

# Probabilistic logic and statistical inference

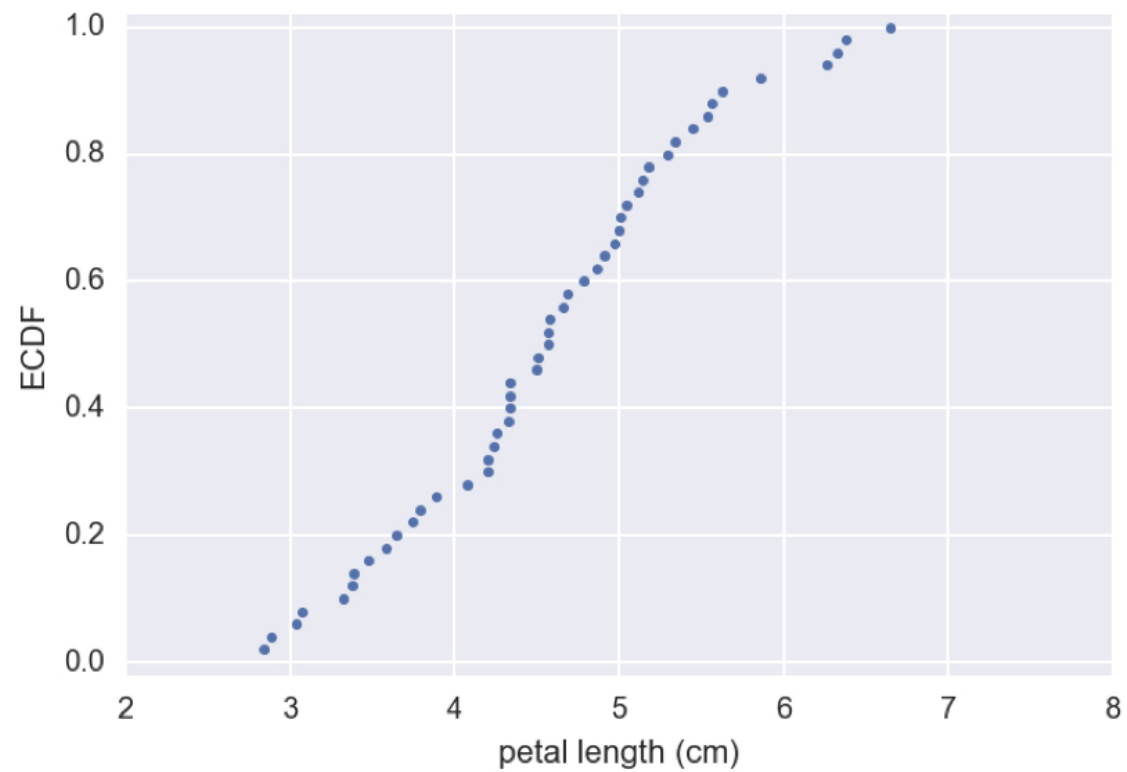
STATISTICAL THINKING IN PYTHON (PART 1)



