Chapter 2: Random Numbers and Probability

Created	@April 16, 2024 6:41 PM
Class	Introdiction to Statistics in Python

Measuring Chance with Probability

Introduction

- People talk about chance frequently in various scenarios
- How do we measure chance?

Probability

- Measure chances of an event using probability
- Calculate probability: (number of ways the event can happen) / (total number of possible outcomes)

$$P(event) = \frac{Number\ of\ ways\ events\ can\ happen}{Total\ number\ of\ possibleoutcomes}$$

- Example: Flipping a coin
- Probability of getting heads = 1/2 = 50%
- Range of probability: 0% (impossible) to 100% (certain)

Complex Scenario: Selecting a Salesperson

- Scenario: Sending a salesperson to a meeting with a potential client
- Method: Put names on tickets, randomly pick one from a box
- Example: Brian's name is picked, probability = 1/4 = 25%
- Recreating in Python with `sample()` method; `sample()` randomly selects one row from the DataFrame

Setting Random Seed

- To ensure consistent results, set the random seed using `np.random.seed()`
- Seed is the starting point for the random number generator
- Same seed generates the same random value each time

Sampling Without Replacement

- Scenario: Another client meeting at the same time, need to pick another salesperson
- Cannot pick the same person twice (Brian is already picked)
- Sampling without replacement: Not replacing the name already picked
- Example: Claire is picked, probability = 1/3 ≈ 33%
- In Python: Pass `2` to `sample()` method which will give us 2 rows of DataFrame

Sampling With Replacement

- Scenario: Meetings happening on different days, same person can attend both
- Sampling with replacement: Returning the picked name back to the box
- Example: Claire is picked again, probability = 1/4 = 25%
- In Python: Set `replace=True` in `sample()` method to allow rows to be picked multiple times

Independent Events

- Two events are independent if the probability of the second event is not affected by the outcome of the first event
- Sampling with replacement: Each pick is independent
- Sampling without replacement: Each pick is dependent on the previous pick(s)

Dependent events

- Two events are dependent if the probability of the second event is affected by the outcome of the first event.

Probability Distributions

Introduction

- Taking a deeper dive into probability and probability distributions

Dice Roll Scenario

- Consider rolling a standard six-sided die
- There are six possible outcomes (1, 2, 3, 4, 5, 6)
- Each outcome has a 1/6 ≈ 17% probability of being rolled
- Example of a probability distribution

Probability Distribution

- Describes the probability of each possible outcome in a scenario

Expected Value

- The mean of a probability distribution
- Calculated by multiplying each value by its probability and summing
- For a fair die, the expected value is 3.5

Visualizing Probability Distributions

- Use a bar plot, where each bar represents an outcome
- Bar height represents the probability of that outcome

Calculating Probabilities

- Probabilities can be calculated by taking areas of the probability distribution
- Example: Probability of rolling ≤ 2 on a fair die is 1/3 (area of bars for 1 and 2)

Biased Die Example

- A die where 2 is replaced with another 3
- Probability of rolling 2 is now 0%, and probability of rolling 3 is 1/3
- Expected value changes (slightly higher than a fair die)
- Bars in the visualization are no longer even

Discrete Probability Distributions

- Represent situations with discrete (countable) outcomes
- Example: Counting dots on a die (can't roll 1.5 or 4.3)

Discrete Uniform Distribution

- Special case where all outcomes have the same probability (fair die)

Sampling from Probability Distributions

- Can sample from probability distributions, similar to sampling names
- Example: Simulating 10 rolls of a fair die by sampling from a DataFrame

```
# Create a DataFrame representing a fair die
die = pd.Series([1/6, 1/6, 1/6, 1/6, 1/6, 1/6], index=[1, 2,
3, 4, 5, 6])
```

```
# Sample 10 rolls from the die
rolls = die.sample(10, replace=True)

# Visualize the rolls
plt.hist(rolls.index, bins=np.linspace(0.5, 6.5, 7), edgecolo
r='black')
plt.xticks(np.arange(1, 7))
plt.xlabel('Roll Value')
plt.ylabel('Count')
plt.show()
```

Law of Large Numbers

- As the sample size increases, the sample mean approaches the theoretical mean
- Demonstrated by simulating more rolls (100, 1000) and observing the distributions

```
# Simulate 100 rolls
rolls_100 = die.sample(100, replace=True)

# Simulate 1000 rolls
rolls_1000 = die.sample(1000, replace=True)
```

Continuous Probability Distributions

Introduction

- Discrete distributions model countable variables
- How can we model continuous variables?

