Week 8: Hypothesis Testing, Correlation vs. Causation

DSUA111: Data Science for Everyone, NYU, Fall 2020

TA Jeff, `jpj251@nyu.edu`

- This slideshow: https://jjacobs.me/dsua111-sections/week-08 (https://jjacobs.me/dsua111-sections/week-08)
- All materials: https://github.com/jpowerj/dsua111-sections (https://github.com/jpowerj/dsua111-sections)

Outline

1. Hypothesis Testing Overview
2. Testing Coins
3. Null vs. Alternative Hypotheses
4. Test Statistics
5. The Normal Distribution
6. Correlation vs. Causation

Hypothesis Testing Overview

tl;dr

- If your theory was true, what would the data look like?
- Now compare that to the actual, **observed** data

Testing Coins

Example: I walk up to you and say "Hey, wanna gamble? We'll each put in a dollar, then I'll flip this coin. Heads I get the $2, tails you get the \$2"

- Xavier's Theory: I think the coin is fair. Heads and tails will come up about the same number of times
- Yasmin's Theory: I don't trust this guy, I think heads will come up more often than tails

"Suit yourself -- here, take the coin and do whatever tests you want with it!"

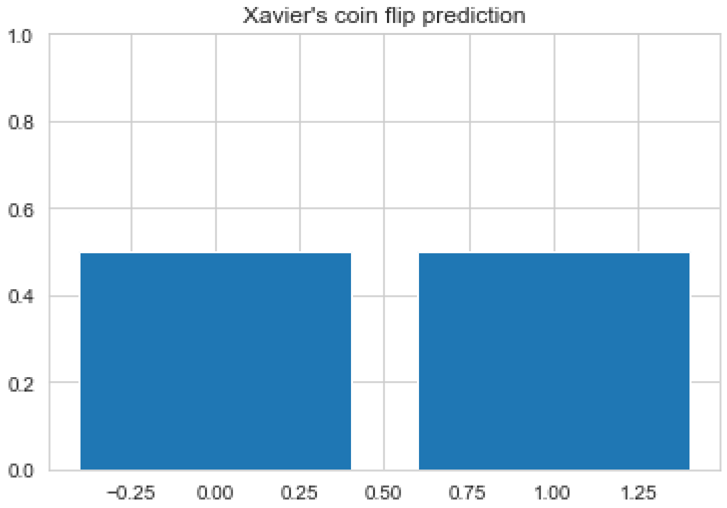
What do the two theories predict in terms of the outcome of a series of coin flips?

In [81]:

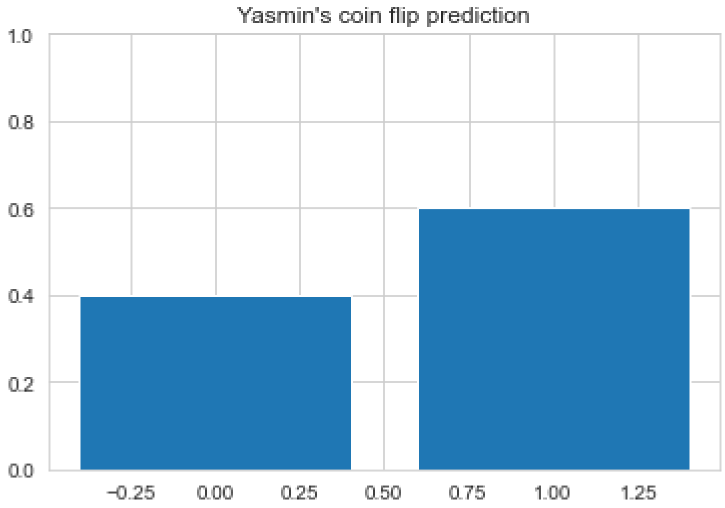
```
x_predictions = [0.5, 0.5]
y_predictions = [0.4, 0.6]

import matplotlib.pyplot as plt
def plot_prediction(prediction, who):
    plt.bar([0,1], prediction)
    plt.ylim([0,1])
    plt.title(f"{who}'s coin flip prediction")
    plt.show()
```