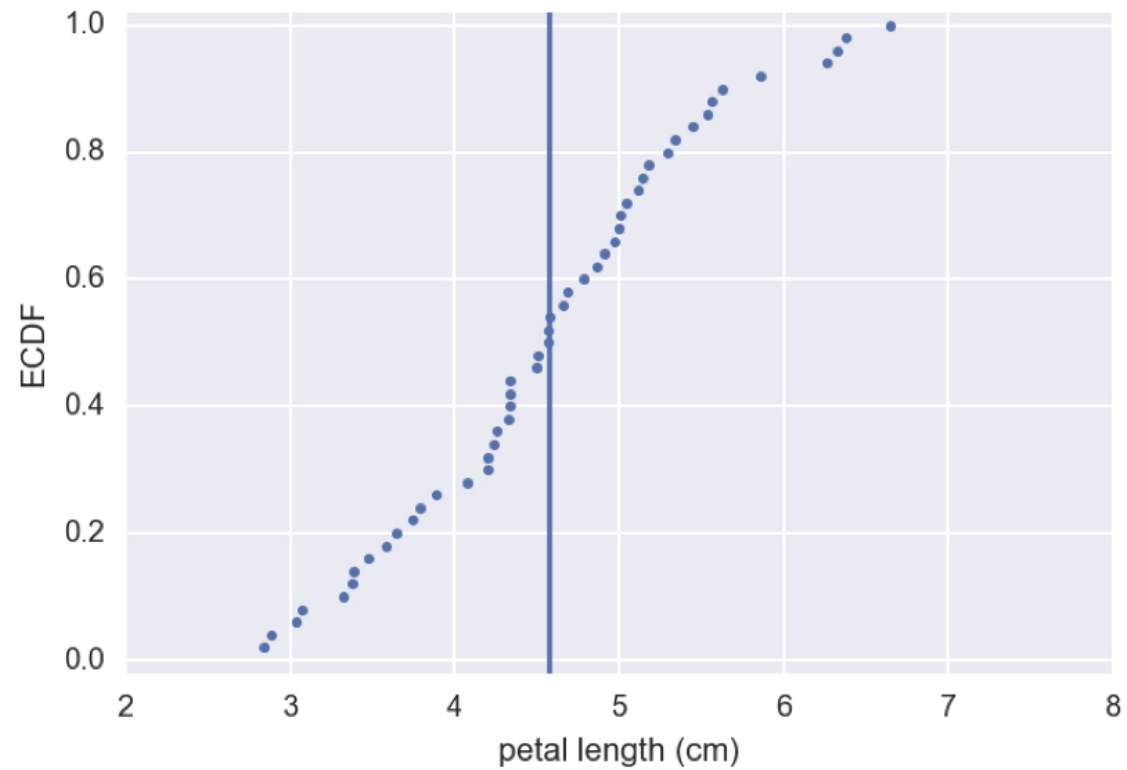
**Justin Bois**

Lecturer at the California Institute of  
Technology

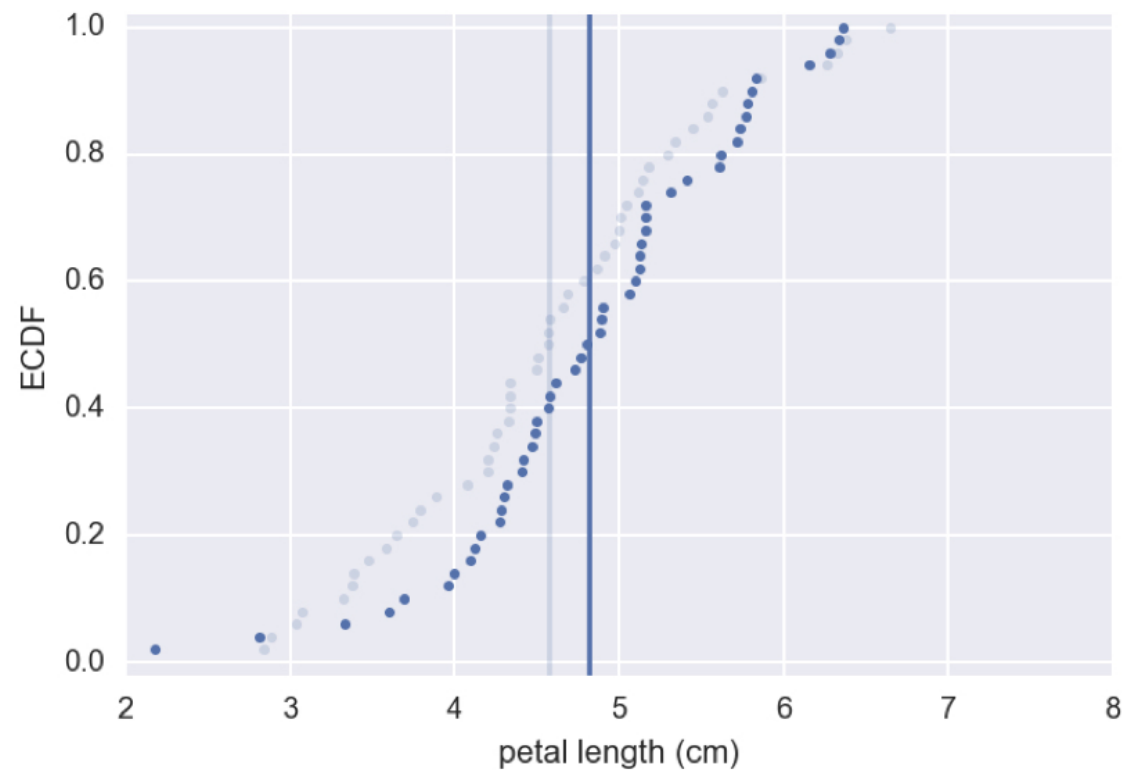
# 50 measurements of petal length



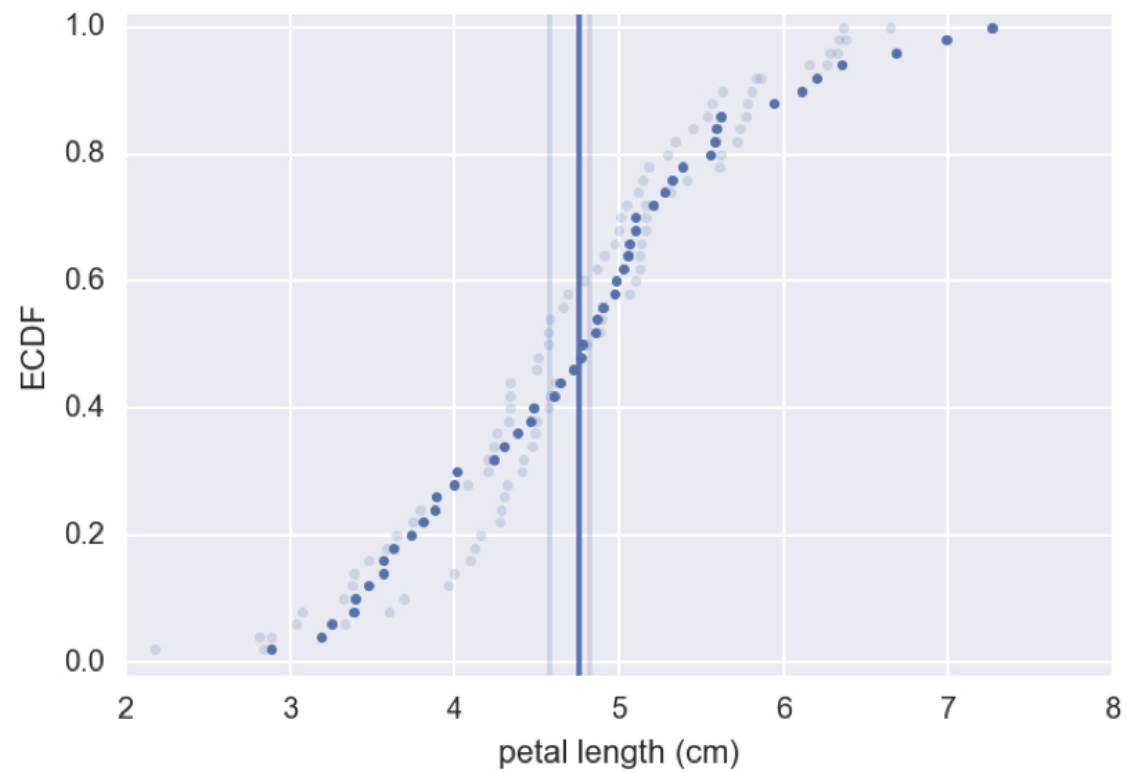
# 50 measurements of petal length



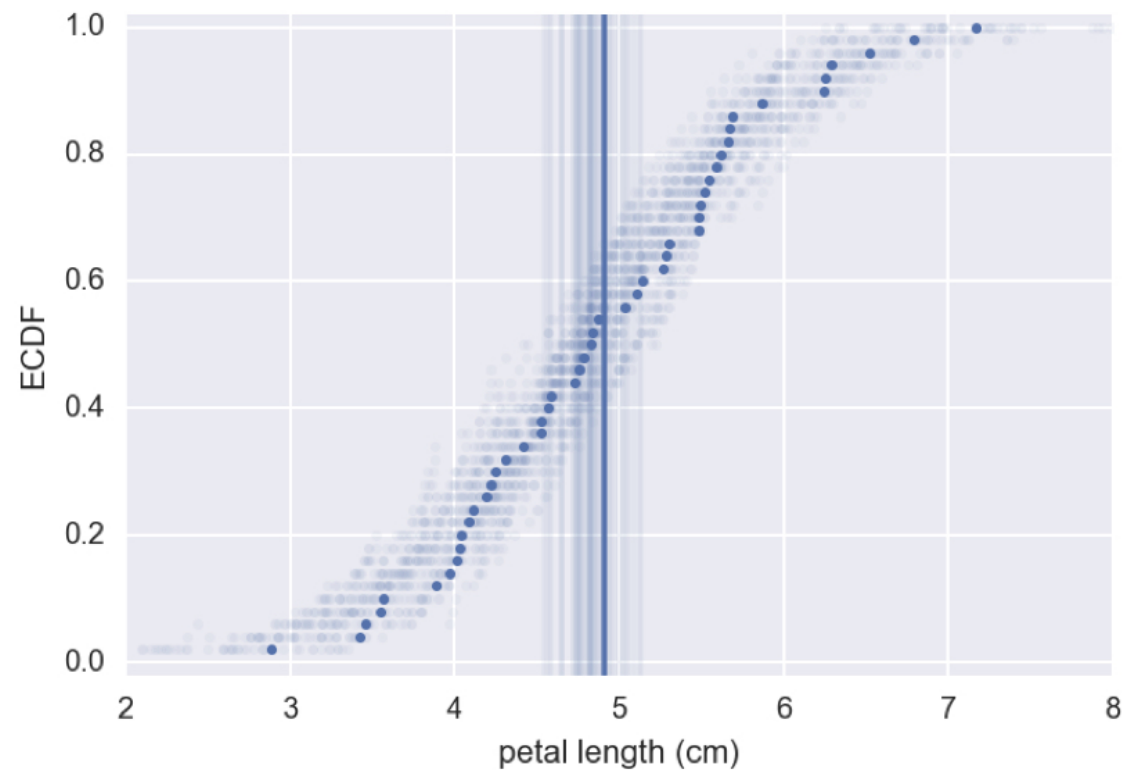
