11.2) Causality

Vitor Kamada

December 2019

Reference

Tables, Graphics, and Figures from

Computational and Inferential Thinking: The Foundations of Data Science

Adhikari & DeNero (2019): Ch 12.3 Causality

https://www.inferentialthinking.com/

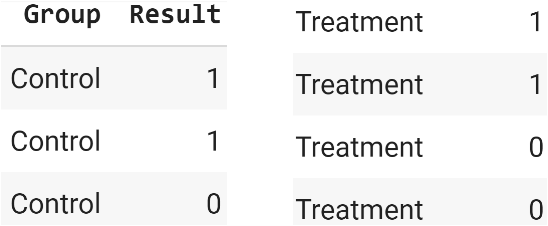
Treating Chronic Back Pain

Foster, Clapp, and Jabbari (2001)

Randomized Controlled Trial (RCT)

Effect of Botulinum Toxin A

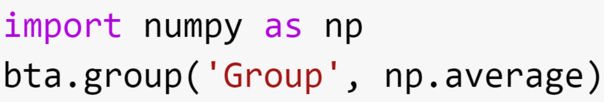
```
from datascience import *
path_data = 'https://github.com/data-8/textbook/raw/gh-pages/data/'
bta = Table.read_table(path_data + 'bta.csv')
```



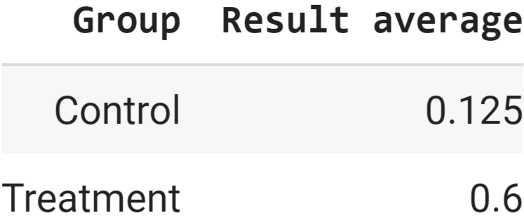
Group	Result
Control	1
Control	1
Control	0
Treatment	1
Treatment	1
Treatment	0
Treatment	0

Proportion of Patients who had Pain Relief

$$H_0 : p_t = p_c \text{ vs } H_A : p_t \neq p_c$$



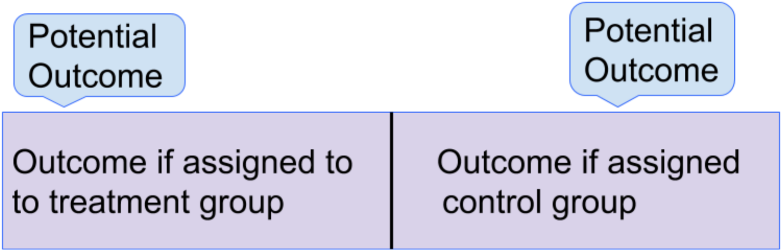
```python
import numpy as np
bta.group('Group', np.average)
```



Group	Result average
Control	0.125
Treatment	0.6

Before Randomization

Imaginary ticket for each of the 31 participants



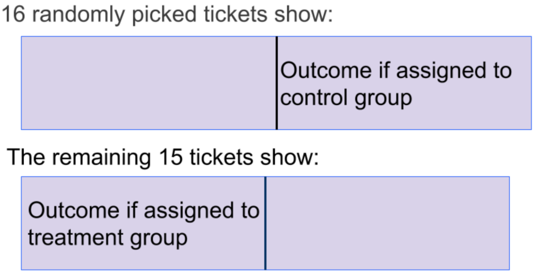
Potential Outcome

Potential Outcome

Outcome if assigned to to treatment group

Outcome if assigned control group

Potential Outcomes

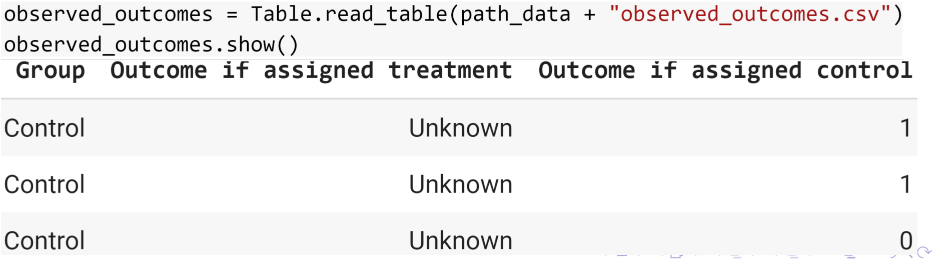


16 randomly picked tickets show:

Outcome if assigned to control group

The remaining 15 tickets show:

Outcome if assigned to treatment group

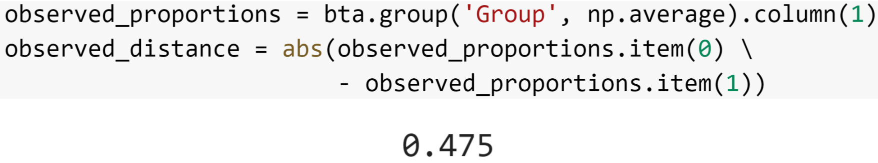


```python
observed_outcomes = Table.read_table(path_data + "observed_outcomes.csv")
observed_outcomes.show()
```

Group	Outcome if assigned treatment	Outcome if assigned control
Control	Unknown	1
Control	Unknown	1
Control	Unknown	0

