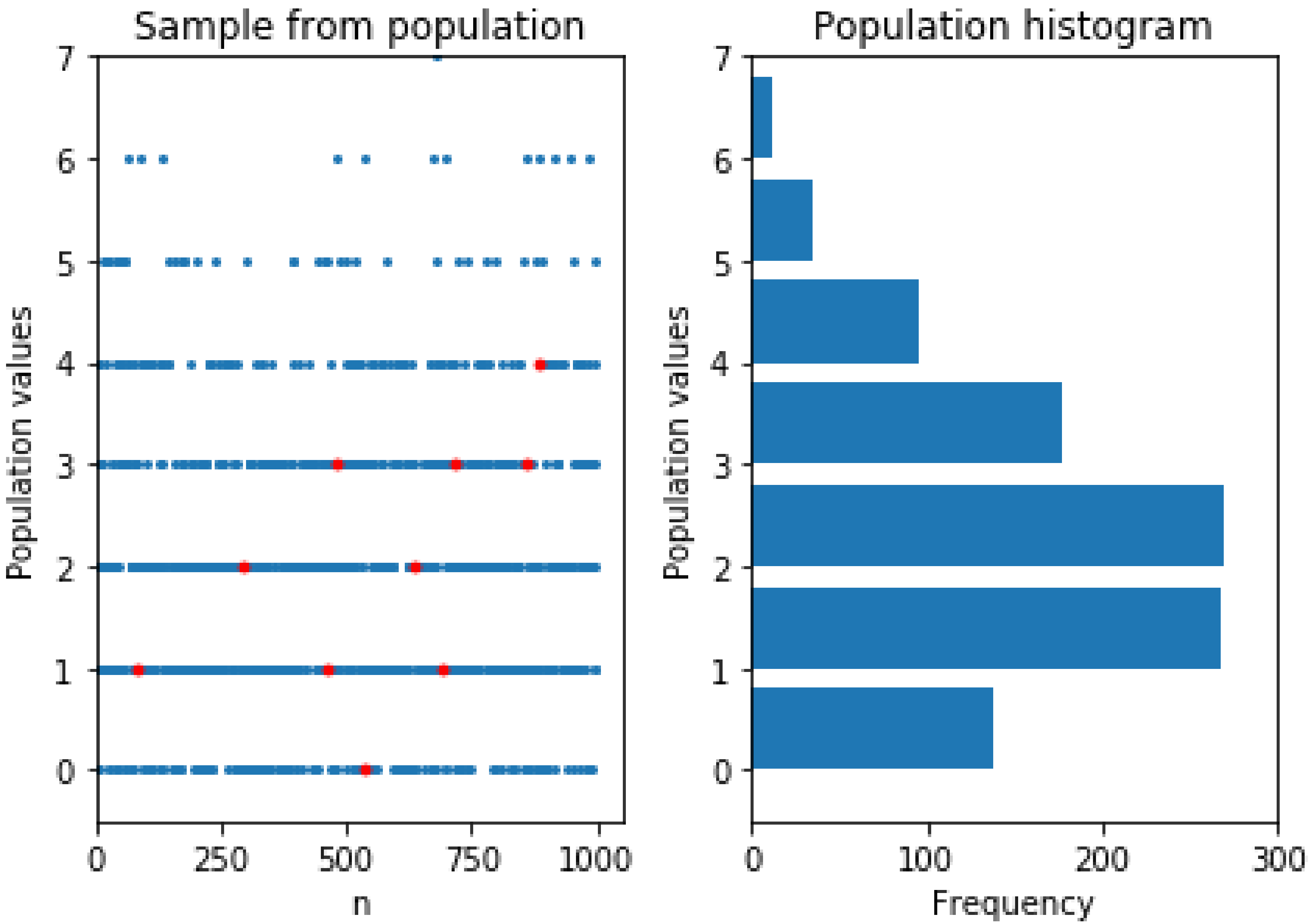
The central limit theorem (CLT)

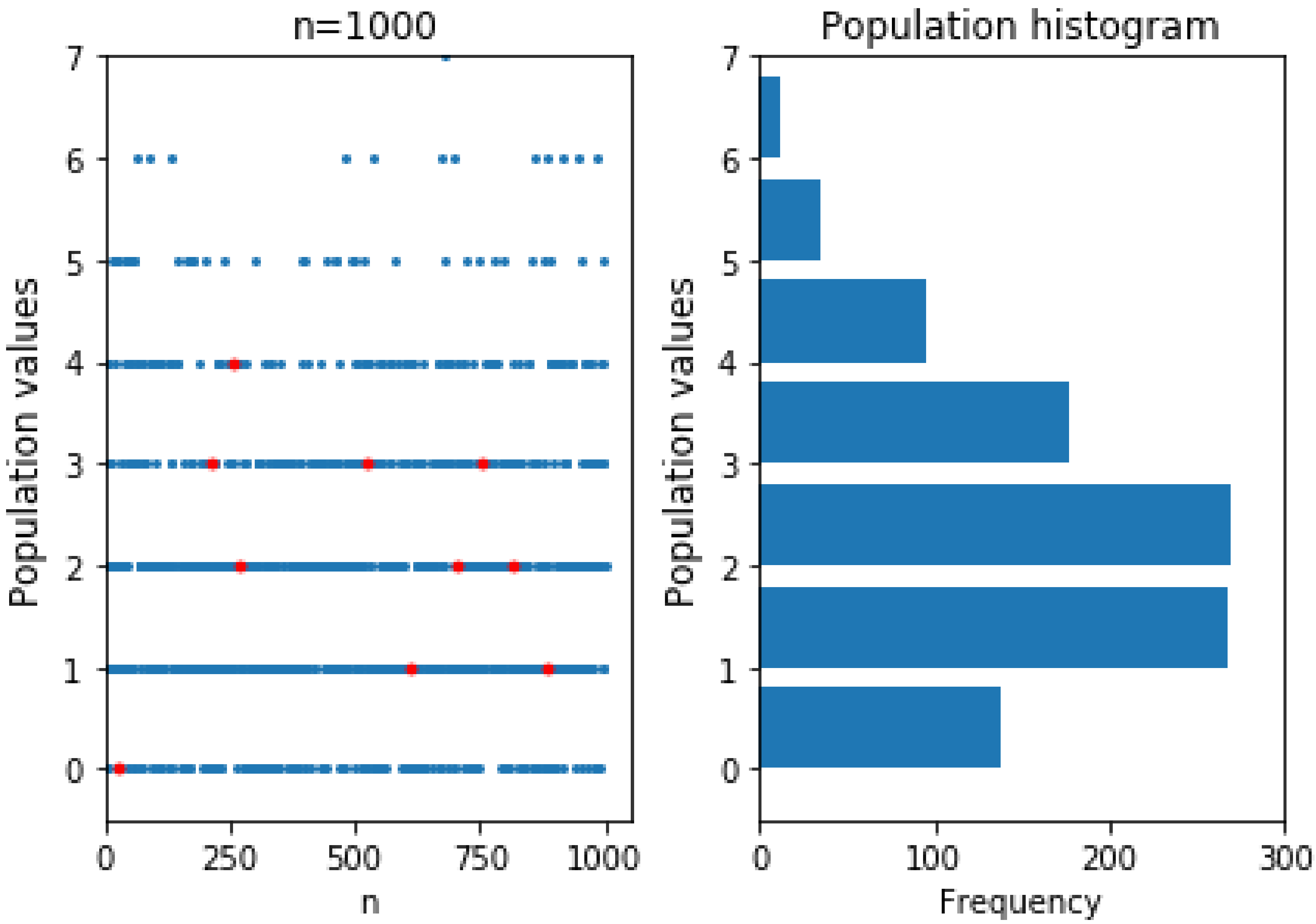
> The sum of random variables tends to a normal distribution as the number of them grows to infinity.

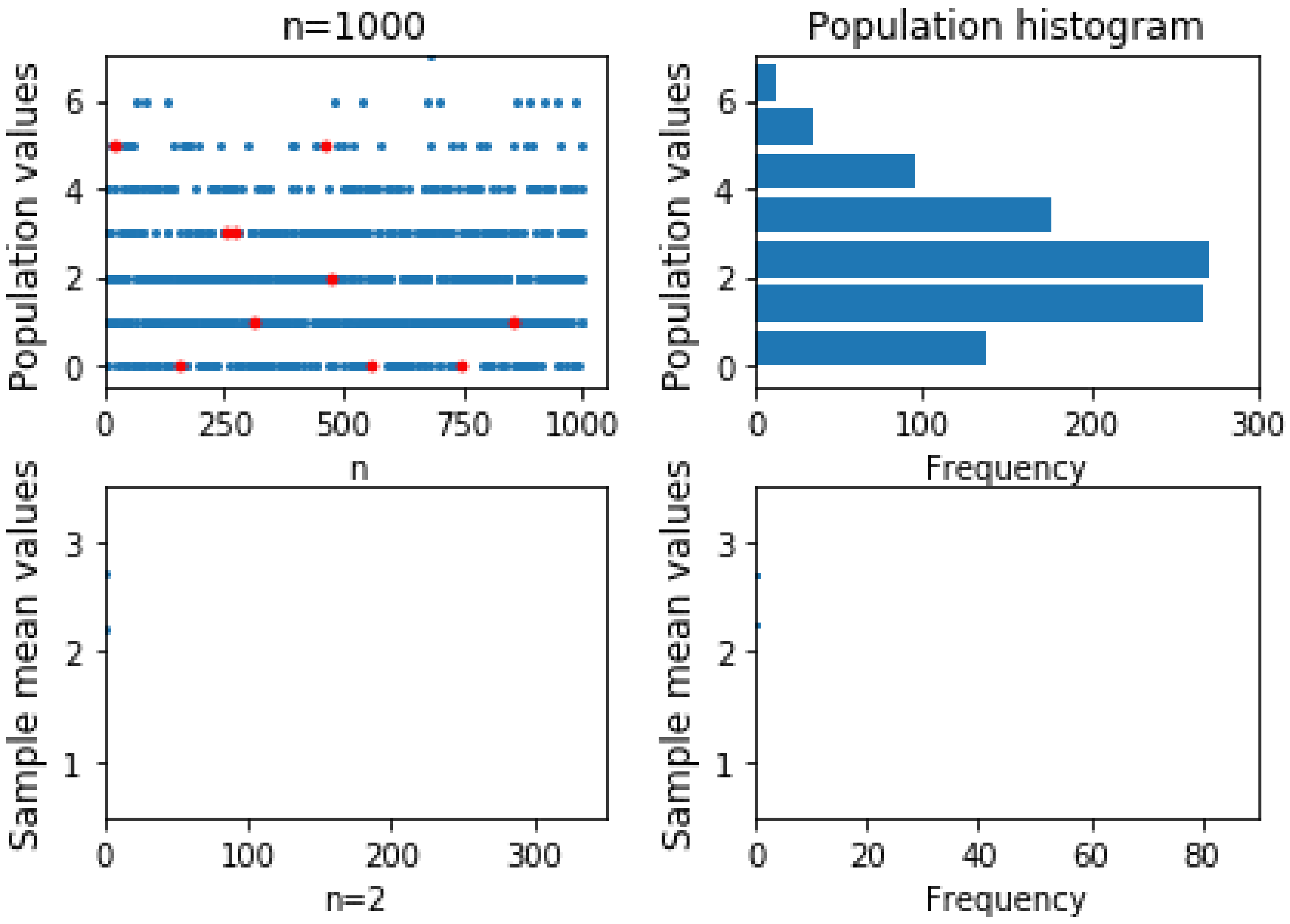
Conditions:

- The variables must have the same distribution.
- The variables must be independent.


n=5
Population values
n
0
250
500
750
1000
Population histogram
Population values
Frequency
0
100
200
300
0
1
2
3
4
5
6
7


Sample from population
Population values
n
Population histogram
Population values
Frequency


n=1000
Population values
7
6
5
4
3
2
1
0
0
250
500
750
1000
n
Population histogram
Population values
0
100
200
300
Frequency



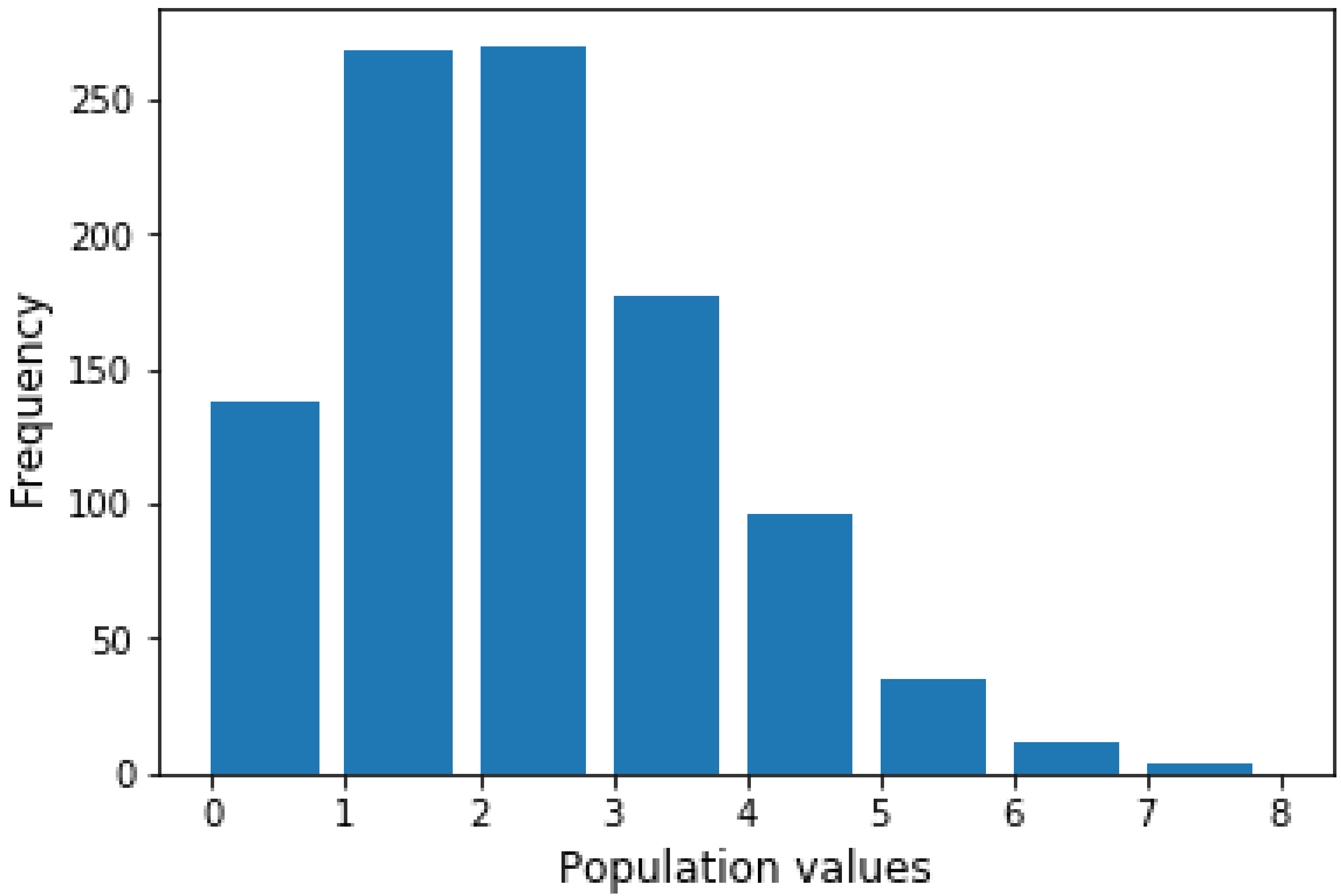
n=1000
Population values
n
Population histogram
Frequency
Sample mean values
n=2