# 50 measurements of petal length



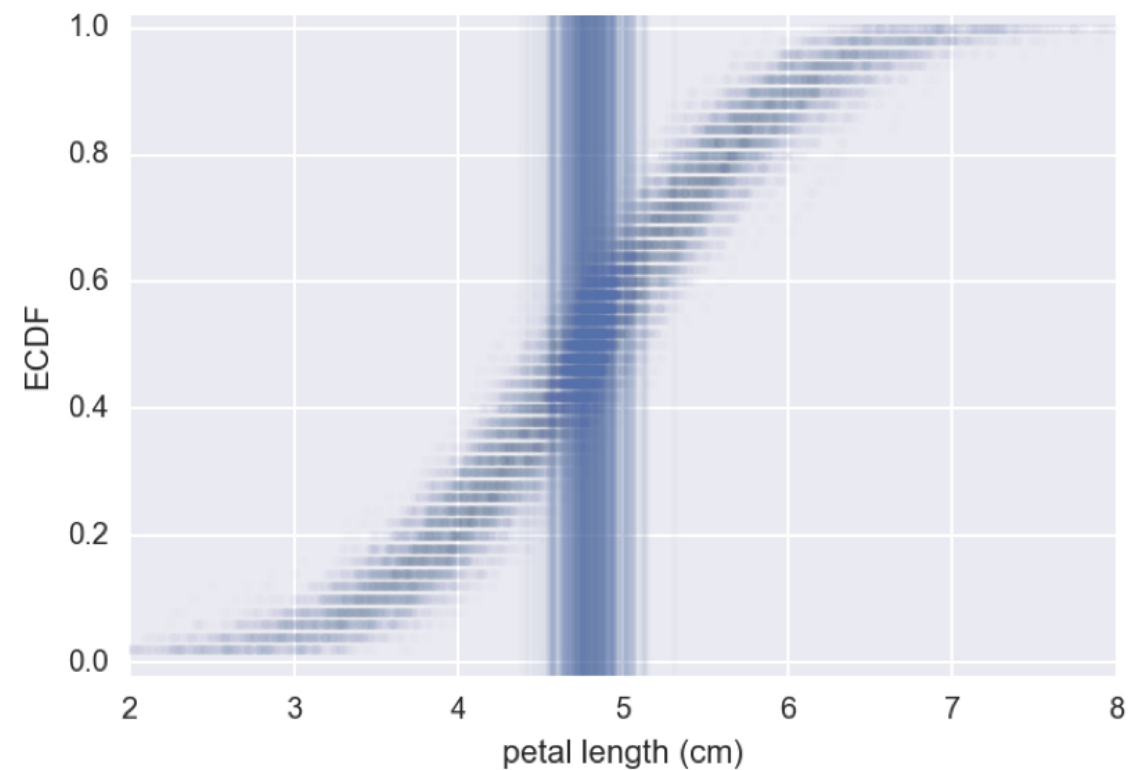
# 50 measurements of petal length



# 50 measurements of petal length



# Repeats of 50 measurements of petal length



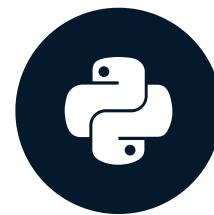
# Let's practice!

STATISTICAL THINKING IN PYTHON (PART 1)



# Random number generators and hacker statistics

STATISTICAL THINKING IN PYTHON (PART 1)



**Justin Bois**

Lecturer at the California Institute of  
Technology

# Hacker statistics

- Uses simulated repeated measurements to compute probabilities.