In [87]:
```
plot_prediction(x_predictions, "Xavier")
```



In [88]:
```
plot_prediction(y_predictions, "Yasmin")
```



- Data:

In [83]:

```python
from collections import Counter
import pandas as pd
import numpy as np
import secret_coin

def do_coin_flips(N):
    coin_flips = np.array([secret_coin.flip() for i in range(N)])
    return coin_flips

def get_flip_distributions(x_predictions, y_predictions, results):
    flip_counter = Counter(results)
    flip_counts = np.array([flip_counter[0], flip_counter[1]]) / len(results)
    flip_df = pd.DataFrame({'outcome':["Tails","Heads"],'p':flip_counts,'which':['Actual outcome','Actual outcome']})
    x_pred_df = pd.DataFrame({'outcome':["Tails","Heads"],'p':x_predictions,'which':["Xavier's prediction","Xavier's prediction"]})
    y_pred_df = pd.DataFrame({'outcome':["Tails","Heads"],'p':y_predictions,'which':["Yusuf's prediction","Yusuf's prediction"]})
    full_df = pd.concat([flip_df, x_pred_df, y_pred_df]).reset_index()
    return full_df
```

In [84]:
```python
flips_100 = do_coin_flips(100)
flips_100
```

Out[84]:
```
array([1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1,
       1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0,
       1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1,
       1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1])
```

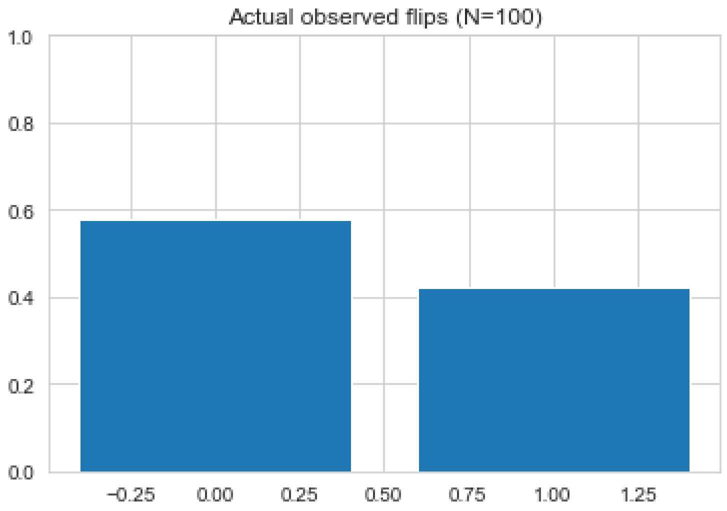
In [85]:
```python
dist_df = get_flip_distributions(x_predictions, y_predictions, flips_100)
dist_df
```

Out[85]:

	index	outcome	p	which
0	0	Tails	0.58	Actual outcome
1	1	Heads	0.42	Actual outcome
2	0	Tails	0.50	Xavier's prediction
3	1	Heads	0.50	Xavier's prediction
4	0	Tails	0.40	Yusuf's prediction
5	1	Heads	0.60	Yusuf's prediction