Poisson population plot

```
# Add the imports
from scipy.stats import poisson, describe
from matplotlib import pyplot as plt
import numpy as np
```

```
# Generate the population
population = poisson.rvs(mu=2, size=1000, random_state=20)
```

```
# Draw the histogram with labels
plt.hist(population, bins=range(9), width=0.8)
plt.show()
```


250
200
150
100
50
0
Frequency
0
1
2
3
4
5
6
7
8
Population values

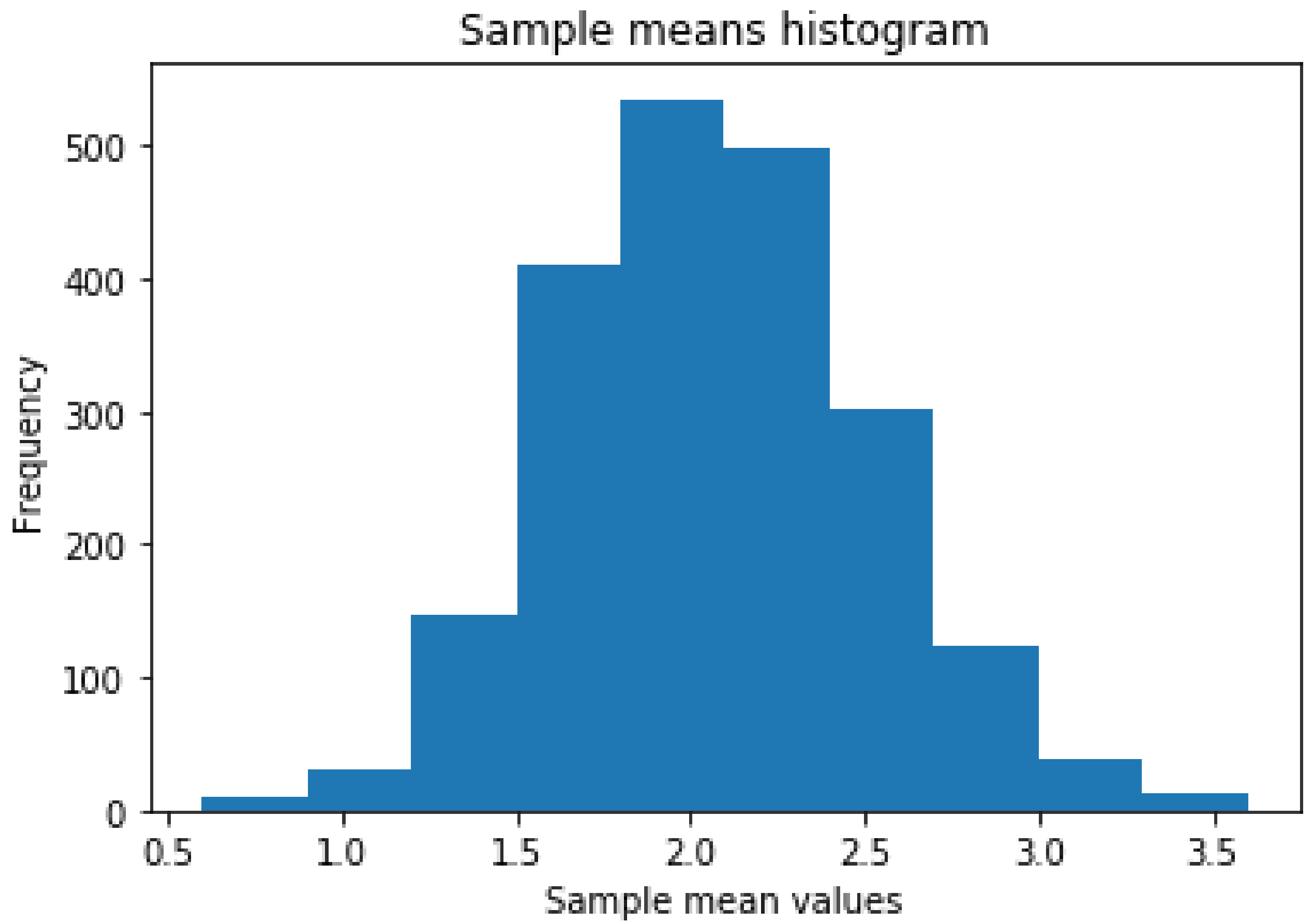
Sample means plot

```
# Generate 350 sample means, selecting
# from population values
np.random.seed(42)

# Define list of sample means
sample_means = []
for _ in range(350):
    # Select 10 from population
    sample = np.random.choice(population, 10)
    # Calculate sample mean of sample
    sample_means.append(describe(sample).mean)
```

Sample means plot (Cont.)

```
# Draw histogram with labels
plt.xlabel("Sample mean values")
plt.ylabel("Frequency")
plt.title("Sample means histogram")
plt.hist(sample_means)
plt.show()
```



