Data Club Fall 24 Pumpkin Bowling Project

January 4, 2025

[33]:
```
import numpy as np
from datascience import *

import matplotlib
%matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
import warnings
warnings.simplefilter('ignore', FutureWarning)

from ipywidgets import interact, interactive, fixed, interact_manual
import ipywidgets as widgets
```

[34]:
```
PumpkinBowling = Table().read_table("PumpkinBowlingData.csv")
PumpkinBowling
```

[34]:
```
Ghost | Frankenstein | Pumpkin
5     | 3            | 1
5     | 2            | 1
5     | 3            | 1
6     | 2            | 1
4     | 3            | 0
2     | 2            | 0
4     | 3            | 1
4     | 3            | 1
4     | 2            | 1
3     | 3            | 1
… (90 rows omitted)
```

[2]:
```
#Null Hypothesis: Players will not target the pumpkin pin, but aim to just␣
 ↪knock down as many pins as they can.

#Alternative Hypothesis: Players will aim their throws specifically at the␣
 ↪pumpkin pin over the other 2 types of pins because it guarantees a prize if␣
 ↪knocked.
```

```
#Test Statistic: The proportion of pumpkin pins knocked down after 100 trials/␣
 ↪Proportion of pumpkin pins knocked out of
#705 total pins knocked
```

[36]:
```
def sumUpPins (Ghost, Frank, Pumpkin):
    '''Return the number of pins that were knocked down for every trial'''
    totalPins = Ghost + Frank + Pumpkin
    return totalPins
```

[37]:
```
totalPins = PumpkinBowling.apply(sumUpPins, "Ghost", "Frankenstein", "Pumpkin")
```

[38]:
```
PumpkinBowling = PumpkinBowling.with_column("Total Pins Knocked", totalPins)
PumpkinBowling
```

[38]:
```
Ghost | Frankenstein | Pumpkin | Total Pins Knocked
5     | 3            | 1       | 9
5     | 2            | 1       | 8
5     | 3            | 1       | 9
6     | 2            | 1       | 9
4     | 3            | 0       | 7
2     | 2            | 0       | 4
4     | 3            | 1       | 8
4     | 3            | 1       | 8
4     | 2            | 1       | 7
3     | 3            | 1       | 7
… (90 rows omitted)
```

[39]:
```
#Total pins knocked after 100 trials
pinsAltogether = sum(PumpkinBowling.column("Total Pins Knocked"))
#Observed Test Statistic
pumpkinProportion = sum(PumpkinBowling.column("Pumpkin"))/pinsAltogether
pumpkinProportion
```

[39]:
```
0.097872340425531917
```

1 Simulating Method 1 (Inaccurate)

[41]:
```
pinsKnocked = make_array()

for i in np.arange(10):
    shuffledLabels = PumpkinBowlingSimulation.sample(with_replacement = True).
 ↪column("Pins")
    arrayTable = PumpkinBowlingSimulation.with_column("Shuffled",␣
 ↪shuffledLabels)
    arrayValue = arrayTable.where("Knocked", are.equal_to(True)).column(1)
    pinsKnocked = np.append(pinsKnocked, arrayValue)
```

```
arrayTable
```

```
[41]: Knocked | Pins         | Shuffled
      True    | Pumpkin      | Pumpkin
      False   | Ghost        | Frankenstein
      False   | Ghost        | Pumpkin
      False   | Ghost        | Ghost
      False   | Ghost        | Frankenstein
      False   | Ghost        | Ghost
      False   | Ghost        | Ghost
      False   | Frankenstein | Ghost
      False   | Frankenstein | Ghost
      False   | Frankenstein | Pumpkin
```