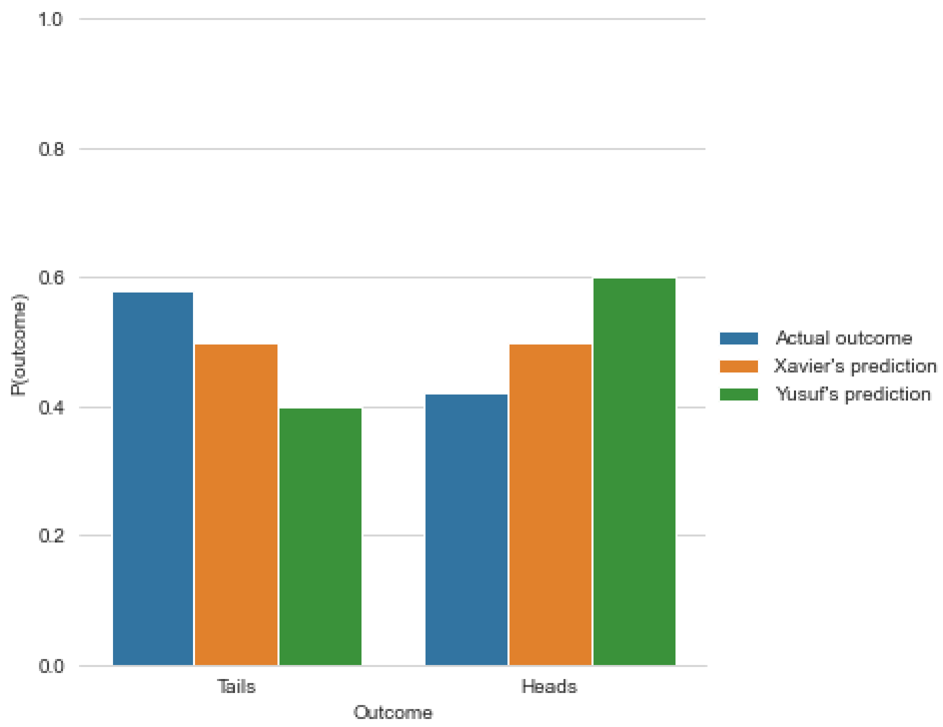
In [86]:

```
plt.bar([0,1], [dist_df.iloc[0]['p'],dist_df.iloc[1]['p']])
plt.title("Actual observed flips (N=100)")
plt.ylim([0,1])
plt.show()
```



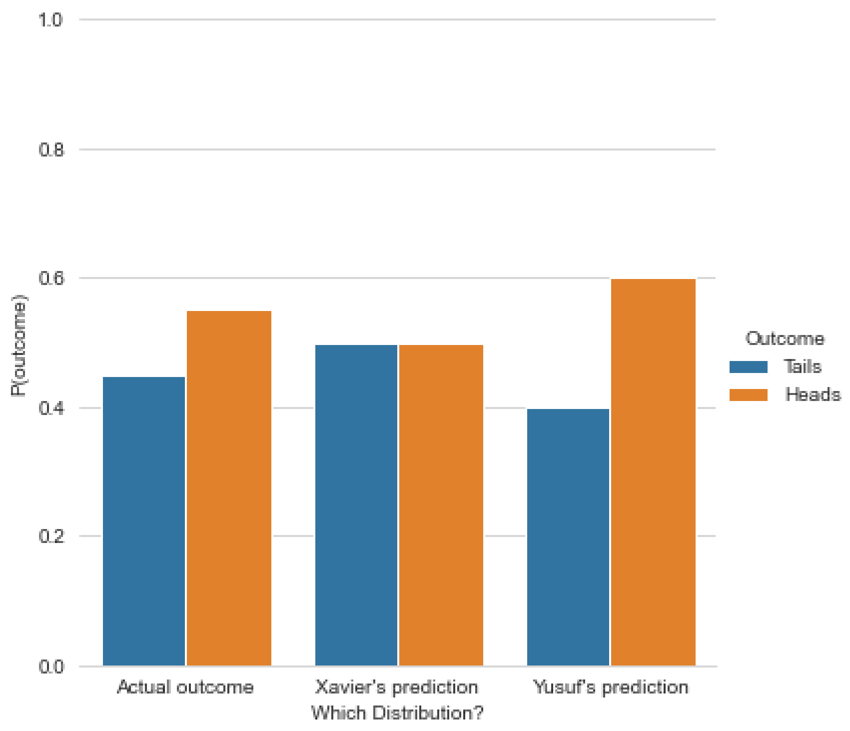
But... **how** wrong is our prediction?