Let's add random variables

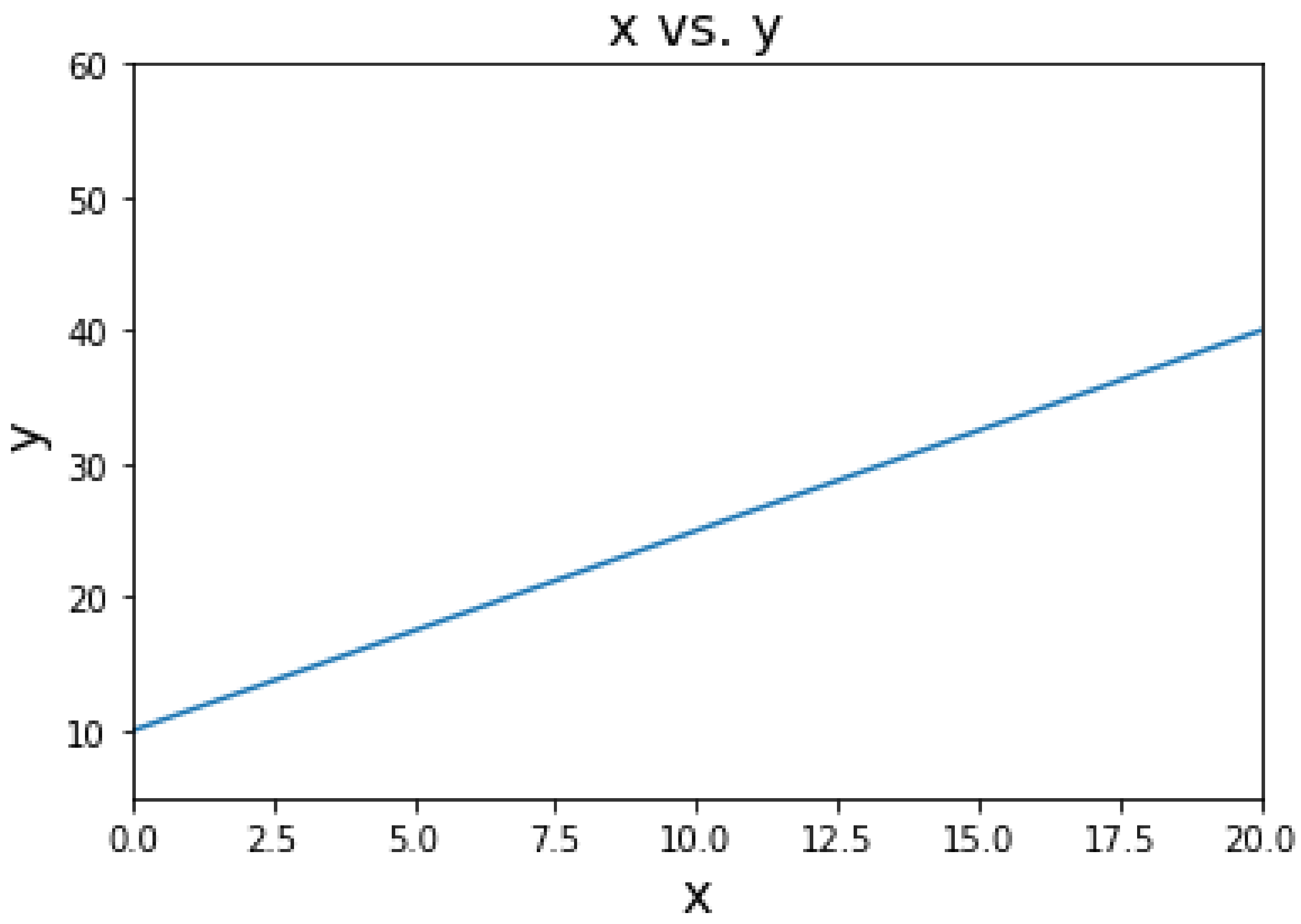
FOUNDATIONS OF PROBABILITY IN PYTHON

Linear regression

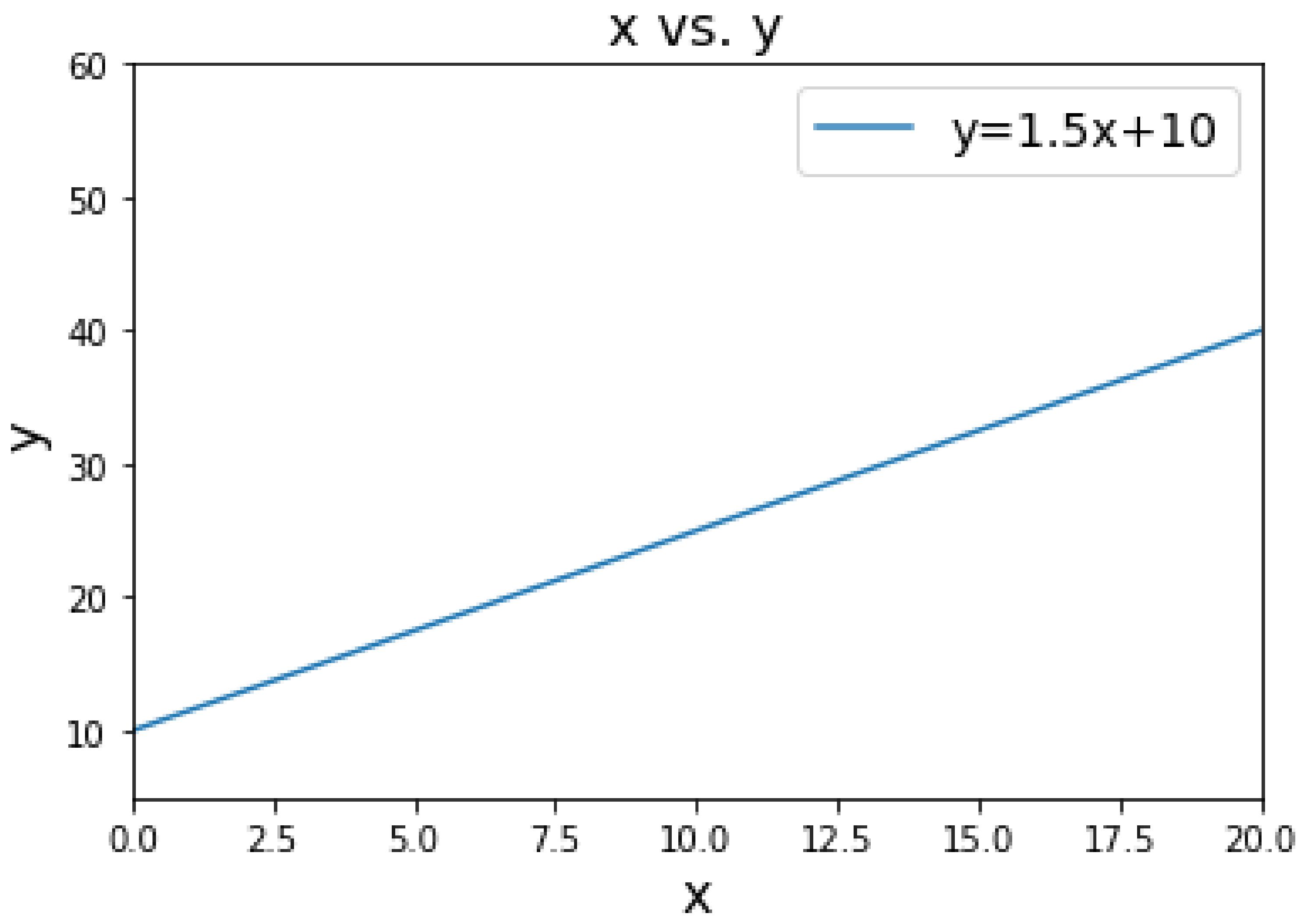
FOUNDATIONS OF PROBABILITY IN PYTHON

Alexander A. Ramírez M.
CEO @ Synergy Vision

Linear functions

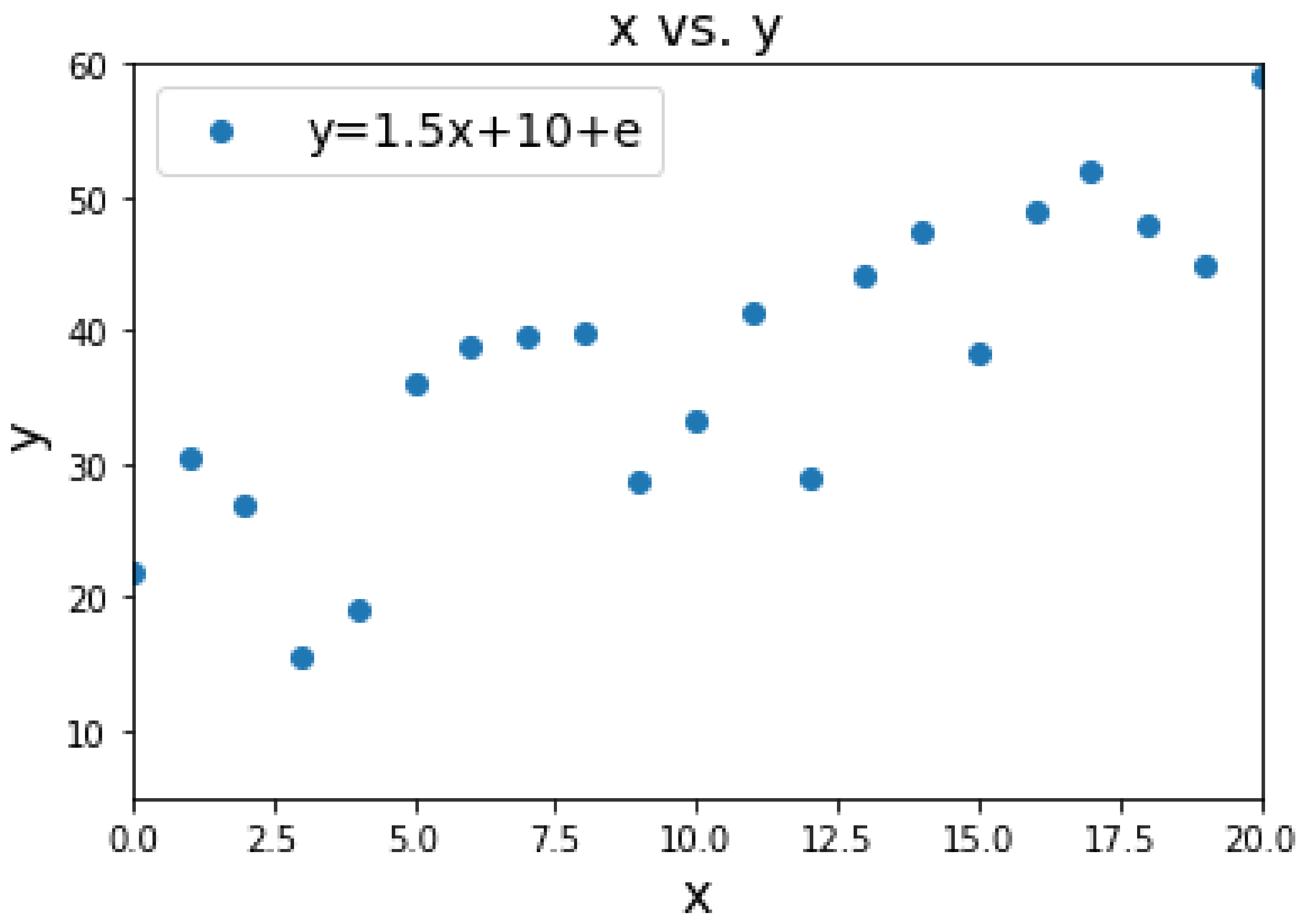


Linear function parameters



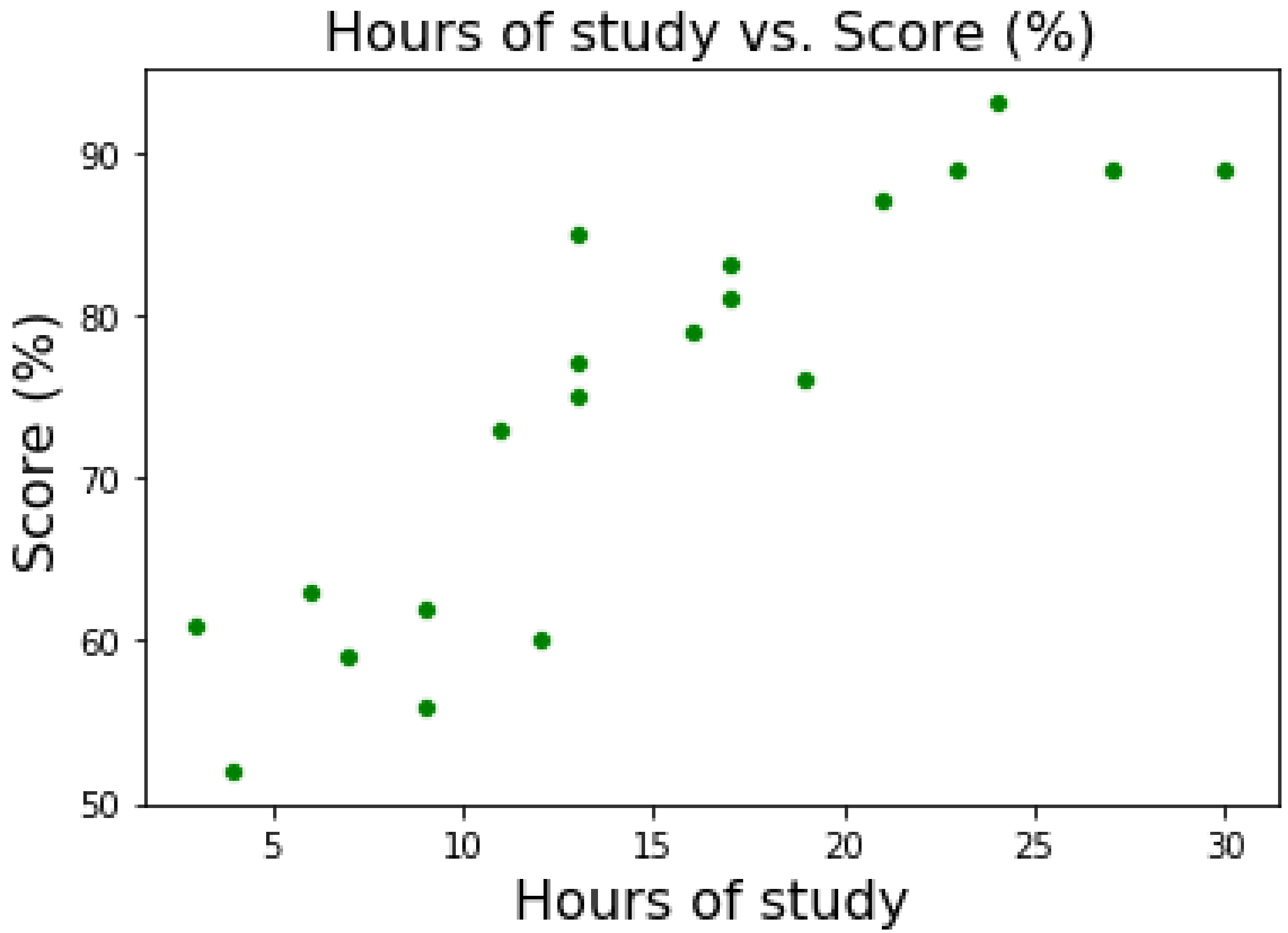
$$y = slope * x + intercept$$

Linear function with random perturbations

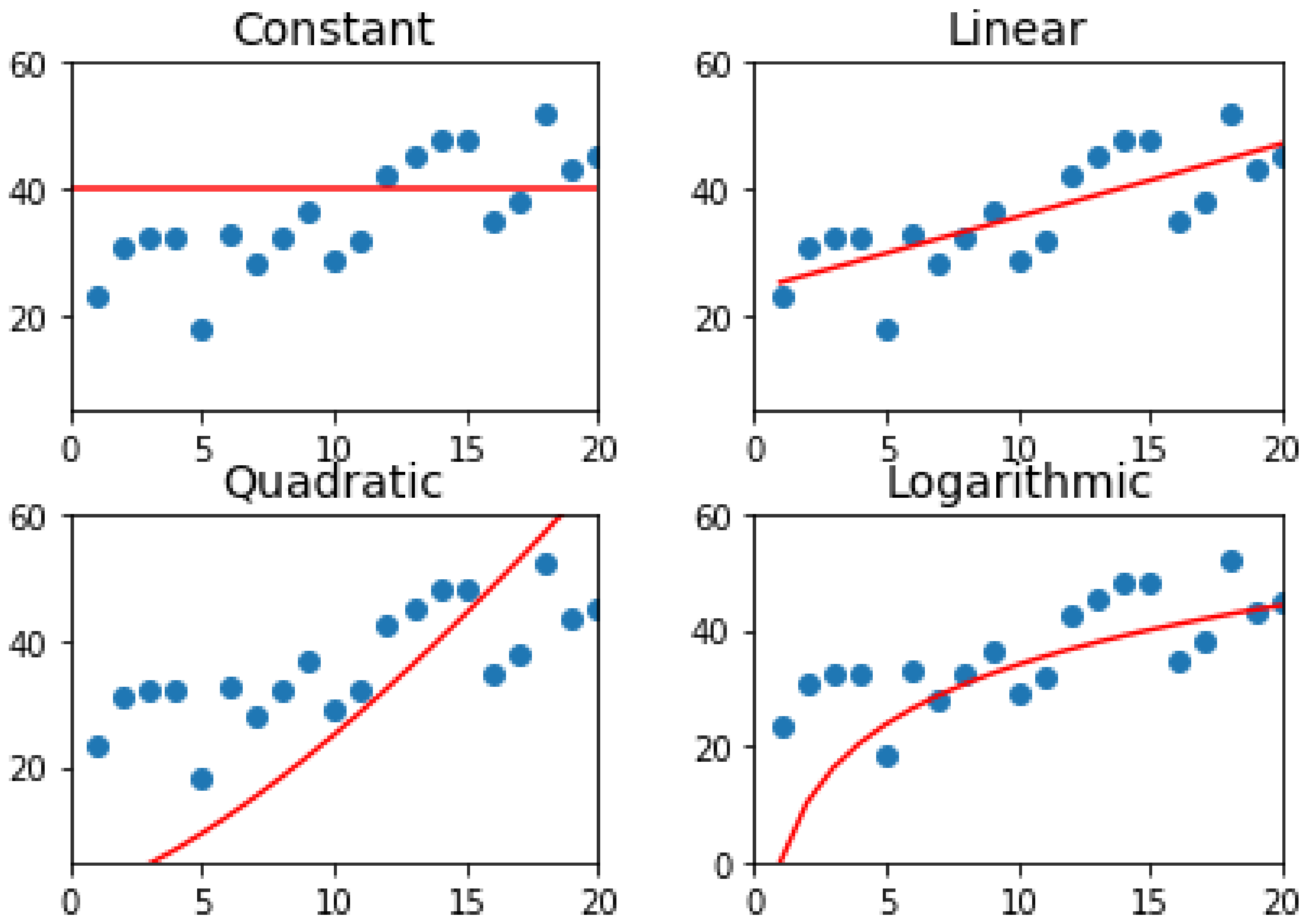


$$y = slope * x + intercept + random\_number$$

Start from the data and find a model that fits

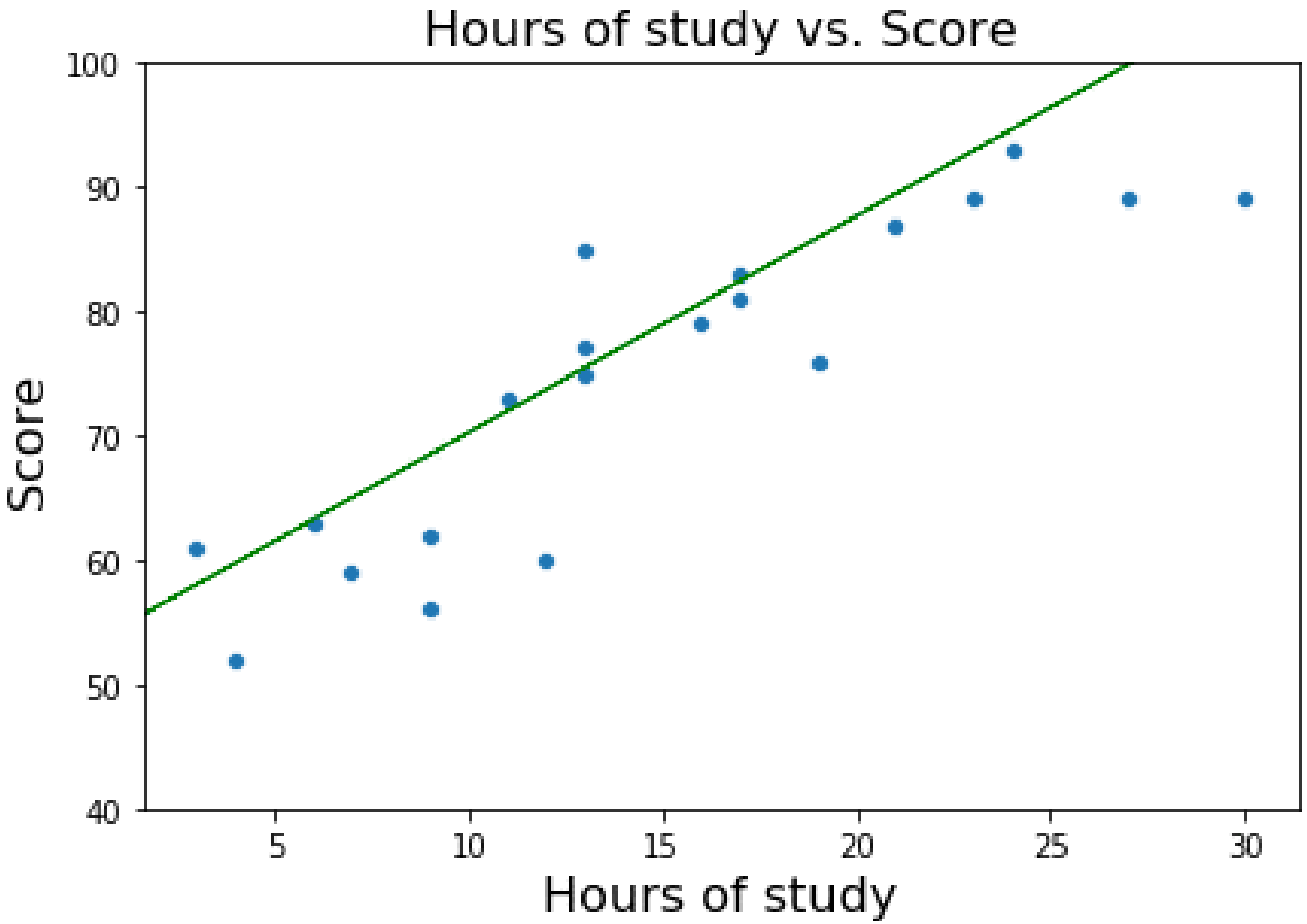


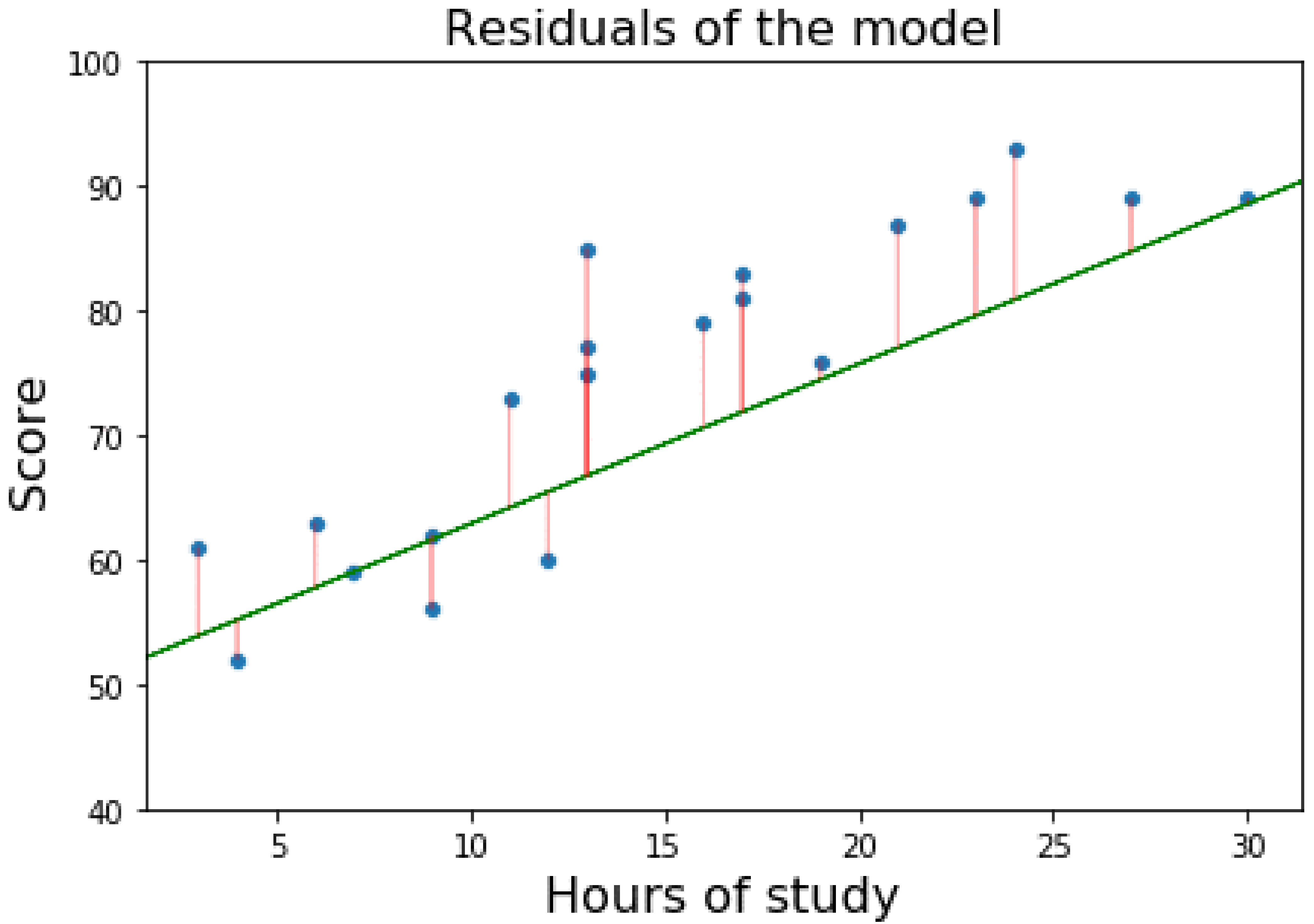
What model will fit the data?

