20200212【Data Analyst Nanodegree】P04M01L08

Part 04 : Practical Statistics
Learn how to apply inferential statistics and probability to important, real-world scenarios, such as analyzing A/B tests and building supervised learning models.

Module 01: Practical Stats

Lesson 08: Python Probability Practice

> Take what you have learned in the last lessons and put it to practice in Python.

01. Introduction

Now that you've seen some examples and the math involved with them, you're going to apply this knowledge to problems using Python.

This lesson includes screencast and Jupyter notebooks to help you practice using Python to explore the topics of probability you just learned.

02. Simulating Coin Flips

Simulating Coin Flips

```
import numpy as np

# Flip 10000 with fair coin, and P(X=0.5 and P(1)=0.5
np.random.randint(2, size=10000).mean()
```

```
0.4972
```

```
# Flip 10000 with loaded Coin and P(X=0.6 and P(1)=0.3
np.random.choice([0,1], size=10000, p=[0.6, 0.2]).mean()
```

```
0.2026
```

03. Probability Quiz

Probability Quiz

In this quiz, you will **simulate coin flips** and **do rolls** to compute proportions for the following outcomes.

1. Two fair coin flips produce exactly two heads
2. Three fair coin flips produce exactly one head
3. Three biased coin flips with P(H) = 0.6 produce exactly one head
4. A die rolls an even number
5. Two dice roll a double

Then, you'll compare these proportions with probabilities in the quizzes below.

When simulating coin flips, use 0 to represent heads and 1 to represent tails. When simulating die rolls, use the correct integers to match the numbers on the sides of a standard 6 sided die.

Also, notice that you can look at the solutions if you get stuck by clicking the orange button in the top left.

Solution

```
# simulating coin_flips
import numpy as np

# Two fair coin flips produce exactly two heads
coins = np.random.randint(2, size=(int(1e5), 2))
prop = (coins.sum(axis=1) == 2).mean()
print("Proportion: {}%".format(round(prop*100, 2)))
```

```
Proportion: 24.99%
```

```
# Three fair coin flips produce exactly one head
coins = np.random.randint(2, size=(int(1e5), 3))
prop = (coins.sum(axis=1) == 1).mean()
print("Proportion: {}%".format(round(prop*100, 2)))
```

```
Proportion: 37.47%
```

```
# Three biased coin flips with P(H) = 0.6 produce exactly one head
coins = np.random.choice([0, 1], size=(int(1e5), 3), p=[0.4, 0.6])
prop = (coins.sum(axis=1) == 1).mean()
print("Proportion: {}%".format(round(prop*100, 2)))
```

```
Proportion: 28.62%
```

```
# A die rolls an even number
dice = np.random.randint(1, 7, size=int(1e5))
prop = (dice%2 == 0).mean()
print("Proportion: {}%".format(round(prop*100, 2)))
```

```
Proportion: 49.82%
```

```
# Two dice roll a double
dice = np.random.randint(1, 7, size=(int(1e5), 2))
dice[:,1] = dice[:, 0]
prop = (dice.sum(axis=1) == 0).mean()
print("Proportion: {}%".format(round(prop*100, 2)))
```

```
Proportion: 16.68%
```

04. Simulating Many Coin Flips

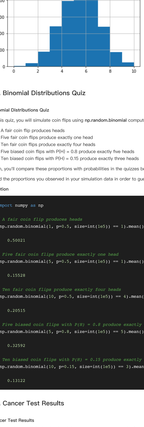
Simulating Many Coin Flips

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

# Samples are drawn from a binomial distribution with specified
# parameters, n trials and p probability of success where
# n an integer >= 0 and p is in the interval [0,1].
np.random.binomial(n=10, p=0.5, size=10000).mean()
```

```
4.9799
```

```
# Plot a histogram of the outcomes in this simulation
plt.hist(np.random.binomial(n=10, p=0.5, size=10000));
```



05. Binomial Distributions Quiz

Binomial Distributions Quiz

In this quiz, you will simulate coin flips using **np.random.binomial** compute proportions for the following outcomes.

1. A fair coin flip produces heads
2. Five fair coin flips produce exactly one head
3. Ten fair coin flips produce exactly four heads
4. Five biased coin flips with P(H) = 0.8 produce exactly five heads
5. Ten biased coin flips with P(H) = 0.15 produce exactly three heads

Then, you'll compare these proportions with probabilities in the quizzes below.

Used the proportions you observed in your simulation data in order to guess the probabilities of the following outcomes.

Solution

```
import numpy as np

# A fair coin flip produces heads
(np.random.binomial(1, p=0.5, size=int(1e5)) == 1).mean()
```

```
0.50021
```

```
# Five fair coin flips produce exactly one head
(np.random.binomial(5, p=0.5, size=int(1e5)) == 1).mean()
```

```
0.15528
```

```
# Ten fair coin flips produce exactly four heads
(np.random.binomial(10, p=0.5, size=int(1e5)) == 4).mean()
```

```
0.20515
```

```
# Five biased coin flips with P(H) = 0.8 produce exactly five heads
(np.random.binomial(5, p=0.8, size=int(1e5)) == 5).mean()
```

```
0.32762
```

```
# Ten biased coin flips with P(H) = 0.15 produce exactly three heads
(np.random.binomial(10, p=0.15, size=int(1e5)) == 3).mean()
```