In [90]:
```
import seaborn as sns; sns.set_style("whitegrid")
def plot_distributions_a(dist_df):
    g = sns.catplot(data=dist_df, kind="bar",x="outcome", y="p", hue="which")
    g.despine(left=True)
    g.set_axis_labels("Outcome", "P(outcome)")
    g.legend.set_title("")
    g.set(ylim=(0,1))
plot_distributions_a(dist_df)
```



In [70]:

```
def plot_distributions_b(dist_df):
    g = sns.catplot(
        data=dist_df, kind="bar",
        x="which", y="p", hue="outcome"
    )
    g.despine(left=True)
    g.set_axis_labels("Which Distribution?", "P(outcome)")
    g.legend.set_title("Outcome")
    g.set(ylim=(0,1))
plot_distributions_b(dist_df)
```



Hmm... that actual outcome still looks kinda sketchy. Let's try one more time

In [71]:
```
flips_100_2 = do_coin_flips(100)
flips_100_2
```

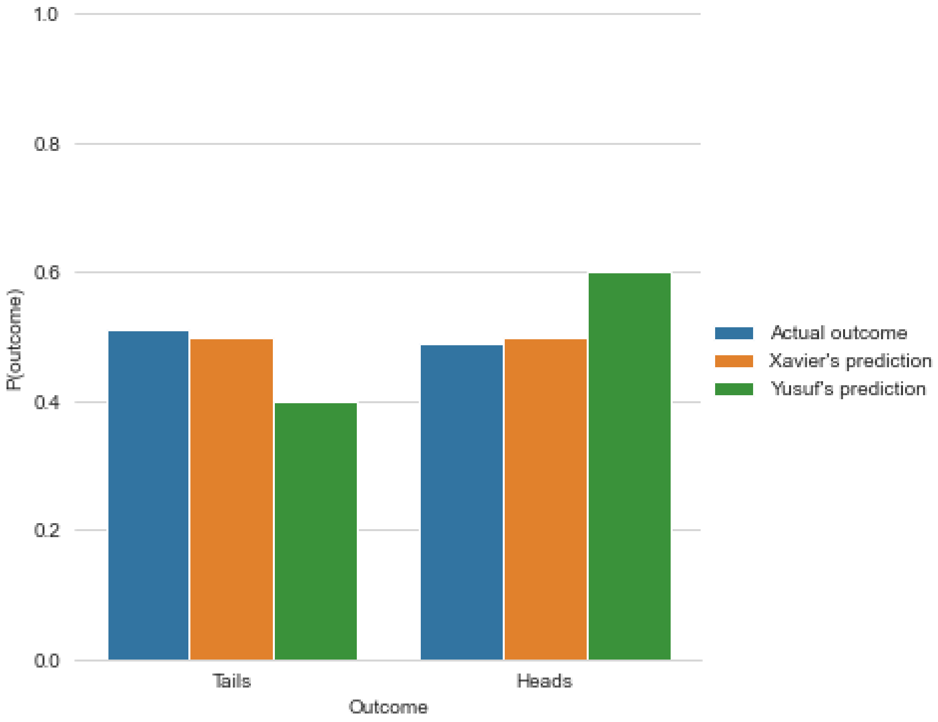
Out[71]:
```
array([0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0,
       1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0,
       1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0])
```

In [72]:
```
dist_df2 = get_flip_distributions(x_predictions, y_predictions, flips_100_2)
dist_df2
```