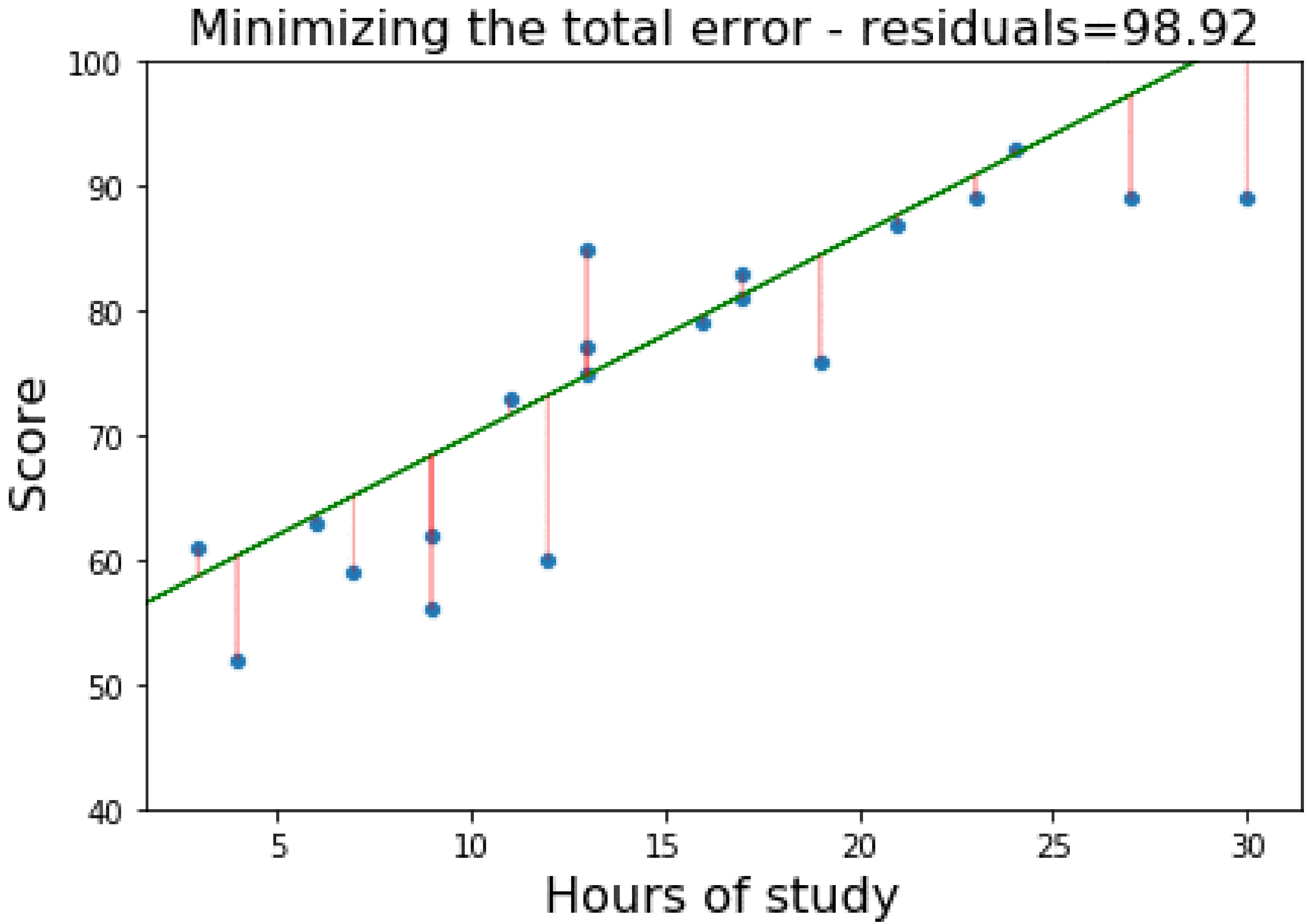


What would be the criteria to determine which is the best model?

What model will fit the data? (Cont.)

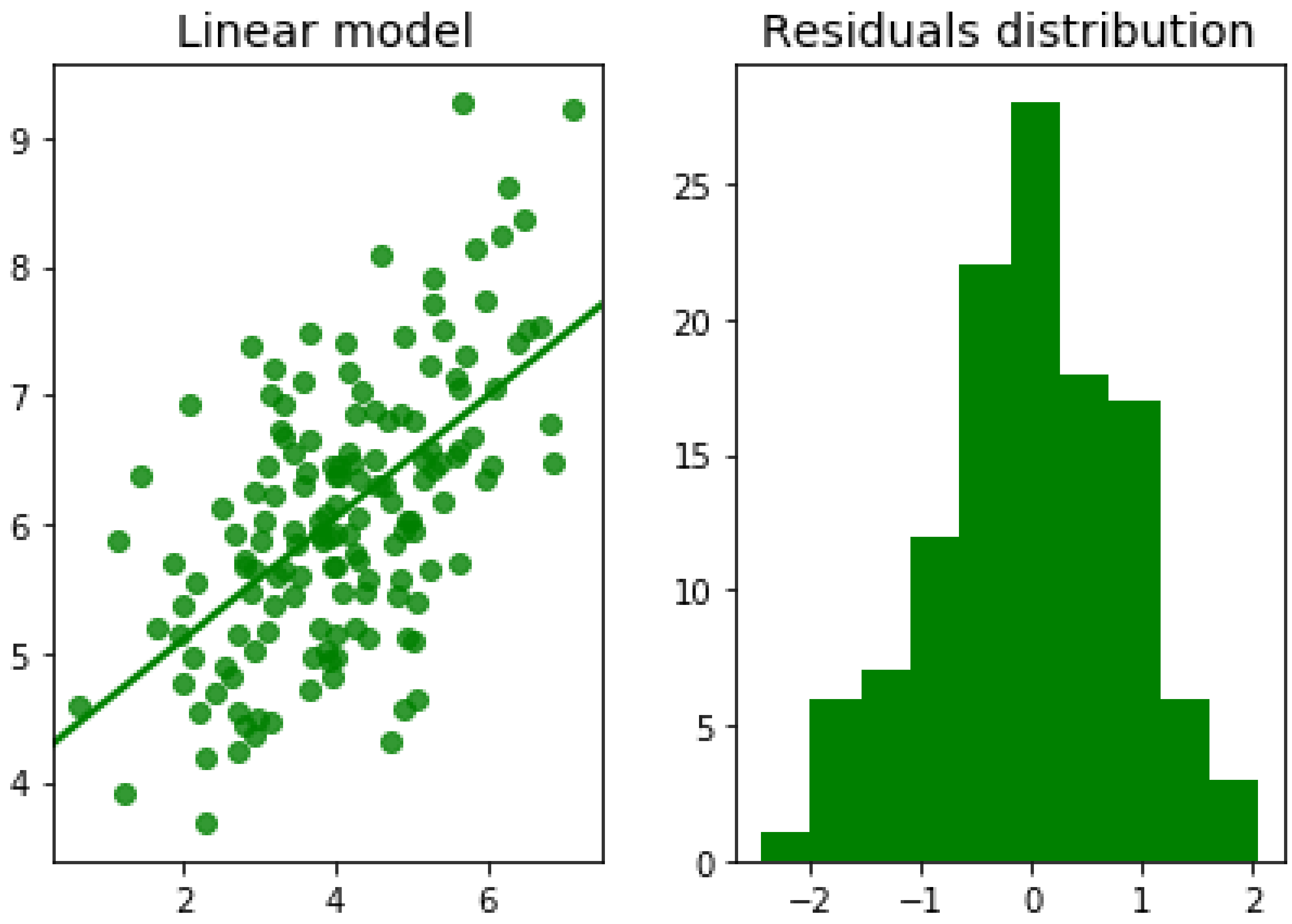



Residuals of the model
Score
Hours of study
100
90
80
70
60
50
40
5
10
15
20
25
30



Minimizing the total error - residuals=98.92
Score
Hours of study
100
90
80
70
60
50
40
5
10
15
20
25
30

Probability and statistics in action



Calculating linear model parameters

```python
# Import LinearRegression
from sklearn.linear_model import LinearRegression
```

```python
# sklearn linear model
model = LinearRegression()
model.fit(hours_of_study, scores)
```

```python
# Get parameters
slope = model.coef_[0]
intercept = model.intercept_
```

```python
# Print parameters
print(slope, intercept)
```

```
(1.496703900384545, 52.44845266434719)
```

Predicting scores based on hours of study

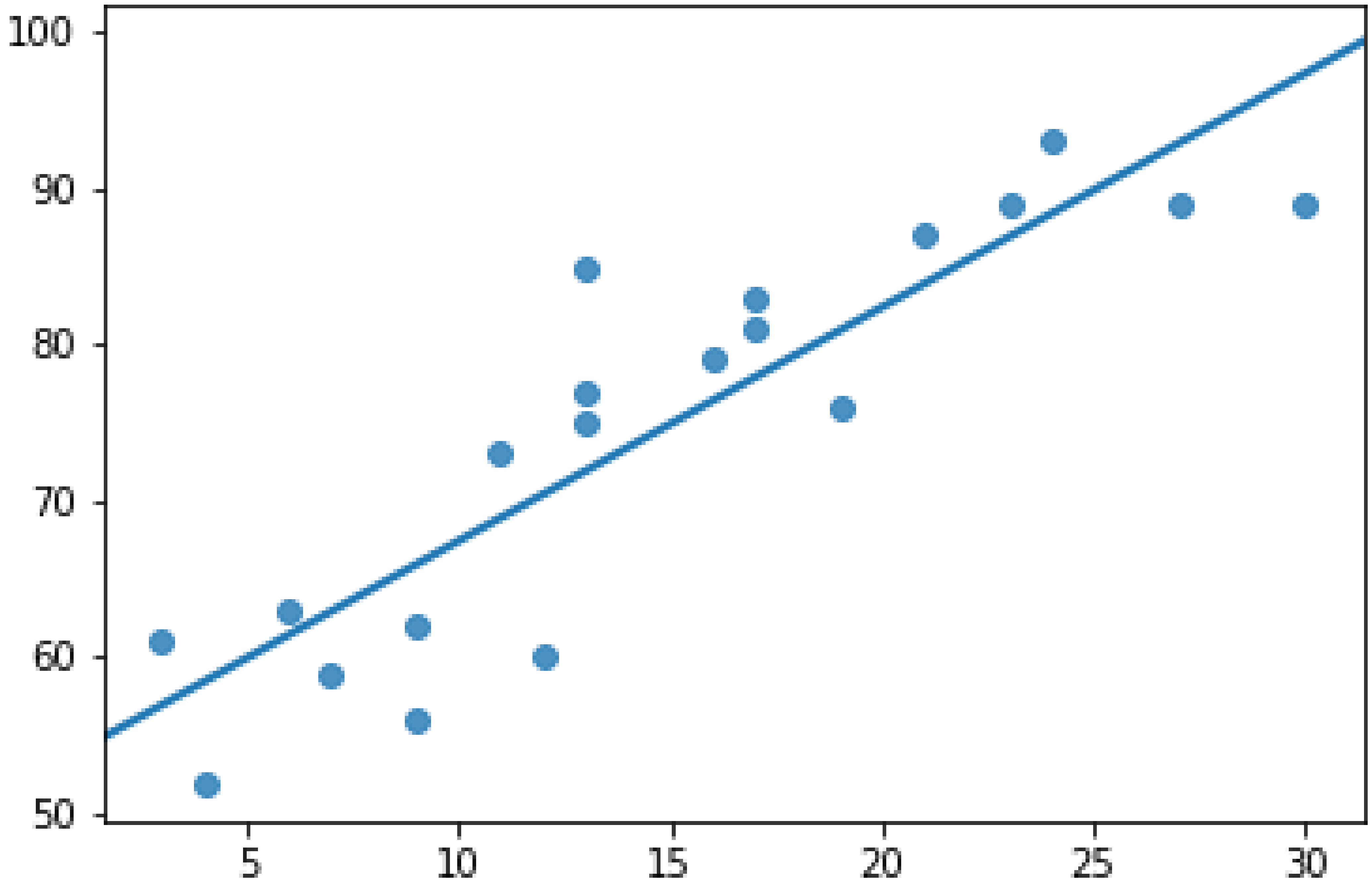
```python
# Score prediction
score = model.predict(np.array([[15]]))
print(score)
```

```
[74.89901117]
```

Plotting the linear model

```
import matplotlib.pyplot as plt
```

```
plt.scatter(hours_of_study, scores)
plt.plot(hours_of_study_values, model.predict(hours_of_study_values))
plt.show()
```