Blaise Pascal

<sup>1</sup> Image: artist unknown



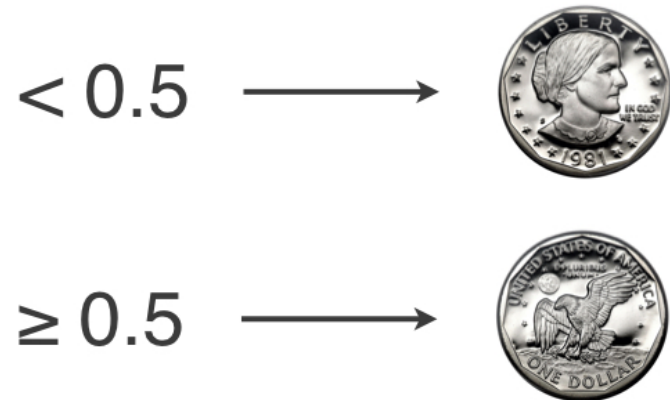
<sup>1</sup> Image: Heritage Auction

# The `np.random` module

- Suite of functions based on random number generation
- `np.random.random()` : draw a number between 0 and 1

# The np.random module

- Suite of functions based on random number generation
- `np.random.random()` : draw a number between 0 and 1



# Bernoulli trial

- An experiment that has two options, "success" (True) and "failure" (False).

# Random number seed

- Integer fed into random number generating algorithm
- Manually seed random number generator if you need reproducibility
- Specified using `np.random.seed()`



# Simulating 4 coin flips

```
import numpy as np
np.random.seed(42)
random_numbers = np.random.random(size=4)
random_numbers
```

```
array([ 0.37454012,  0.95071431,  0.73199394,  0.59865848])
```

```
heads = random_numbers < 0.5
heads
```

```
array([ True, False, False, False], dtype=bool)
```

```
np.sum(heads)
```

```
1
```

# Simulating 4 coin flips

```
n_all_heads = 0 # Initialize number of 4-heads trials
for _ in range(10000):
    heads = np.random.random(size=4) < 0.5
    n_heads = np.sum(heads)
    if n_heads == 4:
        n_all_heads += 1

n_all_heads / 10000
```

```
0.0621
```

# Hacker stats probabilities

- Determine how to simulate data
- Simulate many many times
- Probability is approximately fraction of trials with the outcome of interest

# Let's practice!

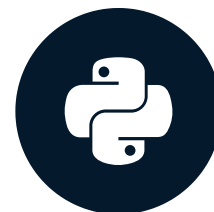
STATISTICAL THINKING IN PYTHON (PART 1)

# Probability distributions and stories: The Binomial distribution

STATISTICAL THINKING IN PYTHON (PART 1)

**Justin Bois**

Lecturer at the California Institute of  
Technology









# Probability mass function (PMF)

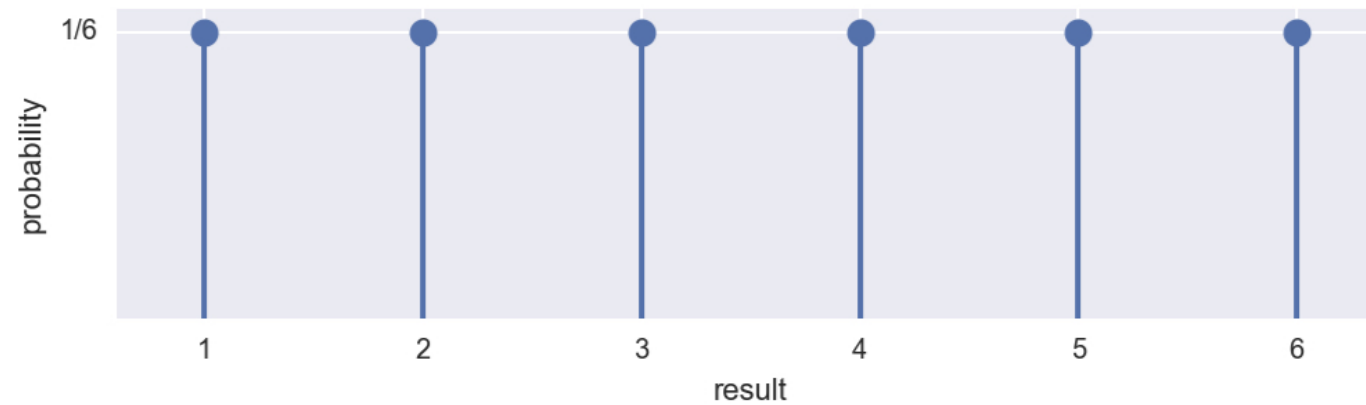
- The set of probabilities of discrete outcomes

# Discrete Uniform PMF

Tabular

					
1/6	1/6	1/6	1/6	1/6	1/6

Graphical



# Probability distribution

- A mathematical description of outcomes



# Discrete Uniform distribution: the story

The outcome of rolling a single fair die is

- Discrete
- Uniformly distributed.