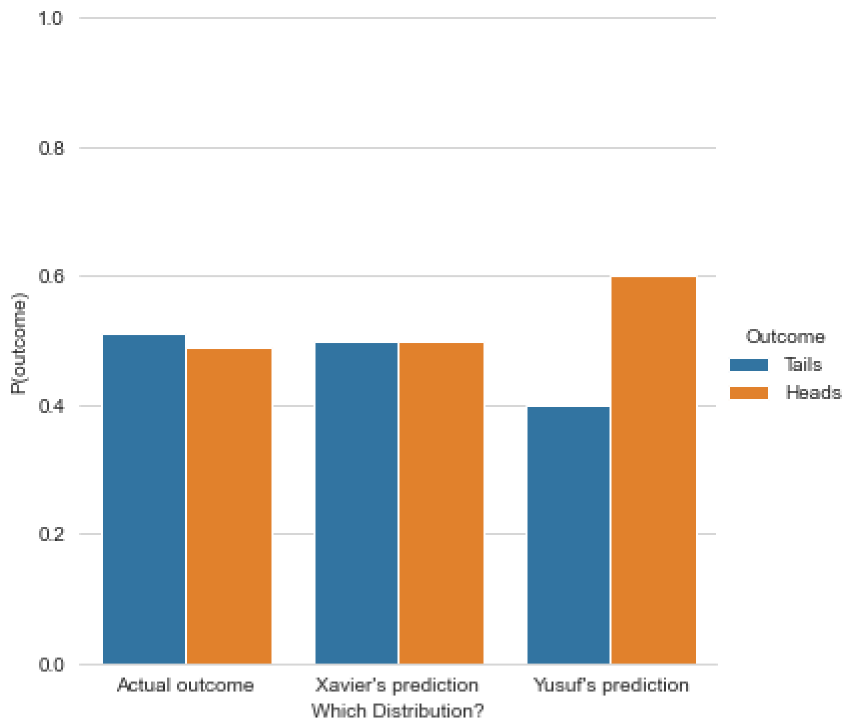
Out[72]:

	index	outcome	p	which
0	0	Tails	0.51	Actual outcome
1	1	Heads	0.49	Actual outcome
2	0	Tails	0.50	Xavier's prediction
3	1	Heads	0.50	Xavier's prediction
4	0	Tails	0.40	Yusuf's prediction
5	1	Heads	0.60	Yusuf's prediction

In [73]:
```
plot_distributions_a(dist_df2)
```



In [74]:
```
plot_distributions_b(dist_df2)
```



So, we need to **formalize** how to measure **how bad** a prediction is.

Enter... statistics!

Null vs. Alternative Hypotheses

- **Null Hypothesis** (H_0): The skeptical hypothesis... "Nothing interesting is going on here, any patterns were simply due to chance"
    - The coin is not weird. $P(\text{heads}) = 0.5$
- **Alternative Hypothesis** (H_A): Something *other than* chance is generating the pattern we observe
    - The coin is loaded! $P(\text{heads}) \neq 0.5$

ONLY TWO POSSIBLE CONCLUSIONS FROM YOUR HYPOTHESIS TEST

1. "We **reject** the null hypothesis"
2. "We **fail to reject** the null hypothesis"

Test Statistic

- Computed from the **observed** data
- A measure of how reasonable our alternative hypothesis is for explaining this data

(This is the number we were looking for before!)

So, how bad were our coin flip predictions?

In [91]:

```
def test_stat_a(coin_flips):
    # Num heads - num tails
    num_heads = len([f for f in coin_flips if f == 1])
    num_tails = len([f for f in coin_flips if f == 0])
    return num_heads - num_tails
```

In [92]:

```
print(test_stat_a(do_coin_flips(100)))
print(test_stat_a(do_coin_flips(100)))
```

```
-2
4
```

What would this test statistic look like if the coin was *actually* fair?

In [93]:
```
def fair_coin_flips(N):
    return np.array([np.random.binomial(1, 0.5) for i in range(N)])
```

In [94]:
```
fair_coin_flips(100)
```

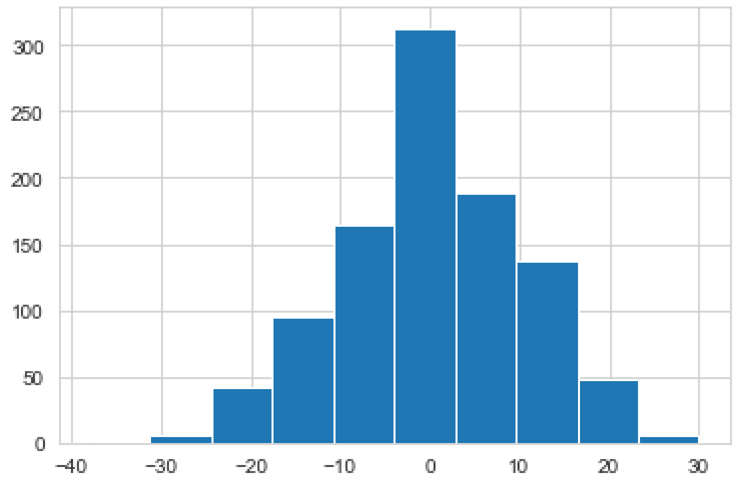
Out[94]:
```
array([0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1,
       1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0,
       0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1,
       0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1])
```