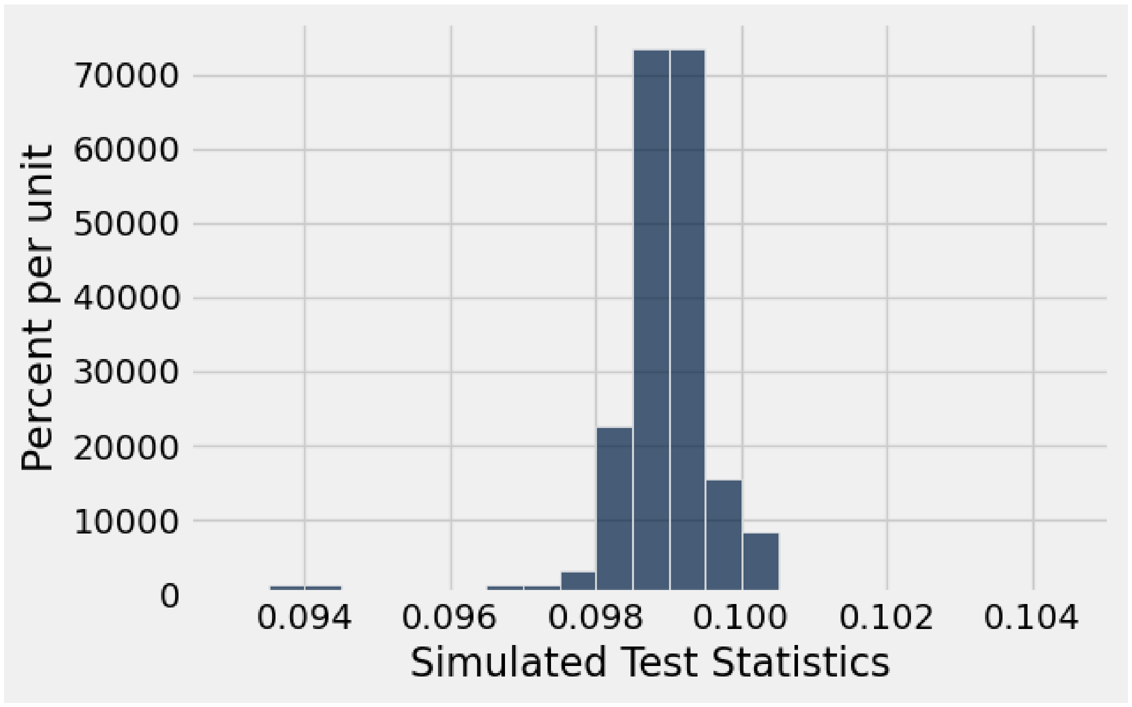
```
[42]: #200 trials of simulating 100 rounds of rolling the pumpkin.
      #This cell produces an array of 100 simulated test statistics which was␣
       ↪acquired under null hypothesis.
      appendedArrayFinal = make_array()
      appendedArray = make_array()

      for j in np.arange(200):
          for i in np.arange(100):
              pinsArray = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
              numberKnocked = np.random.choice(pinsArray)
              individualPins = ("Pumpkin", "Ghost", "Ghost", "Ghost", "Ghost",␣
       ↪"Ghost", "Ghost", "Frankenstein", "Frankenstein", "Frankenstein")
              oneSimulatedTrial = np.random.choice(individualPins, numberKnocked)
              appendedArray = np.append(appendedArray, oneSimulatedTrial)
          myTable = Table().with_column("Simulated pins knocked", appendedArray)
          simulatedTestStatistic = (myTable.where("Simulated pins knocked", are.
       ↪equal_to("Pumpkin"))).num_rows/myTable.num_rows
          appendedArrayFinal = np.append(appendedArrayFinal, simulatedTestStatistic)

      appendedArrayFinal
```

```
[42]: array([ 0.09703947,  0.08547718,  0.08747856,  0.09372313,  0.09075044,
              0.08952604,  0.09406694,  0.096767  ,  0.09780876,  0.09940573,
              0.10006572,  0.10025362,  0.09788868,  0.09868087,  0.09817189,
              0.09794734,  0.09877062,  0.09994063,  0.09960608,  0.09972324,
              0.09914821,  0.09845576,  0.09840675,  0.09850075,  0.09869887,
              0.098716  ,  0.0989011 ,  0.09865931,  0.09900683,  0.09901761,
              0.09894835,  0.09942014,  0.09930666,  0.10010084,  0.09993283,
              0.10032167,  0.10002932,  0.10018562,  0.09987018,  0.09985522,
              0.09974357,  0.09985328,  0.09980636,  0.09997532,  0.09987113,
              0.10010641,  0.10005016,  0.09967166,  0.09969688,  0.0992289 ,
              0.09891281,  0.09887202,  0.09902071,  0.09890551,  0.09870906,
              0.0985425 ,  0.09848772,  0.09864886,  0.09864828,  0.09855099,
              0.09861107,  0.09909413,  0.09913645,  0.09910239,  0.09887248,