100
90
80
70
60
50
5
10
15
20
25
30

Let's practice with linear models

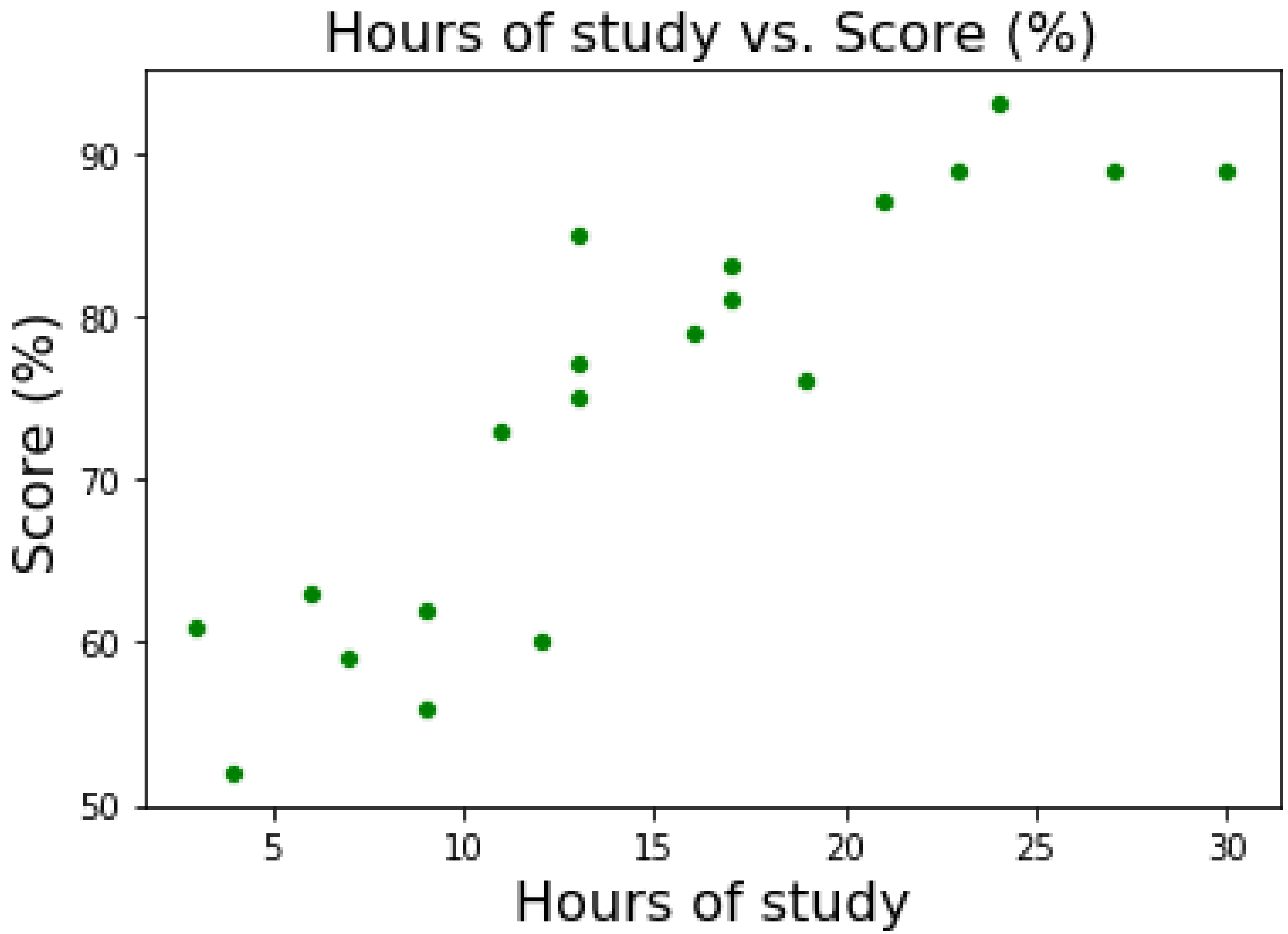
FOUNDATIONS OF PROBABILITY IN PYTHON

Logistic regression

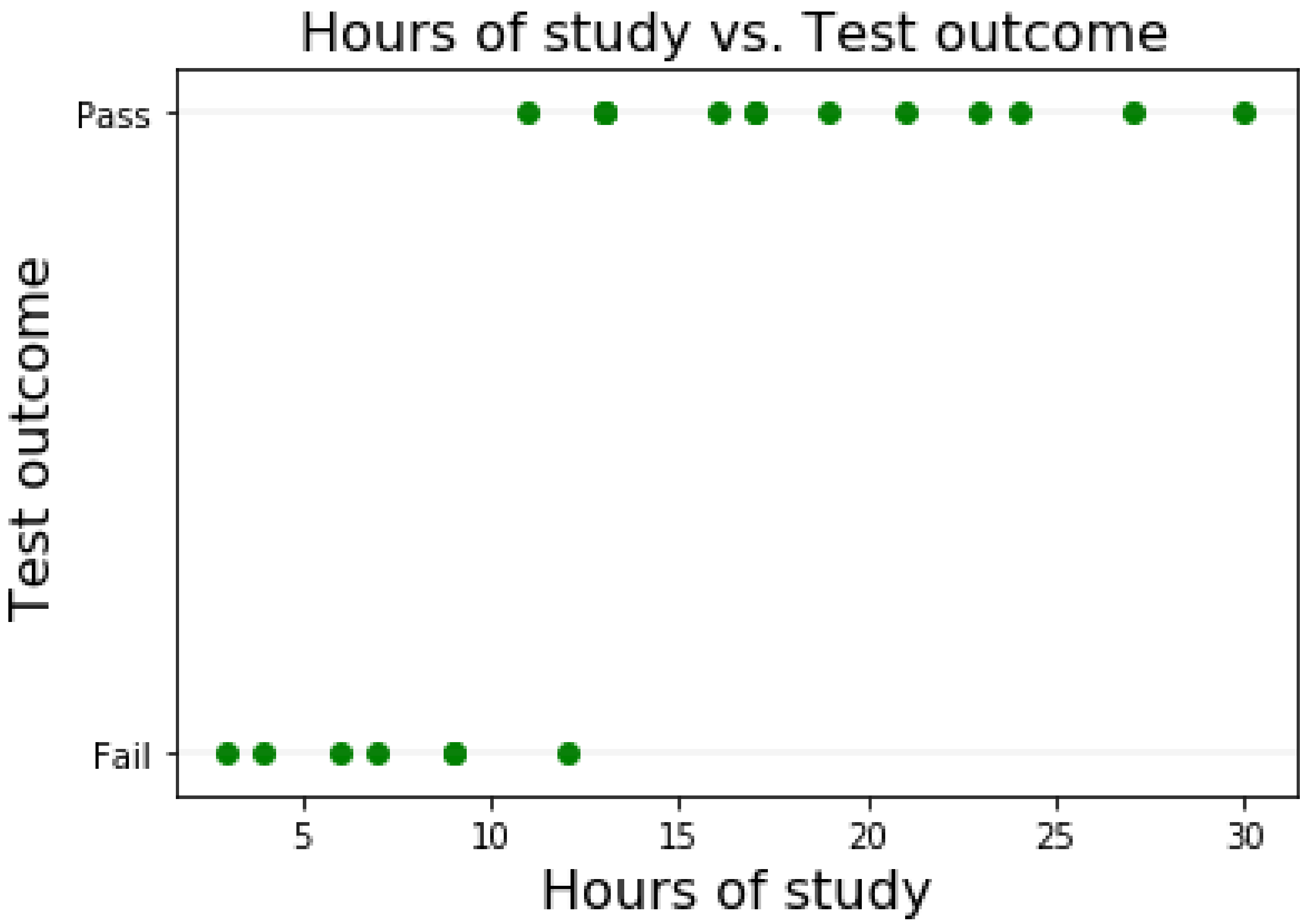
FOUNDATIONS OF PROBABILITY IN PYTHON

Alexander A. Ramírez M.
CEO @ Synergy Vision

Original data

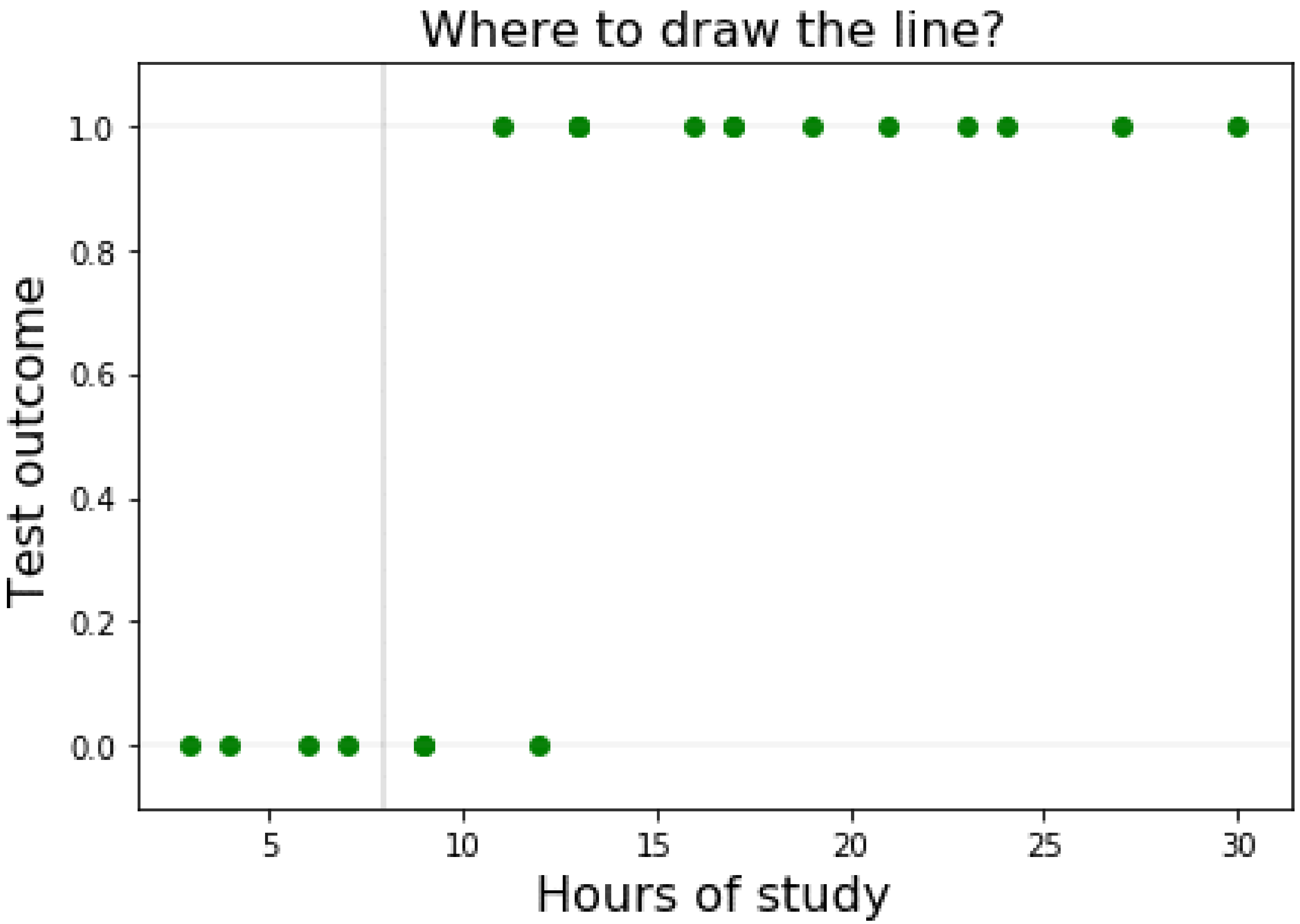


New data

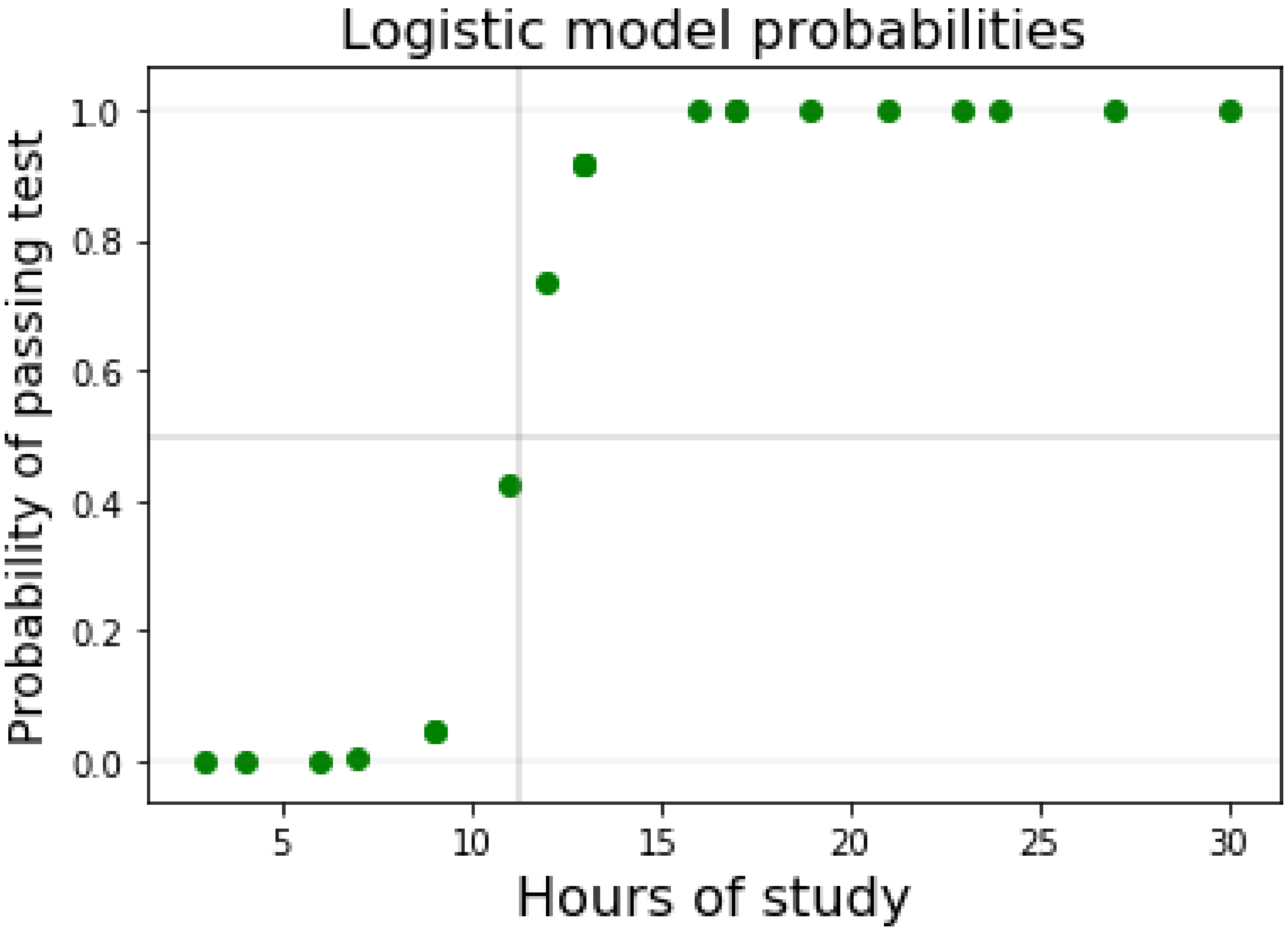


Hours of study vs. Test outcome
Pass
Fail
Test outcome
5
10
15
20
25
30
Hours of study