In [95]:

```
test_stats = []
for trial_num in range(1000):
    test_stats.append(test_stat_a(fair_coin_flips(100)))
```

In [96]:

```
plt.hist(test_stats)
plt.show()
```



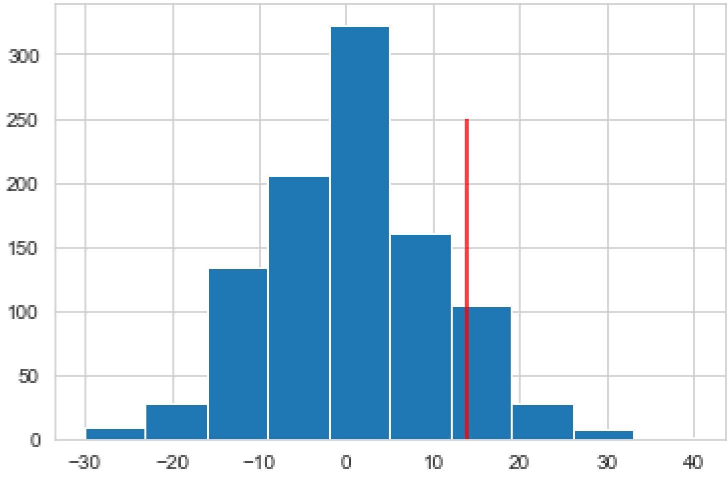
So... now we do 100 flips of our secret coin and see where it falls on this distribution!

In [114]:
```
secret_coin_results = do_coin_flips(100)
```

In [115]:
```
sc_stat = test_stat_a(secret_coin_results)
sc_stat
```

Out[115]:
```
14
```

In [116]:
```
plt.hist(test_stats)
plt.vlines(sc_stat, 0, 250, color='red')
plt.show()
```



Note: This is **not** the only possible test statistic!

- We *make up* the test statistic that can best help us "detect" not-by-chance data

In [117]:

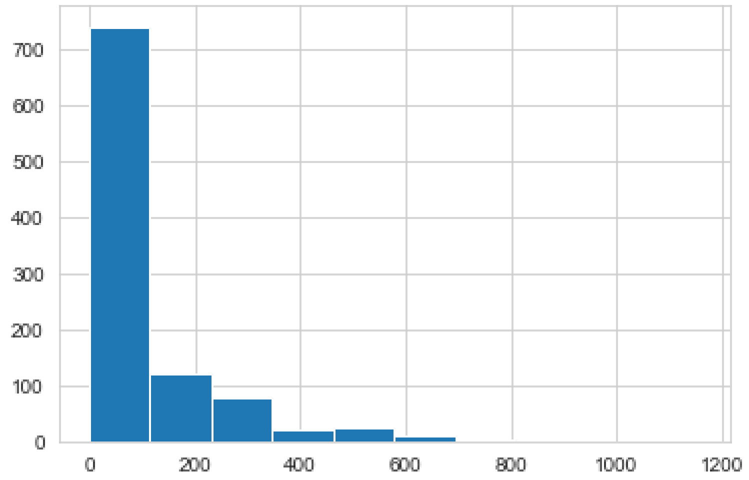
```
def test_stat_b(coin_flips):
    # The squared difference in predicted probabilities
    num_heads = len([f for f in coin_flips if f == 1])
    num_tails = len([f for f in coin_flips if f == 0])
    return (num_heads - num_tails) ** 2
```