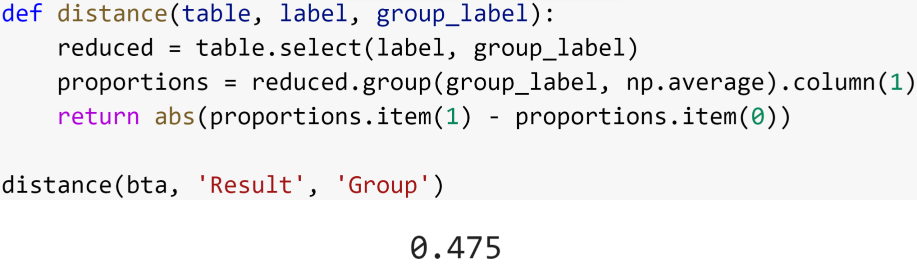
Test Statistic

$$\left|0.6 - 0.125\right| = 0.475$$



```
observed_proportions = bta.group('Group', np.average).column(1)
observed_distance = abs(observed_proportions.item(0) \
                        - observed_proportions.item(1))
```

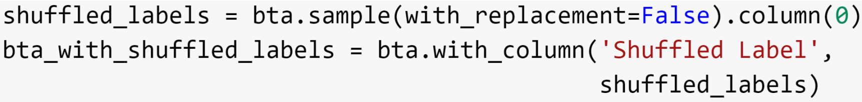
```
0.475
```



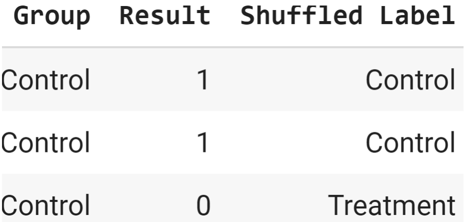
```
def distance(table, label, group_label):
    reduced = table.select(label, group_label)
    proportions = reduced.group(group_label, np.average).column(1)
    return abs(proportions.item(1) - proportions.item(0))

distance(bta, 'Result', 'Group')
```

```
0.475
```

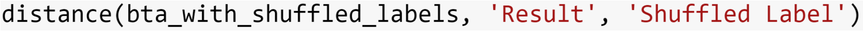
Shuffle Labels



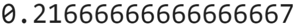
```
shuffled_labels = bta.sample(with_replacement=False).column(0)
bta_with_shuffled_labels = bta.with_column('Shuffled Label',
                                           shuffled_labels)
```



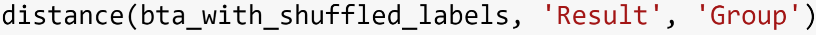
Group	Result	Shuffled Label
Control	1	Control
Control	1	Control
Control	0	Treatment



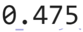
```
distance(bta_with_shuffled_labels, 'Result', 'Shuffled Label')
```



```
0.21666666666666667
```

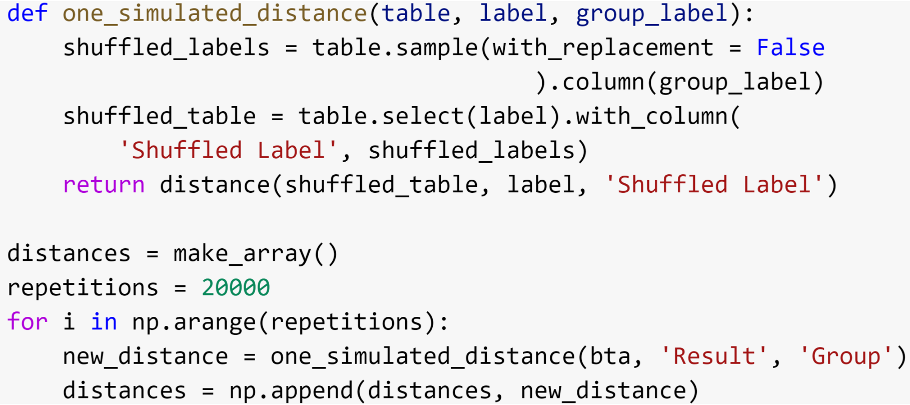


```
distance(bta_with_shuffled_labels, 'Result', 'Group')
```

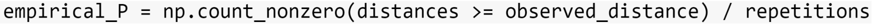


```
0.475
```

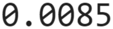
Permutation Test



```python
def one_simulated_distance(table, label, group_label):
    shuffled_labels = table.sample(with_replacement = False
                                        ).column(group_label)
    shuffled_table = table.select(label).with_column(
        'Shuffled Label', shuffled_labels)
    return distance(shuffled_table, label, 'Shuffled Label')

distances = make_array()
repetitions = 20000
for i in np.arange(repetitions):
    new_distance = one_simulated_distance(bta, 'Result', 'Group')
    distances = np.append(distances, new_distance)
```



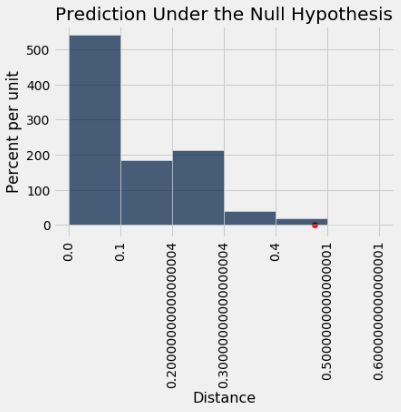
```python
empirical_P = np.count_nonzero(distances >= observed_distance) / repetitions
```



```
0.0085
```

```
Table().with_column('Distance', distances).hist(bins = np.arange(0, 0.7, 0.1))
plots.scatter(observed_distance, 0, color='red', s=40)
plots.title('Prediction Under the Null Hypothesis')
print('Observed Distance', observed_distance)
print('Empirical P-value:', round(empirical_P, 4) *100, '%')
```

```
Observed Distance 0.475
Empirical P-value: 0.8500000000000001 %
```


Prediction Under the Null Hypothesis
Percent per unit
Distance