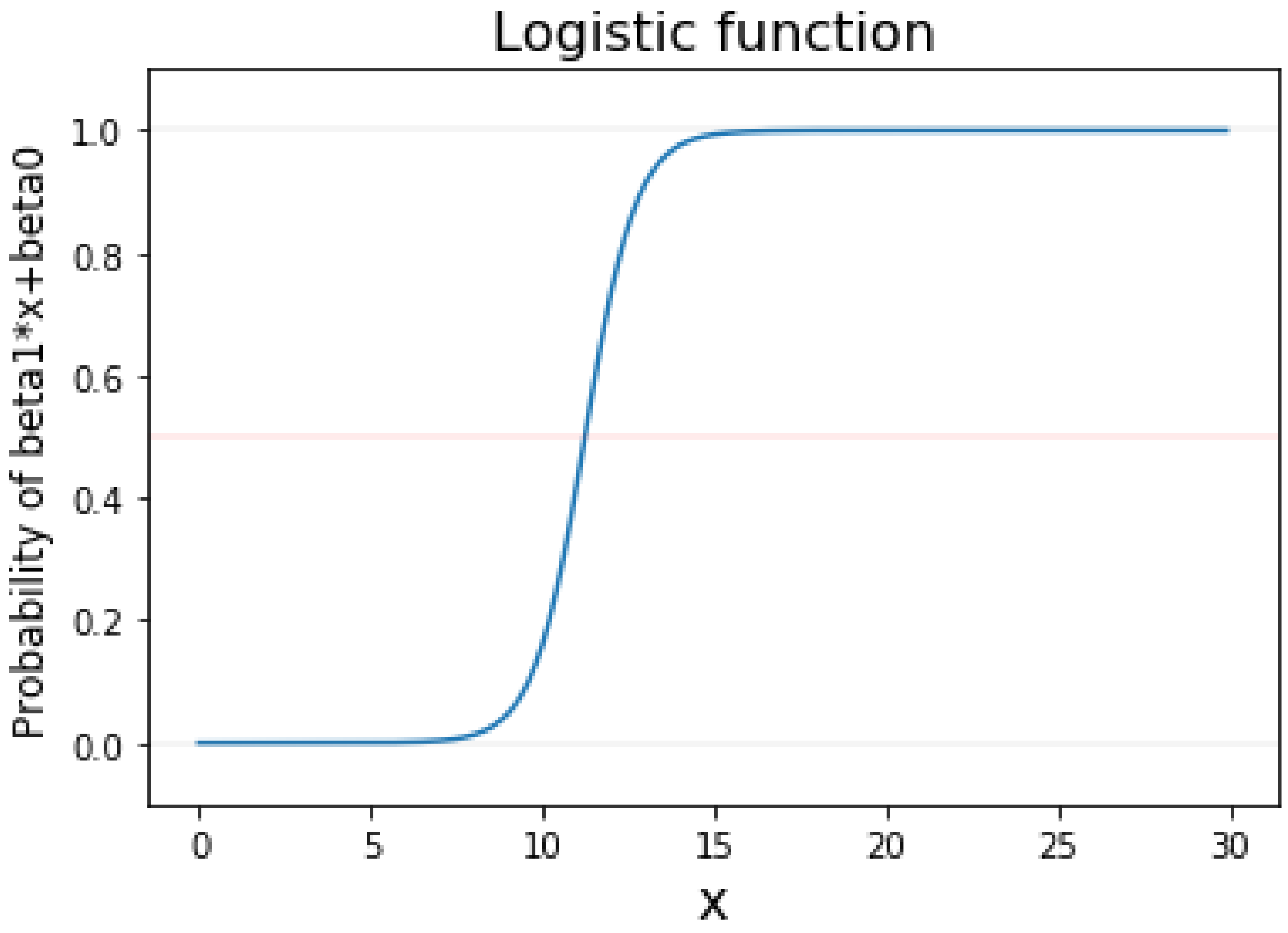
Where would you draw the line?