Bus Arrival Example

- City bus arrives every 12 minutes
- Waiting time can range from 0 minutes (just arrived) to 12 minutes (just missed)
- An infinite number of possible waiting times (1 min, 1.5 min, 1.53 min, etc.)

Continuous Uniform Distribution

- Represented by a flat continuous line
- Equal probability of waiting any time from 0 to 12 minutes

Calculating Probabilities

- Calculate probability by taking the area under the curve
- Example: Probability of waiting between 4 and 7 minutes
- Width = 7 - 4 = 3
- Height = 1/12
- Area = Width × Height = 3/12 = 0.25 (25%)

Python Implementation

```
from scipy.stats import uniform

# Probability of waiting <= 7 minutes in total 12 minutes
uniform.cdf(7, 0, 12)  # Output: 0.5833333333333334

# Probability of waiting > 7 minutes
1 - uniform.cdf(7, 0, 12)  # Output: 0.4166666666666667

# Probability of waiting between 4 and 7 minutes
uniform.cdf(7, 0, 12) - uniform.cdf(4, 0, 12)  # Output: 0.25

# Probability of waiting between 0 and 12 minutes
12 * (1/12)  # Output: 1.0 (100%)
```

Generating Random Values

```
uniform.rvs(0, 5, size=10)  # Generate 10 random values between
```

Other Continuous Distributions

- Continuous distributions can take various shapes
- Some values may have higher probabilities than others
- Area under the curve must always equal 1
- Examples: Normal distribution, Exponential distribution

Conclusion

Concept	Description
Continuous Uniform Distribution	Flat line, equal probability across the range
Calculating Probabilities	Area under the curve
`scipy.stats.uniform`	Python implementation of the uniform distribution
`cdf`	Cumulative Distribution Function
`rvs`	Generate random values
Other Distributions	Normal, Exponential, etc.

The Binomial Distribution

Binary Outcomes

- Example: Flipping a coin (heads or tails)
- Binary outcome: Two possible values (0/1, success/failure, win/loss)

Simulating Coin Flips in Python

```
from scipy.stats import binom

binom.rvs(# of coins, probability of heads/success, size=# of
trials)
# returns an array of results
# Flip 1 coin, 50% probability of heads, 1 time
binom.rvs(1, 0.5, size=1)

# Flip 1 coin, 50% probability of heads, 8 times
binom.rvs(1, 0.5, size=8)
# gives us an array of 8 ones(success) and zeros(failure)

# Flip 8 coins, 50% probability of heads, 1 time
binom.rvs(8, 0.5, size=1)
# gives us one number, which is the total number of heads/suc
cesses

# Flip 3 coins, 50% probability of heads, 10 times
binom.rvs(3, 0.5, size=10)
# returns 10 numbers each representing the total number of he
ad from # each set of flips

# Simulating with a biased coin (25% probability of heads)
binom.rvs(1, 0.25, size=8)
```

Binomial Distribution

- Describes the probability of the number of successes in a sequence of independent trials
- Can tell the probability of getting a certain number of heads in a sequence of coin flips
- Discrete distribution (countable outcome)

Parameters

- n: Total number of trials
- p: Probability of success
- n and p are the third and second arguments of `binom.rvs`

Visualizing the Distribution

- Example: 10 coin flips
- Highest probability of getting 5 heads
- Smaller probability of getting 0 or 10 heads

Calculating Probabilities

```
binom.pmf(num_heads, num_trials, prob_of_heads)
# calculates the probability of getting num_heads in num_trials

# Probability of getting exactly 7 heads out of 10 coins
binom.pmf(7, 10, 0.5)

binom.cdf(num_heads, num_trials, prob_of_heads)
# calculates the probability of getting less than or equal to
# num_heads in num_trials

# Probability of getting 7 or fewer heads out of 10 coins
binom.cdf(7, 10, 0.5)

# Probability of getting more than 7 heads out of 10 coins
1 - binom.cdf(7, 10, 0.5)
```

Expected Value

- Expected number of successes
- Calculated as n * p

- Example: For 10 coin flips with p=0.5, expected value = 10 * 0.5 = 5

Independence Assumption

- Binomial distribution assumes trials are independent
- Outcomes should not affect the probabilities of subsequent trials
- Example: Sampling without replacement violates the independence assumption

Conclusion

Function	Purpose
`binom.rvs`	Generate random variates (simulate trials)
`binom.pmf`	Probability Mass Function (probability of exact number of successes)
`binom.cdf`	Cumulative Distribution Function (probability of successes <= value)
`n * p`	Expected value (expected number of successes)