# Binomial distribution: the story

- The number  $r$  of successes in  $n$  Bernoulli trials with probability  $p$  of success, is Binomially distributed
- The number  $r$  of heads in 4 coin flips with probability 0.5 of heads, is Binomially distributed

# Sampling from the Binomial distribution

```
np.random.binomial(4, 0.5)
```

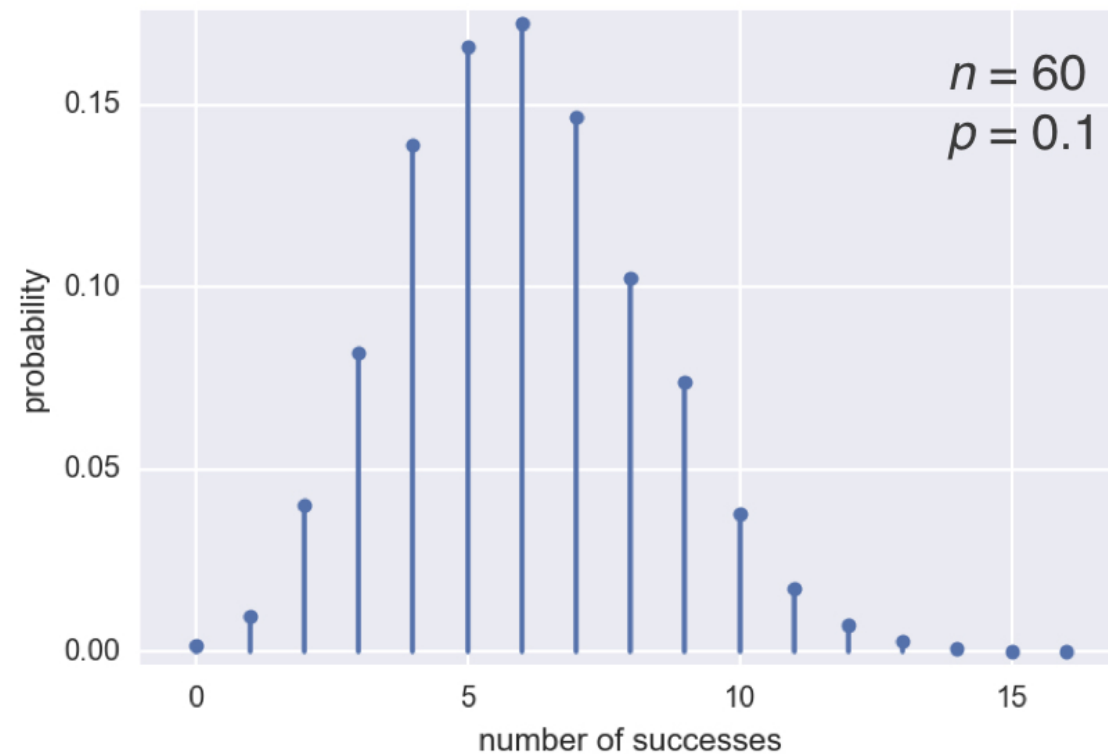
```
2
```

```
np.random.binomial(4, 0.5, size=10)
```

```
array([4, 3, 2, 1, 1, 0, 3, 2, 3, 0])
```