```
0.13122
```

06. Cancer Test Results

Cancer Test Results

In this section, you'll find a simulated dataset on cancer test results for patients and whether they really have cancer. Explore *cancer_test_data.csv* in the Jupyter notebook to answer the following questions.

- How many patients are there in total?
- How many patients have cancer?
- How many patients do not have cancer?
- What proportion of patients have cancer?
- What proportion of patients don't have cancer?
- What proportion of patients with cancer test positive?
- What proportion of patients with cancer test negative?
- What proportion of patients without cancer test positive?
- What proportion of patients without cancer test negative?

Solution

```
import numpy as np
import pandas as pd

df = pd.read_csv('cancer_test_data.csv')

# How many patients are there in total?
df.patient_id.count()
```

```
2914
```

```
# How many patients have cancer?
df.query('has_cancer == True').has_cancer.count()
```

```
306
```

```
# How many patients do not have cancer?
df.query('has_cancer == False').patient_id.count()
```

```
2608
```

```
# What proportion of patients have cancer?
round(df.query('has_cancer == True').patient_id.count() / df.patient_id.count(), 3)
```

```
0.105
```

```
# What proportion of patients don't have cancer?
round(df.query('has_cancer == False').patient_id.count() / df.patient_id.count(), 3)
```

```
0.895
```

```
# What proportion of patients with cancer test positive?
round(df.query('has_cancer == True and test_result == "Positive"').patient_id.count() / df.query('has_cancer == True').patient_id.count(), 3)
```

```
0.905
```

```
# What proportion of patients with cancer test negative?
round(df.query('has_cancer == True and test_result == "Negative"').patient_id.count() / df.query('has_cancer == True').patient_id.count(), 3)
```

```
0.095
```

```
# What proportion of patients without cancer test positive?
round(df.query('has_cancer == False and test_result == "Positive"').patient_id.count() / df.query('has_cancer == False').patient_id.count(), 3)
```

```
0.204
```

```
# What proportion of patients without cancer test negative?
round(df.query('has_cancer == False and test_result == "Negative"').patient_id.count() / df.query('has_cancer == False').patient_id.count(), 3)
```

```
0.796
```

07. Conditional Probability & Bayes Rule Quiz

Conditional Probability & Bayes Rule Quiz

In the previous section, you found the following proportions from the cancer results dataset.

Patients with cancer: 0.105
Patients without cancer: 0.895
Patients with cancer who tested positive: 0.905
Patients with cancer who tested negative: 0.095
Patients without cancer who tested positive: 0.204
Patients without cancer who tested negative: 0.796

Based on the above proportions observed in the data, we can assume the following probabilities.

Probability	Meaning
$P(cancer) = 0.105$	Probability a patient has cancer
$P(\neg cancer) = 0.895$	Probability a patient does not have cancer
$P(positive \mid cancer) = 0.905$	Probability a patient with cancer tests positive
$P(negative \mid cancer) = 0.095$	Probability a patient with cancer tests negative
$P(positive \mid \neg cancer) = 0.204$	Probability a patient without cancer tests positive
$P(negative \mid \neg cancer) = 0.796$	Probability a patient without cancer tests negative

Quiz Questions

Use the probabilities given above and Bayes rule to compute the following probabilities.

1. Probability a patient who tested positive has cancer, or $P(cancer \mid positive)$
2. Probability a patient who tested positive doesn't have cancer, or $P(\neg cancer \mid positive)$
3. Probability a patient who tested negative has cancer, or $P(cancer \mid negative)$
4. Probability a patient who tested negative doesn't have cancer, or $P(\neg cancer \mid negative)$

Solution

```
import pandas as pd

df = pd.read_csv('cancer_test_data.csv')

# Probability a patient who tested positive has cancer, or P(cancer|positive)
round(df.query('has_cancer == True and test_result == "Positive"').patient_id.count()/df.query('test_result == "Positive"').patient_id.count(), 3)
```

```
0.343
```

```
# Probability a patient who tested positive doesn't have cancer, or P(~cancer|positive)
round(df.query('has_cancer == False and test_result == "Positive"').patient_id.count()/df.query('test_result == "Positive"').patient_id.count(), 3)
```

```
0.657
```

```
# Probability a patient who tested negative has cancer, or P(cancer|negative)
round(df.query('has_cancer == True and test_result == "Negative"').patient_id.count()/df.query('test_result == "Negative"').patient_id.count(), 3)
```

```
0.014
```

```
# Probability a patient who tested negative doesn't have cancer, or P(~cancer|negative)
round(df.query('has_cancer == False and test_result == "Negative"').patient_id.count()/df.query('test_result == "Negative"').patient_id.count(), 3)
```

08. Conclusion

Python Probability Conclusion

In this lesson, you put your new probability skills to practice using Python. In order to gain an understanding of some of the complex topics in the next sections, we'll be working with Python more and more.

It's often easier to understand these complex ideas through simulations and Python than it is to prove them mathematically.

You will see a bit of both in the upcoming lessons. So your practice here will definitely come in handy.

09. Appendix: Glossary

- numpy.random.**randint**(low, high=None, size=None, dtype='l')
- numpy.random.**choice**(a, size=None, replace=True, p=None)
- numpy.random.**binomial**(n, p, size=None)