Solution based on probability

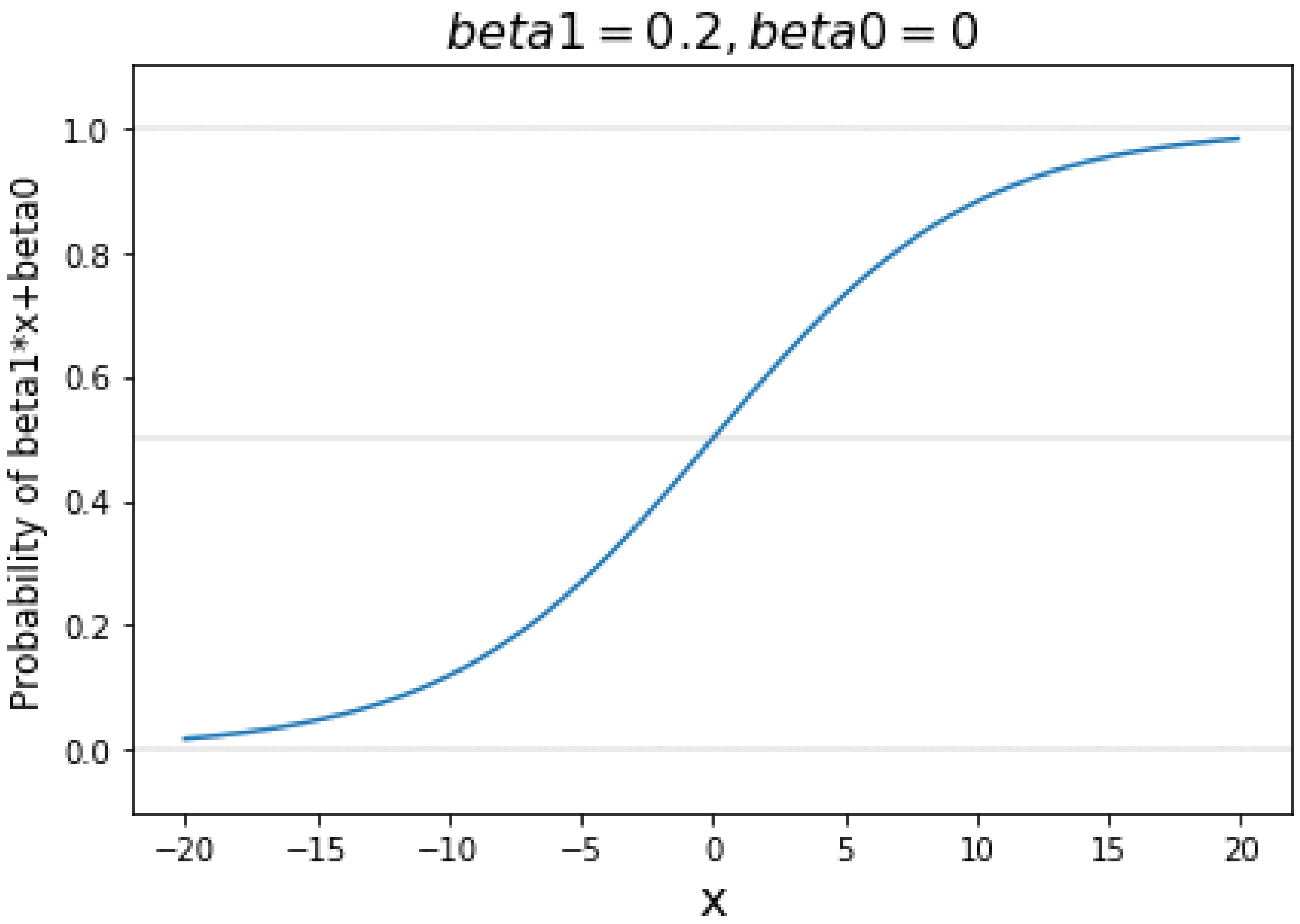


The logistic function

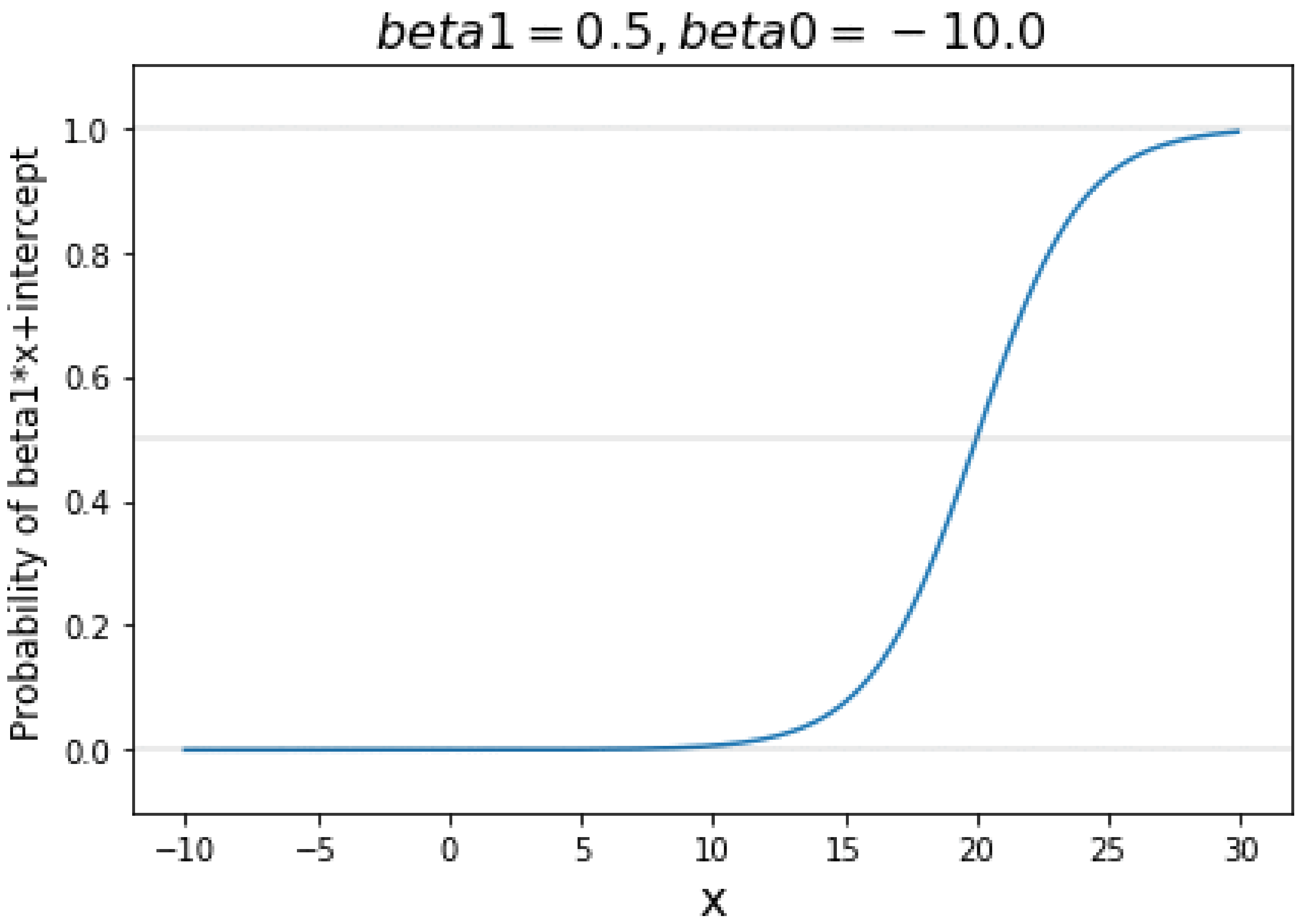


$$logistic(t) = logistic(slope * x + intercept)$$

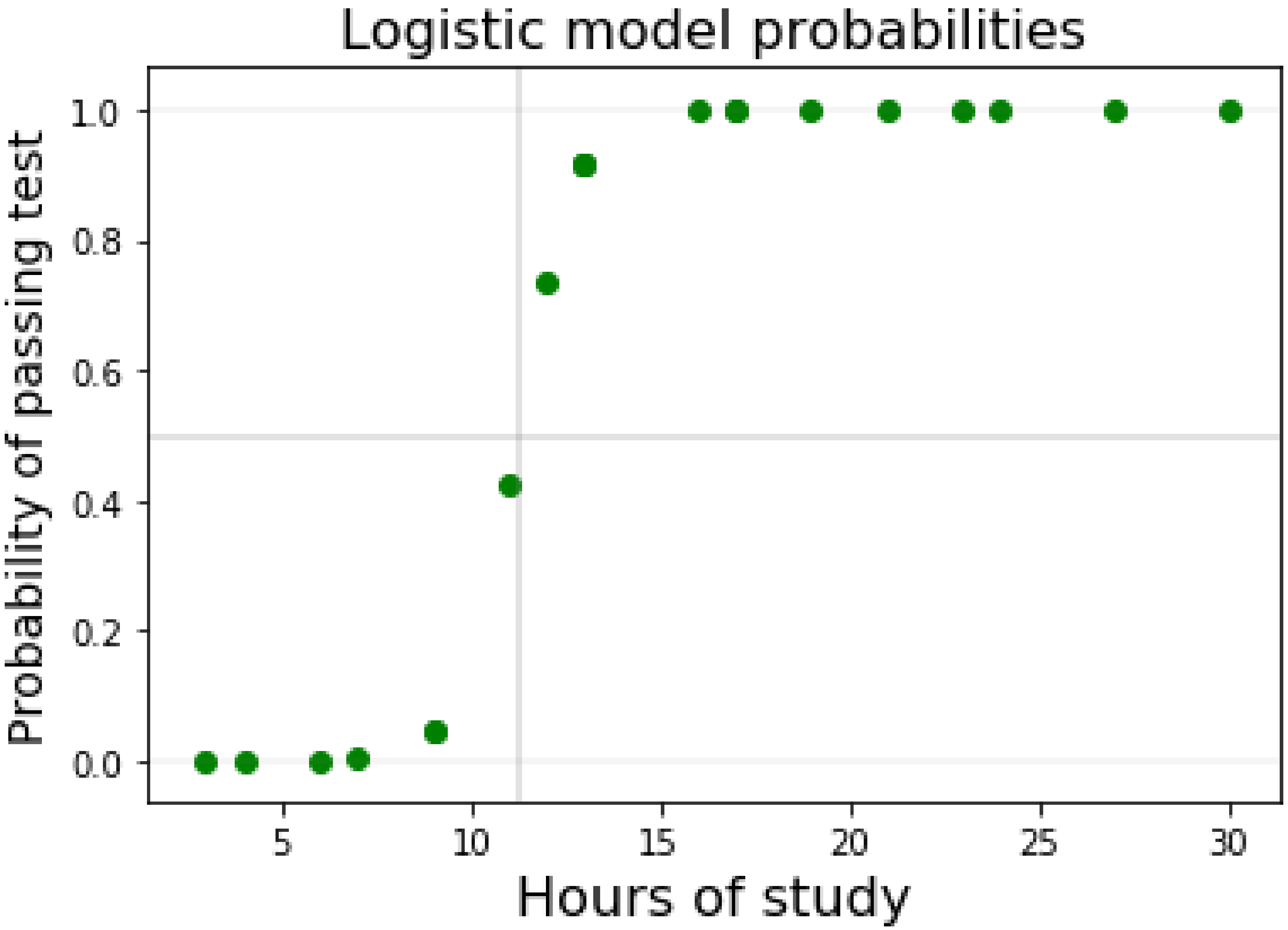
Changing the slope



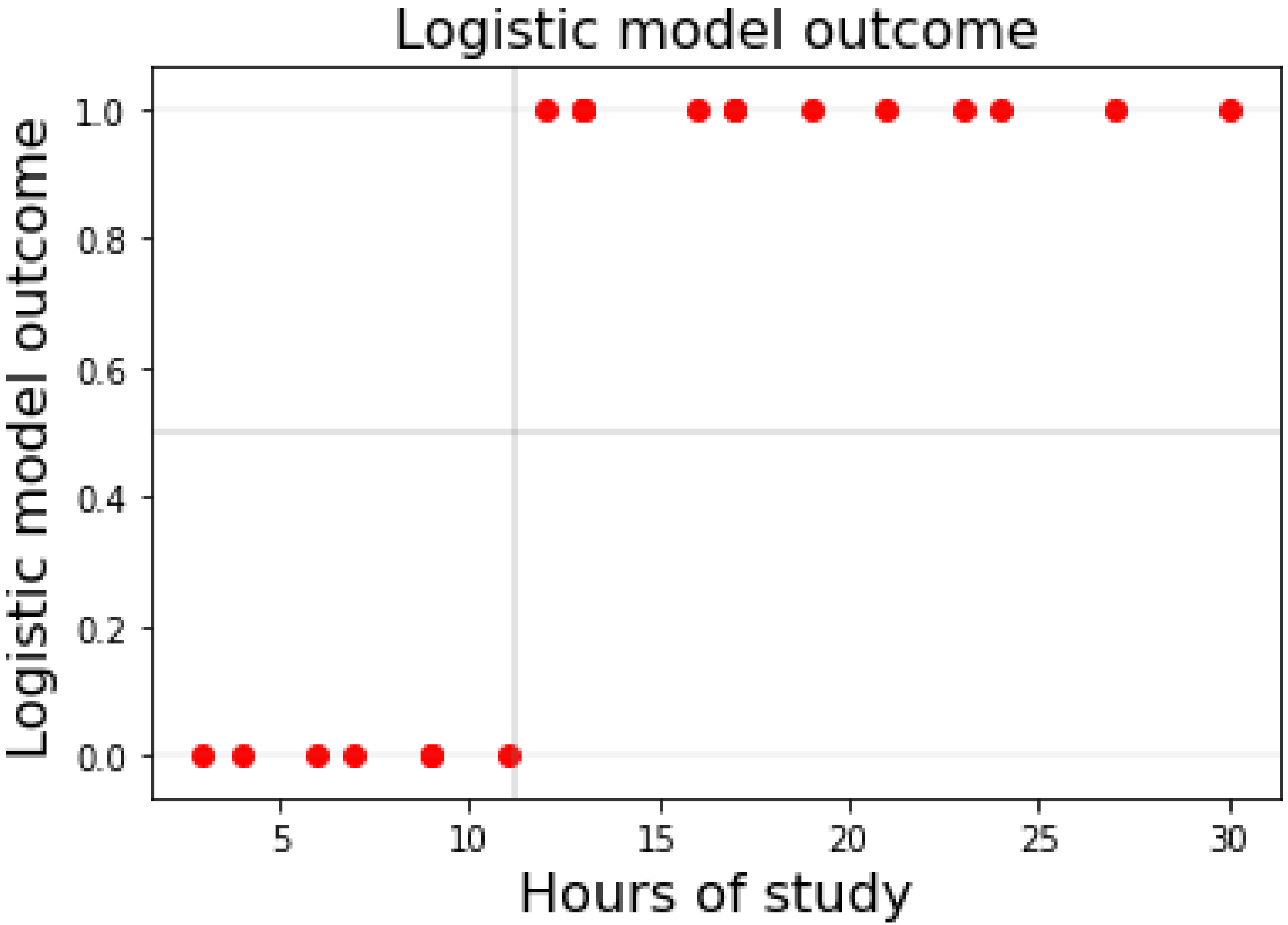
Changing the intercept



From data to probability

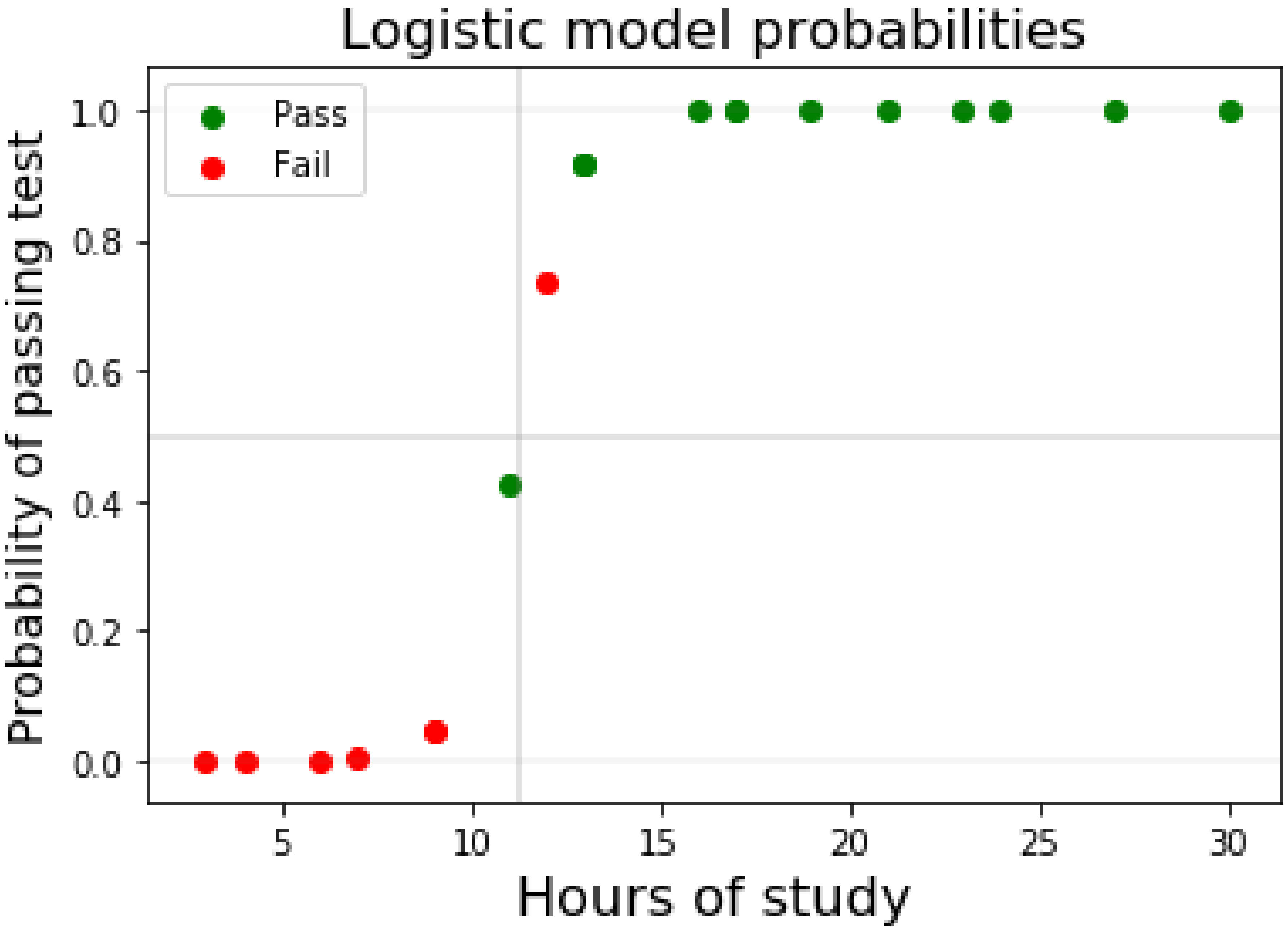


Outcomes



Logistic model outcome
Logistic model outcome
1.0
0.8
0.6
0.4
0.2
0.0
5
10
15
20
25
30
Hours of study

Misclassifications



Logistic regression

```
# Import LogisticRegression
from sklearn.linear_model import LogisticRegression
```

```
# sklearn logistic model
model = LogisticRegression(C=1e9)
model.fit(hours_of_study, outcomes)
```

```
# Get parameters
beta1 = model.coef_[0][0]
beta0 = model.intercept_[0]
```

```
# Print parameters
print(beta1, beta0)
```

```
(1.3406531235010786, -15.05906237996095)
```

Predicting outcomes based on hours of study

```
hours_of_study_test = [[10]]
```

```
outcome = model.predict(hours_of_study_test)
print(outcome)
```

```
array([False])
```

Probability calculation

```
# Put value in an array
value = np.asarray(9).reshape(-1,1)
# Calculate the probability for 9 hours of study
print(model.predict_proba(value)[:,1])
```

```
array([0.04773474])
```

Let's practice!