In [118]:

```
test_stats_b = []
for trial_num in range(1000):
    test_stats_b.append(test_stat_b(fair_coin_flips(100)))
```

In [119]:

```
plt.hist(test_stats_b)
plt.show()
```

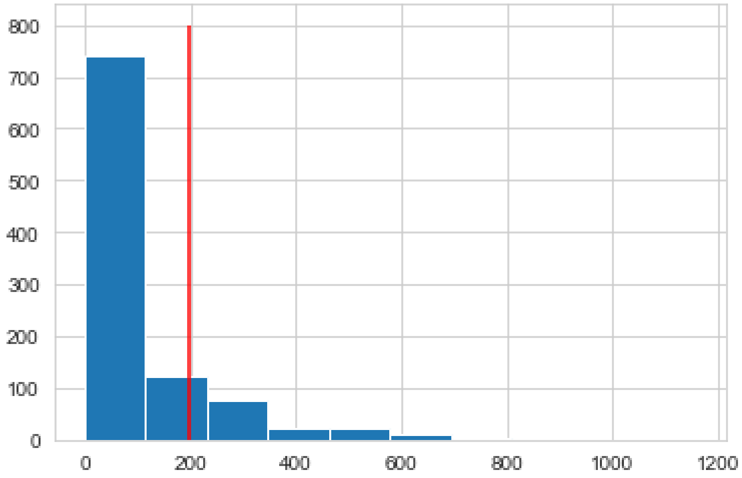


Now let's place our **observed** data on this plot

In [137]:
```
sc_stat_b = test_stat_b(secret_coin_results)
sc_stat_b
```

Out[137]: 196

In [139]:
```
plt.hist(test_stats_b)
plt.vlines(sc_stat_b, 0, 800, color='red')
plt.show()
```



In [141]:

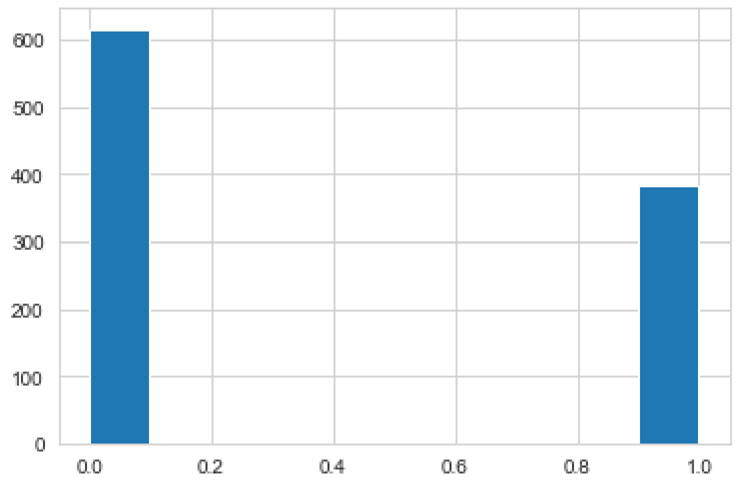
```python
def test_stat_c(coin_flips):
    # 1 if it's within 5, 0 otherwise
    num_heads = len([f for f in coin_flips if f == 1])
    num_tails = len([f for f in coin_flips if f == 0])
    diff = abs(num_heads - num_tails)
    return 1 if diff <= 5 else 0
```

In [128]:

```python
test_stats_c = []
for trial_num in range(1000):
    test_stats_c.append(test_stat_c(fair_coin_flips(100)))
```

In [129]:

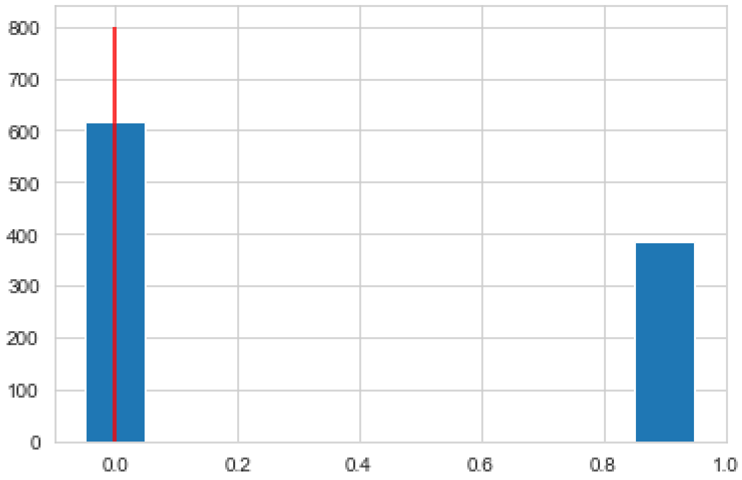
```python
plt.hist(test_stats_c)
plt.show()
```



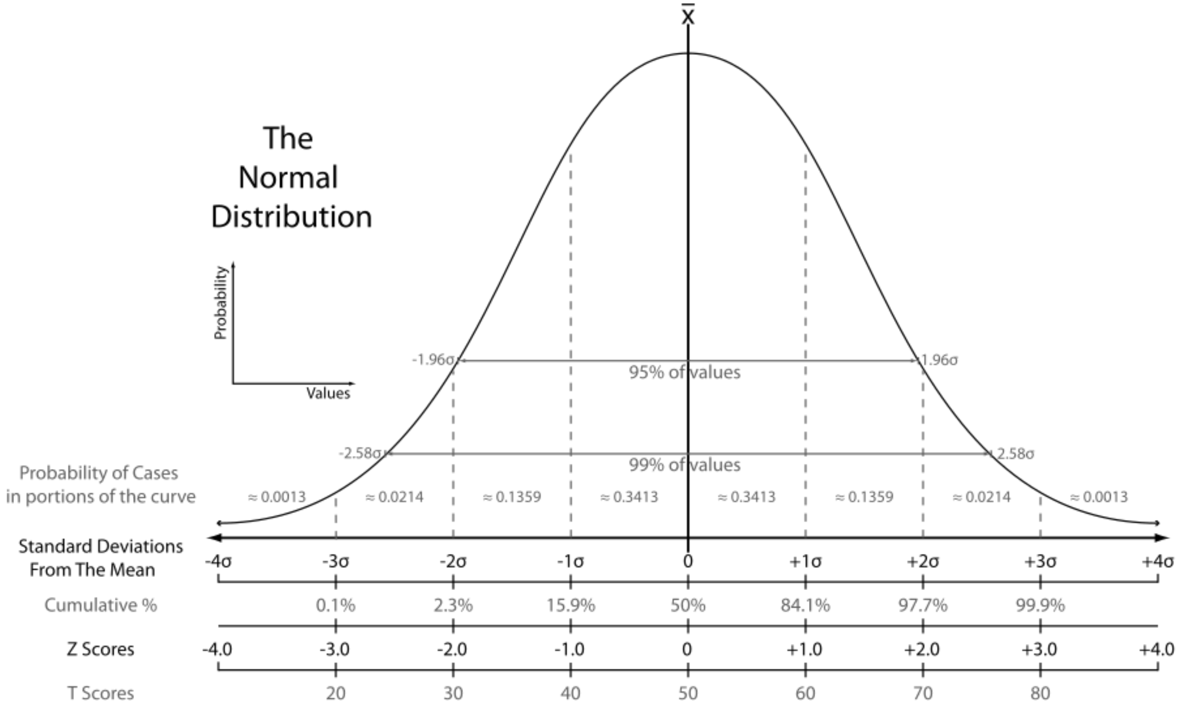
In [130]:
```
sc_stat_c = test_stat_c(secret_coin_results)
sc_stat_c
```

Out[130]: 0

In [136]:
```
plt.hist(test_stats_c, align='left')
plt.vlines(sc_stat_c, 0, 800, color='red')
plt.show()
```



The Normal Distribution



Correlation