# The Binomial PMF

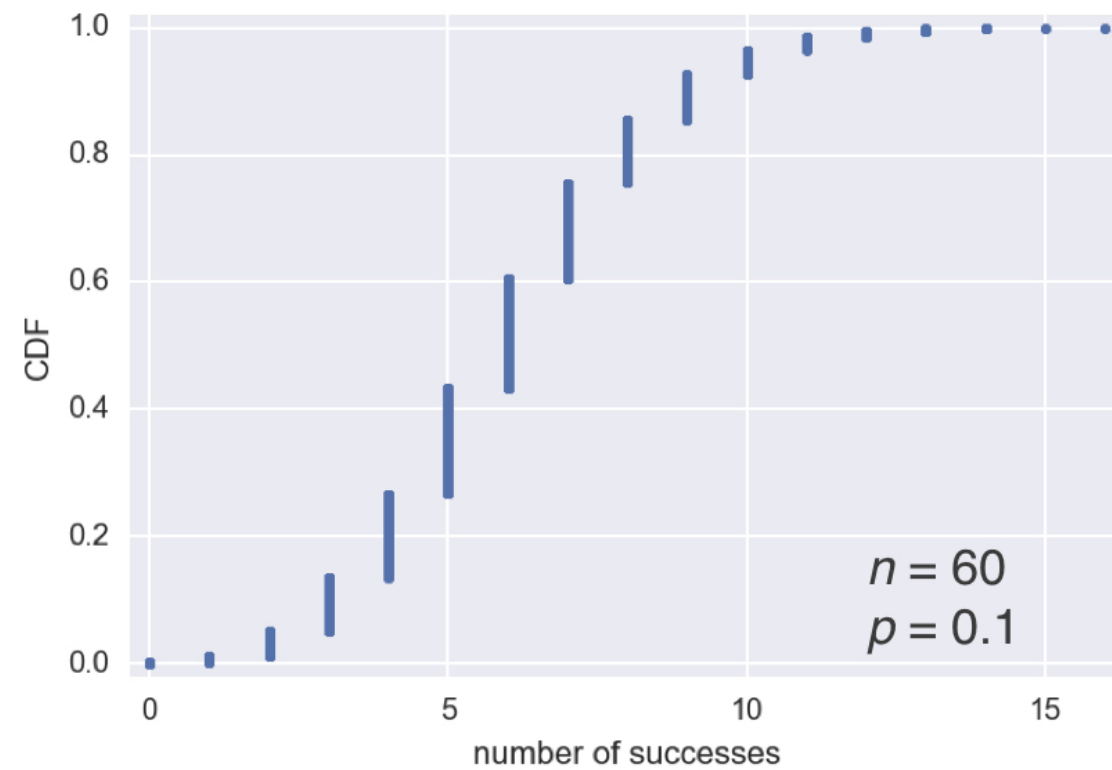
```
samples = np.random.binomial(60, 0.1, size=10000)
n = 60
p = 0.1
```