              0.09889391,  0.09887689,  0.09897502,  0.09899338,  0.09901676,
              0.09881924,  0.09891071,  0.09893575,  0.0989347 ,  0.09910039,
              0.09913417,  0.09910078,  0.09896348,  0.09916474,  0.0990583 ,
              0.09897195,  0.0990021 ,  0.09894413,  0.09906386,  0.09906487,
              0.09876621,  0.09882581,  0.09876971,  0.0986728 ,  0.0986379 ,
              0.09876762,  0.09876904,  0.09876209,  0.09856703,  0.09849431,
              0.09859155,  0.09846701,  0.09833058,  0.09835798,  0.09865028,
              0.09856398,  0.09880873,  0.09883355,  0.09900066,  0.09902567,
              0.09894533,  0.09911198,  0.09906636,  0.09934289,  0.09933244,
              0.09941634,  0.0995148 ,  0.09941502,  0.09934308,  0.0994023 ,
              0.09950797,  0.09928892,  0.09930906,  0.0994644 ,  0.0993648 ,
              0.09922573,  0.09906426,  0.09908168,  0.09903192,  0.09902061,
              0.09894372,  0.09889766,  0.09889103,  0.09878184,  0.09868832,
              0.09876679,  0.09870094,  0.09868064,  0.09867278,  0.09870251,
              0.09858463,  0.09859378,  0.09843415,  0.09828916,  0.09829831,
              0.0984131 ,  0.0983417 ,  0.09828763,  0.09830045,  0.09842103,
              0.09837322,  0.09841601,  0.09852367,  0.09854616,  0.09848695,
              0.09850007,  0.0985025 ,  0.09836201,  0.09849142,  0.09852607,
              0.09849081,  0.09851038,  0.09857479,  0.09850834,  0.09860109,
              0.09863816,  0.09875236,  0.09880784,  0.09881423,  0.09889088,
              0.09896037,  0.09902789,  0.09908186,  0.09899741,  0.09907259,
              0.09912536,  0.09917442,  0.09916587,  0.09928364,  0.09927036,
              0.09918484,  0.09924568,  0.09921284,  0.09922438,  0.09923005,
              0.09933127,  0.0994597 ,  0.09942133,  0.09940682,  0.09943704,
              0.09939071,  0.0994481 ,  0.09941729,  0.09938975,  0.09948047,
              0.09934295,  0.09925397,  0.0992624 ,  0.09927934,  0.09920974,
              0.09912047,  0.09910745,  0.09912198,  0.09914898,  0.0991361 ])
```