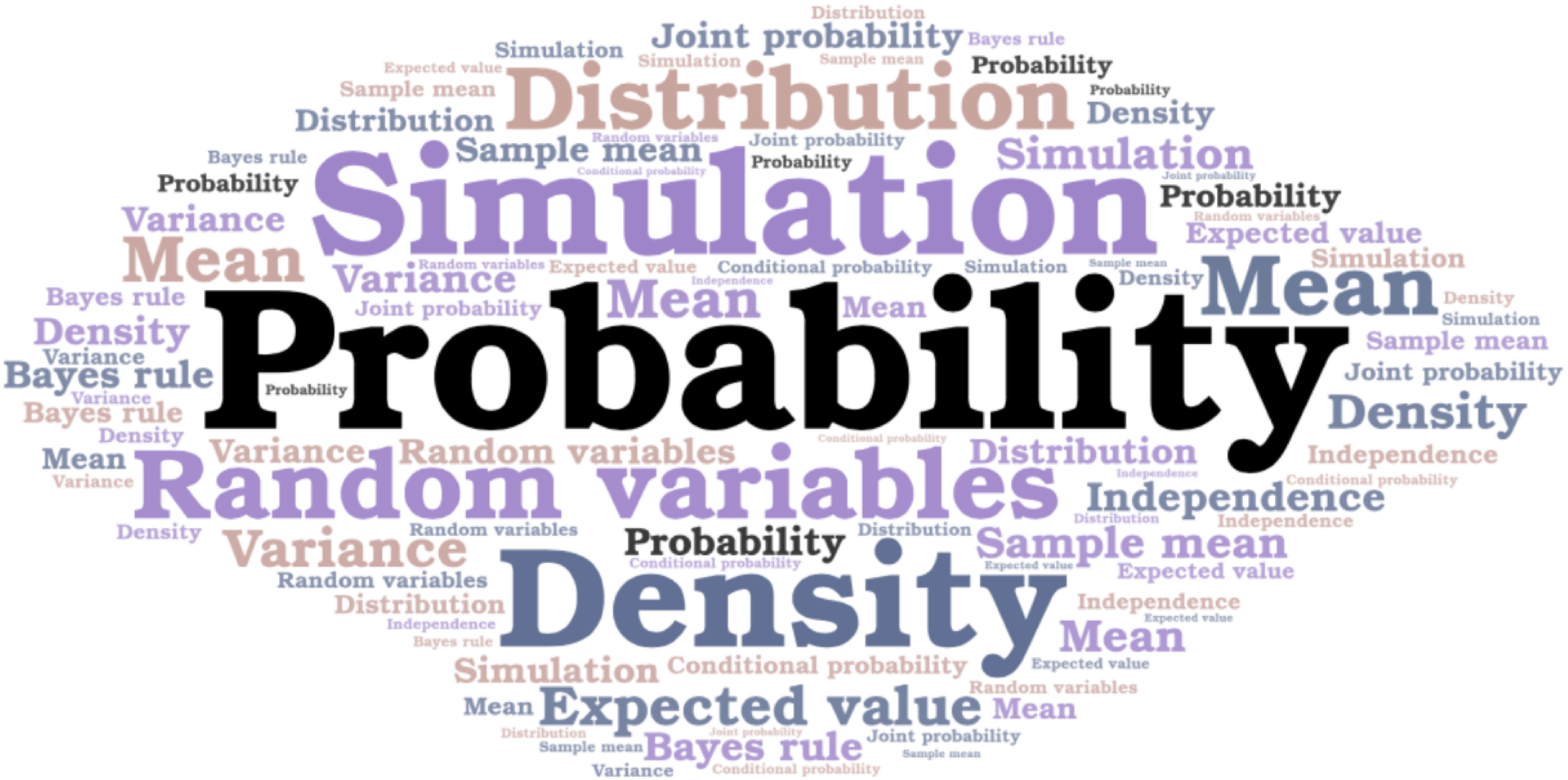
FOUNDATIONS OF PROBABILITY IN PYTHON

Wrapping up

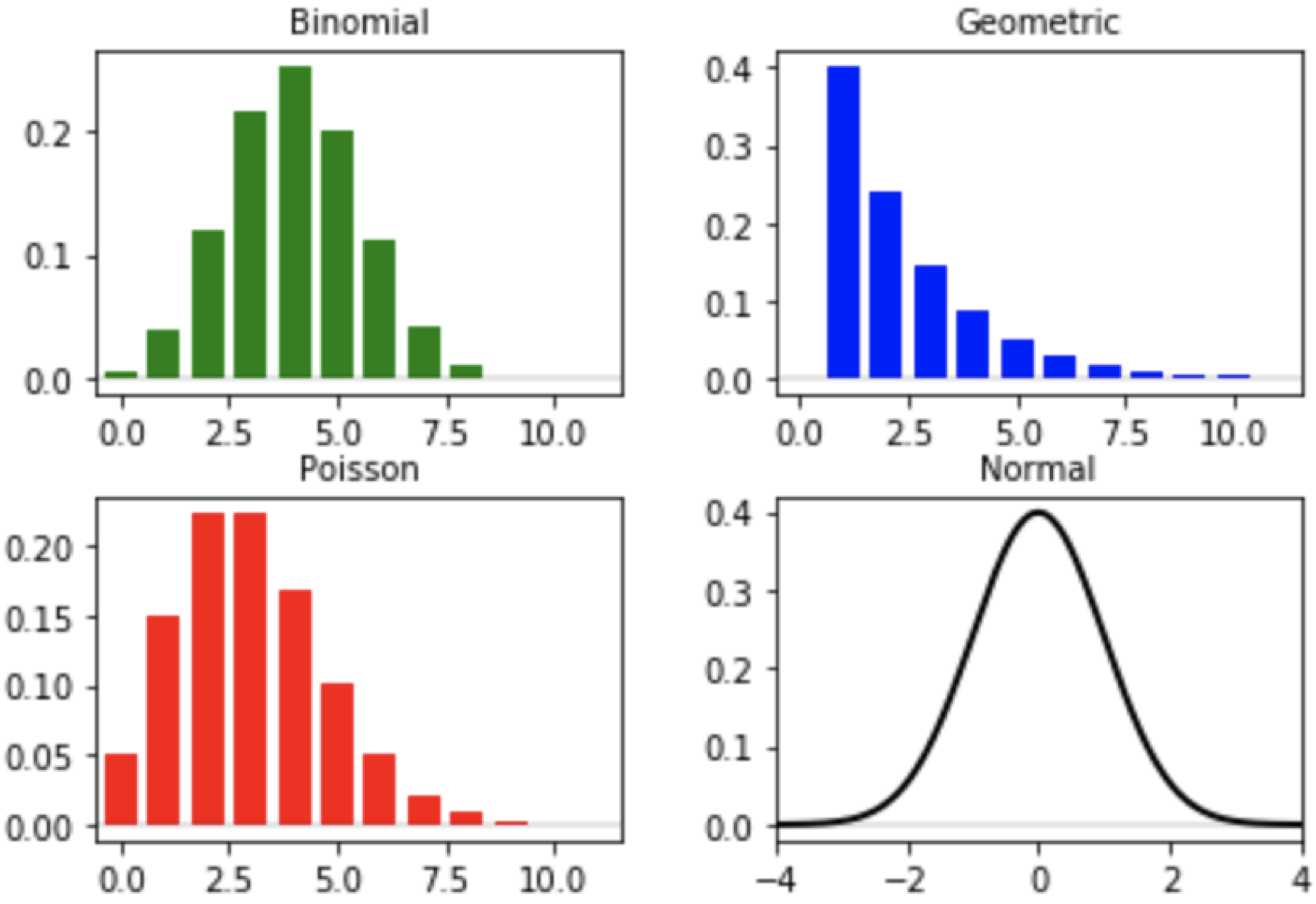
FOUNDATIONS OF PROBABILITY IN PYTHON

Alexander A. Ramírez M.
CEO @ Synergy Vision

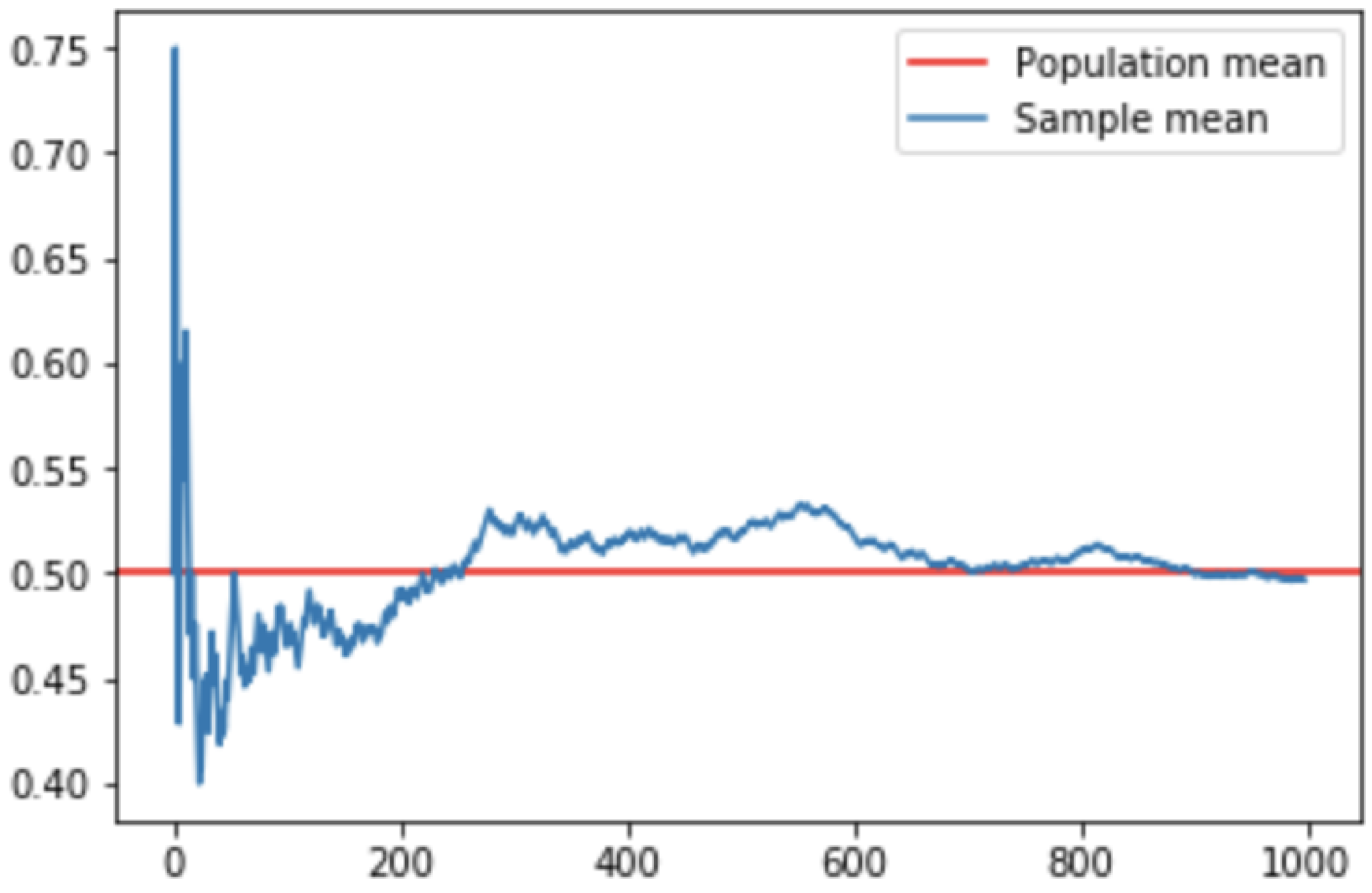
Fundamental concepts

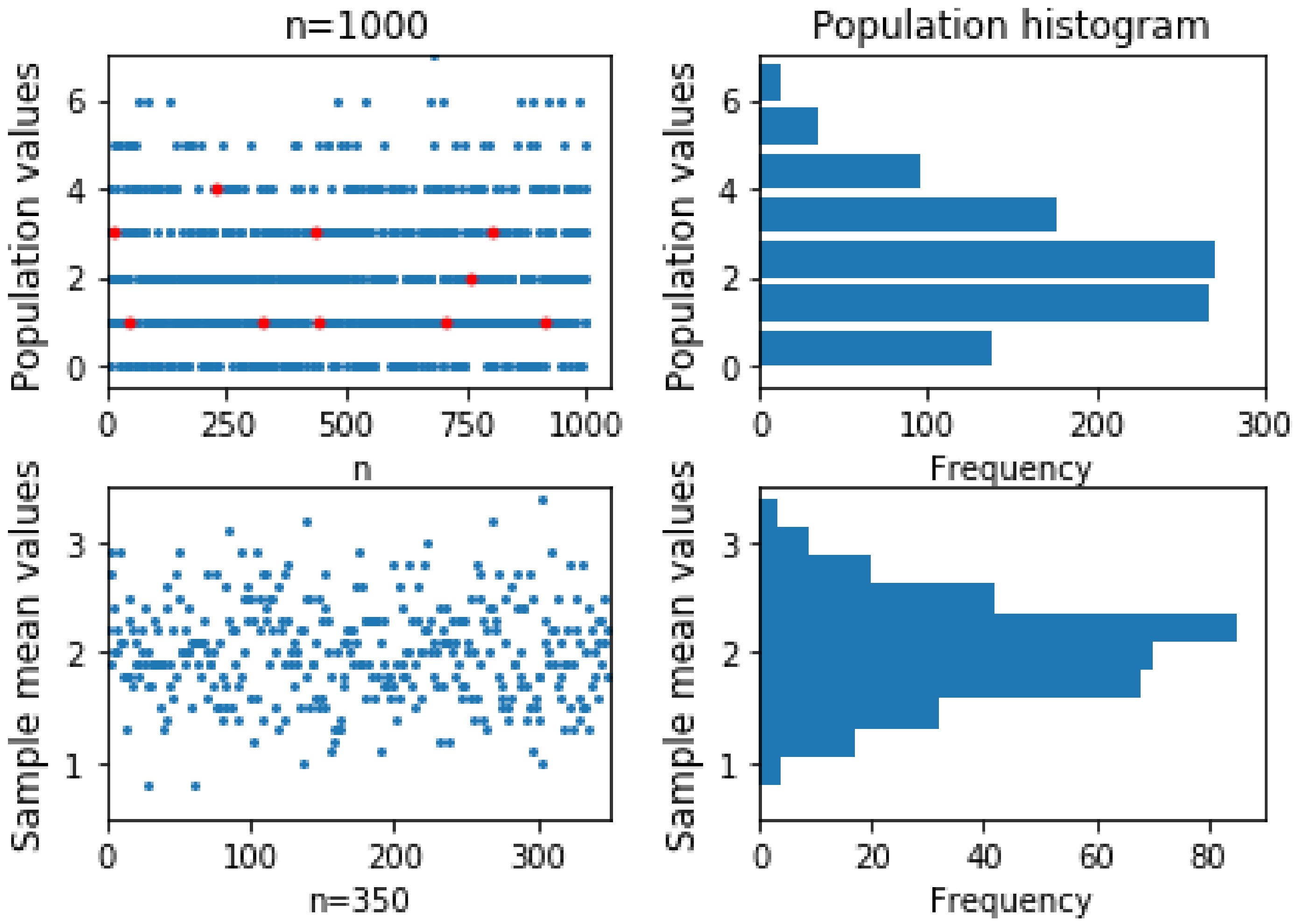


Important probability distributions

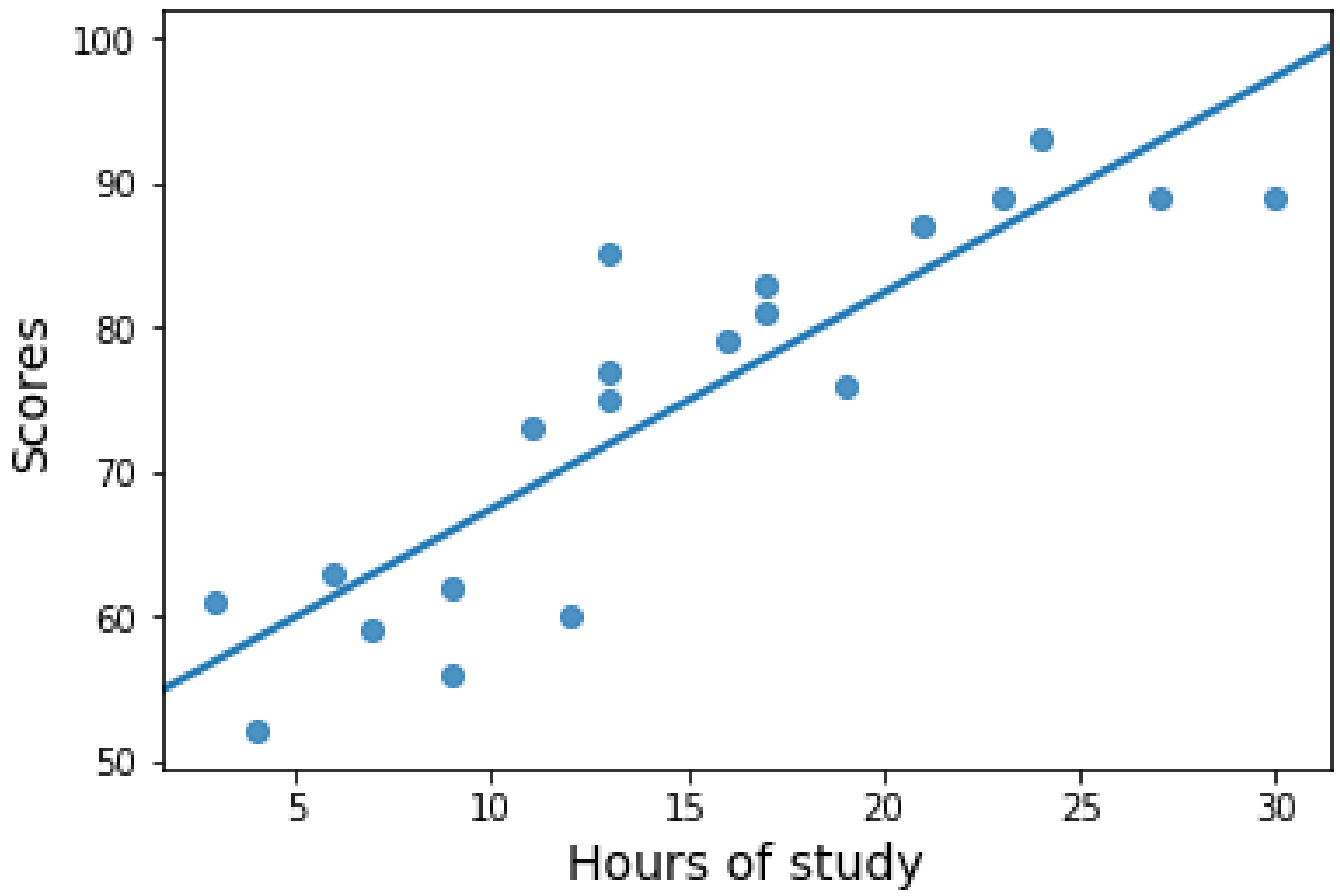


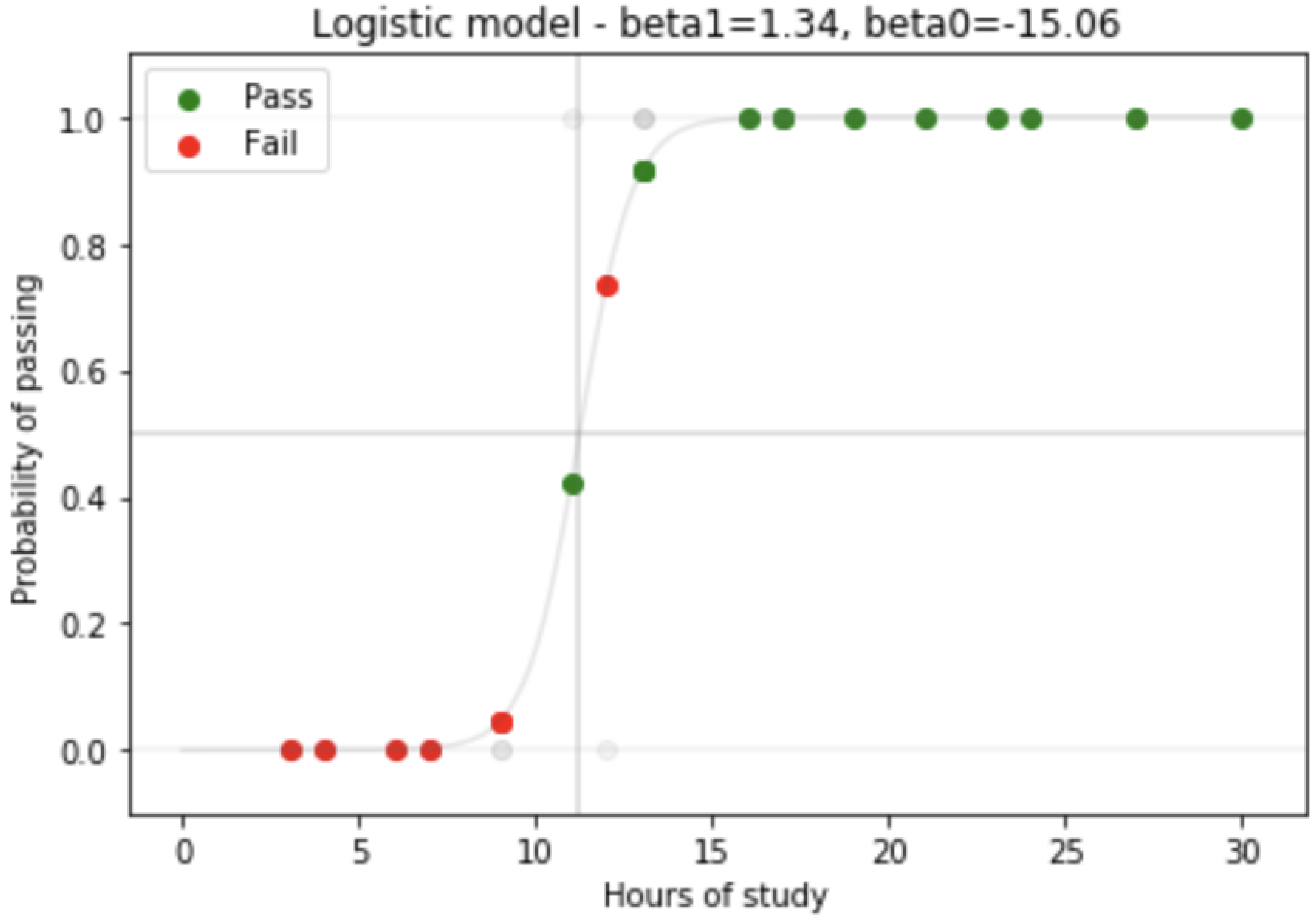
The most important results





Linear and logistic regression





Keep learning at DataCamp!

FOUNDATIONS OF PROBABILITY IN PYTHON