# The Binomial CDF

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
x, y = ecdf(samples)
_ = plt.plot(x, y, marker='.', linestyle='none')
plt.margins(0.02)
_ = plt.xlabel('number of successes')
_ = plt.ylabel('CDF')
plt.show()
```

# The Binomial CDF

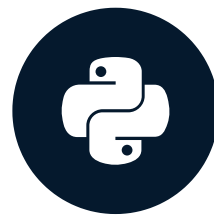


# Let's practice!

STATISTICAL THINKING IN PYTHON (PART 1)

# Poisson processes and the Poisson distribution

STATISTICAL THINKING IN PYTHON (PART 1)



**Justin Bois**

Lecturer at the California Institute of  
Technology



# Poisson process

- The timing of the next event is completely independent of when the previous event happened

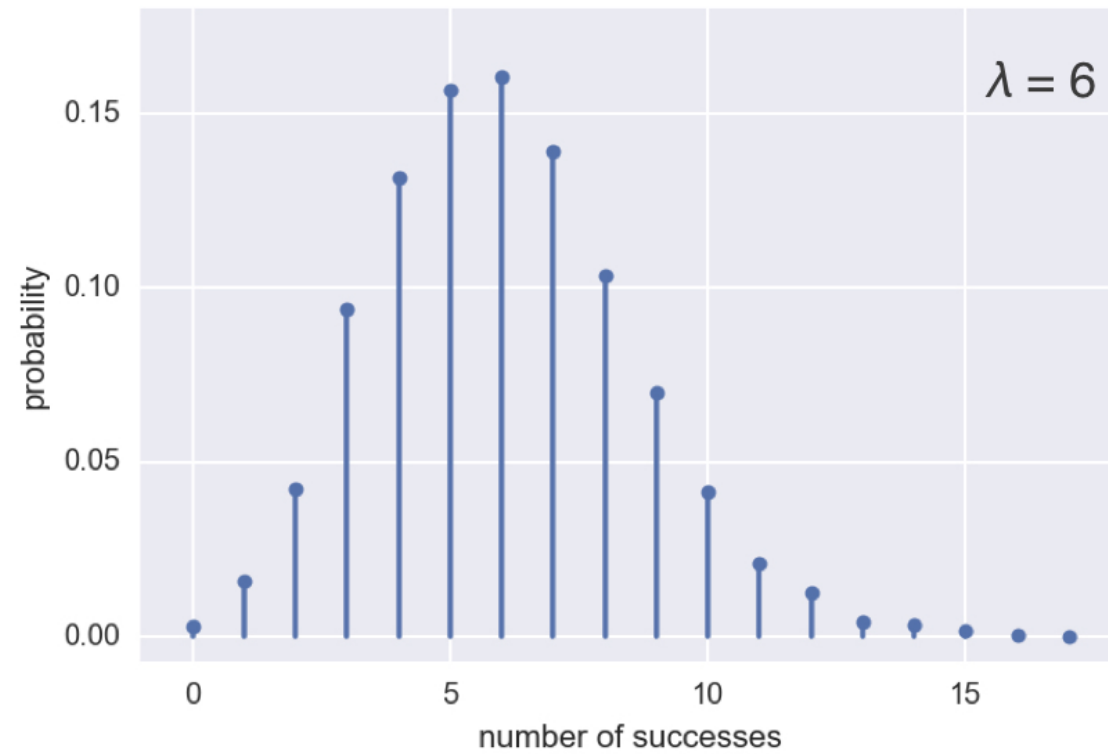
# Examples of Poisson processes

- Natural births in a given hospital
- Hit on a website during a given hour
- Meteor strikes
- Molecular collisions in a gas
- Aviation incidents
- Buses in Poissonville

# Poisson distribution

- The number  $r$  of arrivals of a Poisson process in a given time interval with average rate of  $\lambda$  arrivals per interval is Poisson distributed.
- The number  $r$  of hits on a website in one hour with an average hit rate of 6 hits per hour is Poisson distributed.

# Poisson PMF



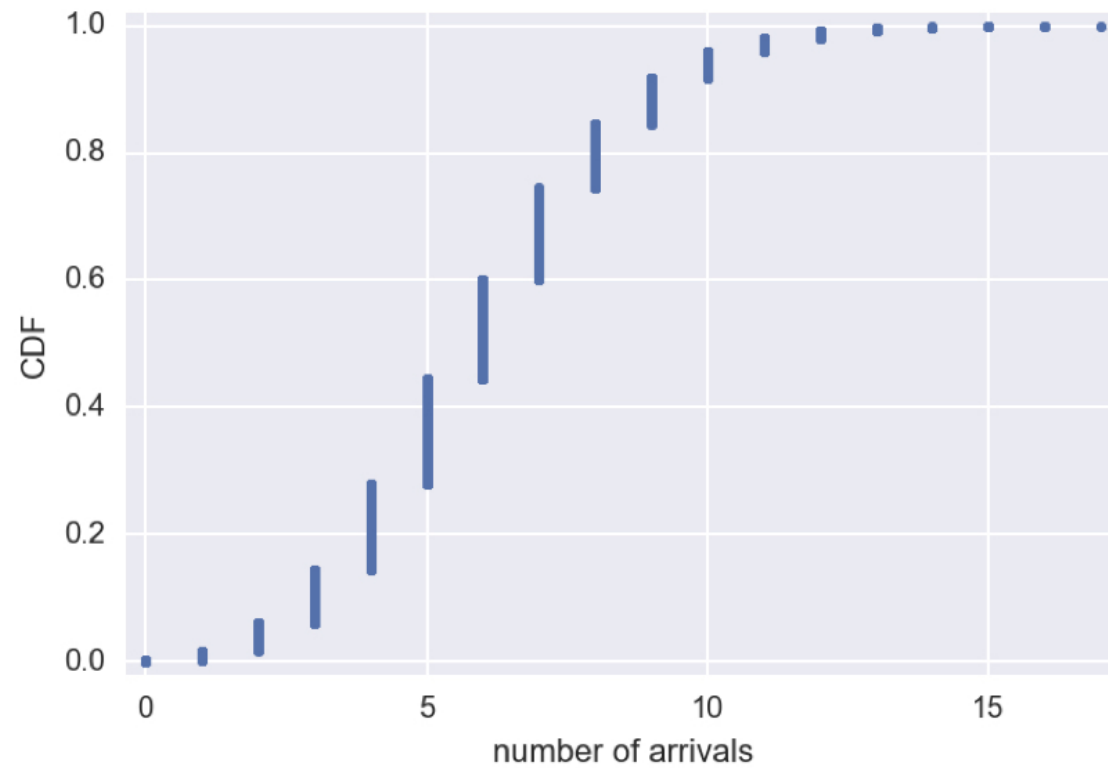
# Poisson Distribution

- Limit of the Binomial distribution for low probability of success and large number of trials.
- That is, for rare events.

# The Poisson CDF

```
samples = np.random.poisson(6, size=10000)
x, y = ecdf(samples)
_ = plt.plot(x, y, marker='.', linestyle='none')
plt.margins(0.02)
_ = plt.xlabel('number of successes')
_ = plt.ylabel('CDF')
plt.show()
```

# The Poisson CDF



# Let's practice!

STATISTICAL THINKING IN PYTHON (PART 1)