[43]:
```
bins = np.arange(0.093, 0.105, 0.0005)
myTable = Table().with_column("Simulated Test Statistics", appendedArrayFinal)
myTable.hist(bins = bins)
```



```
[44]: pValue = np.count_nonzero(appendedArrayFinal <= pumpkinProportion)/myTable.
        ↪num_rows
      pValue
```

```
[44]: 0.045
```

```
[51]: #Simulating Method 1 is inaccurate because the distribution of the simulated␣
        ↪test statistics is not symmetric and centered about
      #0.1 which is the expected proportion of pumpkin pins that would be knocked␣
        ↪after 100 rounds (under Null Hypothesis)
```

2 Simulating Method 2 (Accurate)

```
[50]: #Number of observed pins knocked down (from the actual game)
      pinsAltogether
```

```
[50]: 705
```

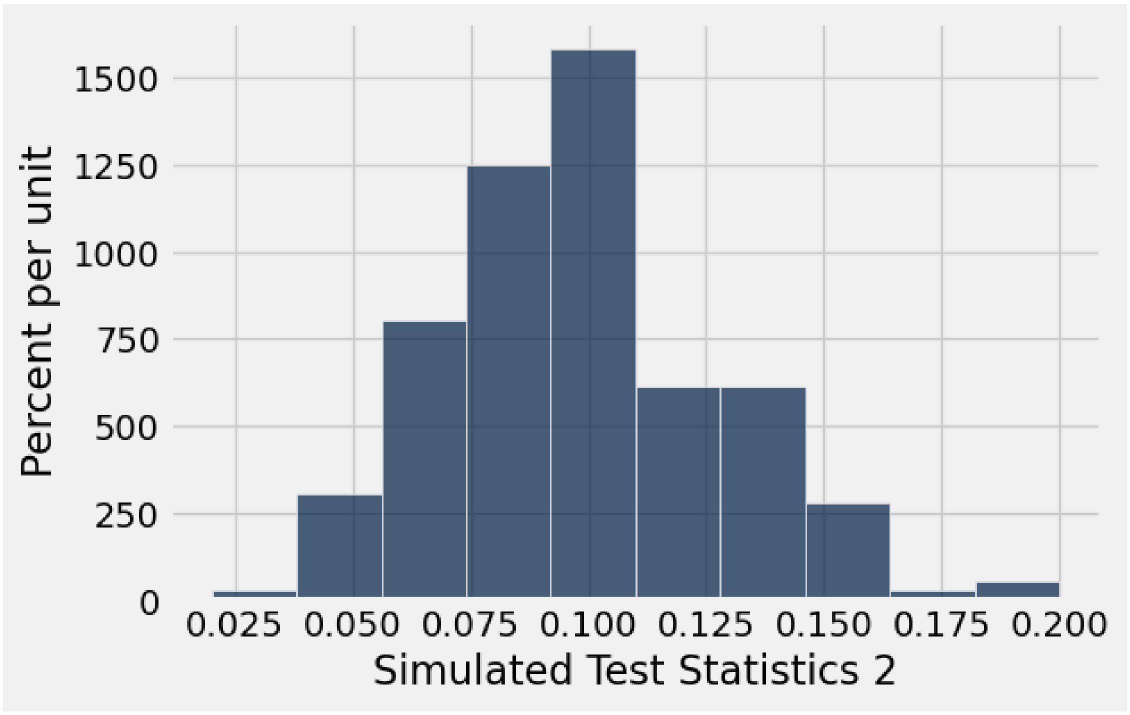
```
[46]: sample_proportions(705, (0.6, 0.3, 0.1))
```

```
[46]: array([ 0.64680851,  0.2751773 ,  0.07801418])
```

```
[47]: modelProportion = make_array(0.6, 0.3, 0.1)
numRepetitions = 200
def simulatePumpkinKnocks(modelProportion):
    '''simulate pumpkin pin knocks proportion in 100 trials for numRepetitions␣
  ↪times'''
    simulatedProportionPerThrow = sample_proportions(100, modelProportion)
    return simulatedProportionPerThrow.item(2)

appendedArrayFinal = make_array()
for i in np.arange(numRepetitions):
    oneSim = simulatePumpkinKnocks(modelProportion)
    appendedArrayFinal = np.append(appendedArrayFinal, oneSim)

myTable = Table().with_column("Simulated Test Statistics 2", appendedArrayFinal)
myTable.hist()
```



```
[48]: pValue = np.count_nonzero(appendedArrayFinal <= pumpkinProportion)/myTable.
  ↪num_rows
pValue
```

```
[48]: 0.43
```

3 Conclusion

Conclusion: Using method 2, we retrieved an accurate simulated distribution (histogram above) of the proportions of pumpkin pins knocked from 100 rounds. As expected, it is symmetric about the 0.1 mark because under the notion that the null hypothesis is correct (Players will not aim specifically for the pumpkin pin), the estimated test statistic should be around 0.1 or 1/10 since only 1 out of the 10 pins were a pumpkin pin. However, according to pumpkinProportion (observed test statistic), 9.79 percent of the 705 pins knocked were pumpkin pins. This means that the observed test statistic is less than expected (0.1 pr 10 percent). This supports the claim that players were approaching the strategy of simply knocking down as many pins as possible instead of specifically aiming for the pumpkin pin. Thus, they were purposely avoding the pumpkin pin and aiming for the general crowd of the pins. Perhaps, players felt that the pumpkin pin was placed at a non-ideal spot that would decrease their overall chance of winning a prize. The pValue of 0.43 of 43 percent was nowhere less than any of the standard P-cutoff values (like 0.1, 1, 5, or 10). Thus, we can accept the null hypothesis which states that "Players will not target the pumpkin pin, but aim to just knock down as many pins as they can."