lec20and21

July 18, 2022

[1]:
```
from datascience import *
import numpy as np

%matplotlib inline
import matplotlib.pyplot as plots
plots.style.use('fivethirtyeight')
```

0.1 Birth Weights

[2]:
```
baby = Table.read_table('baby.csv')
baby
```

[2]:
```
Birth Weight | Gestational Days | Maternal Age | Maternal Height | Maternal
Pregnancy Weight | Maternal Smoker
120          | 284              | 27           | 62              | 100
| False
113          | 282              | 33           | 64              | 135
| False
128          | 279              | 28           | 64              | 115
| True
108          | 282              | 23           | 67              | 125
| True
136          | 286              | 25           | 62              | 93
| False
138          | 244              | 33           | 62              | 178
| False
132          | 245              | 23           | 65              | 140
| False
120          | 289              | 25           | 62              | 125
| False
143          | 299              | 30           | 66              | 136
| True
140          | 351              | 27           | 68              | 120
| False
… (1164 rows omitted)
```

[3]:
```
smoking_and_birthweight = baby.select('Birth Weight', 'Maternal Smoker')
smoking_and_birthweight
```

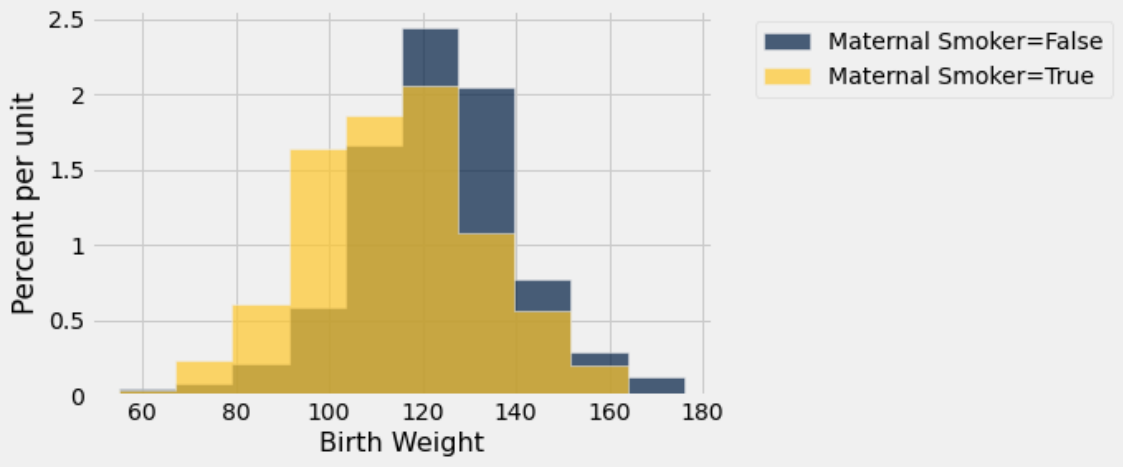
[3]:
```
Birth Weight | Maternal Smoker
120          | False
113          | False
128          | True
108          | True
136          | False
138          | False
132          | False
120          | False
143          | True
140          | False
… (1164 rows omitted)
```

[4]:
```
smoking_and_birthweight.group('Maternal Smoker')
```

[4]:
```
Maternal Smoker | count
False           | 715
True            | 459
```

[5]:
```
smoking_and_birthweight.hist('Birth Weight', group='Maternal Smoker')
```

```
C:\Users\schoend\Anaconda3\lib\site-packages\datascience\tables.py:920:
VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences
(which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths
or shapes) is deprecated. If you meant to do this, you must specify
'dtype=object' when creating the ndarray.
  values = np.array(tuple(values))
```



[6]:
```
means_table = smoking_and_birthweight.group('Maternal Smoker', np.average)
means_table
```

[6]:
```
Maternal Smoker | Birth Weight average
False           | 123.085
True            | 113.819
```

[7]:
```
def diff_between_group_means(tbl):
    means = tbl.group('Maternal Smoker', np.average)
    return means.column(1).item(0) - means.column(1).item(1)
```

[8]:
```
observed_diff = diff_between_group_means(smoking_and_birthweight)
observed_diff
```

[8]:
```
9.266142572024918
```

[9]:
```
# PLAN:
# Shuffle birth weights
# Assign some to group A and some to group B
# Find difference between averages of the two groups (statistic)
# Repeat
```

[10]:
```
weights = smoking_and_birthweight.select('Birth Weight')
weights
```

[10]:
```
Birth Weight
120
113
128
108
136
138
132
120
143
140
… (1164 rows omitted)
```

[11]:
```
smoking = smoking_and_birthweight.select('Maternal Smoker')
smoking
```

[11]:
```
Maternal Smoker
False
False
True
True
False
```

```
False
False
False
True
False
… (1164 rows omitted)
```

[12]:
```
# Shuffle birth weights
weights = smoking_and_birthweight.select('Birth Weight')
```

[13]:
```
# Shuffle birth weights
shuffled_weights = weights.sample(with_replacement=False).column(0)
shuffled_weights
```

[13]:
```
array([112,  96, 129, …, 119, 114,  95])
```

[14]:
```
# Assign some to group A and some to group B
simulated = smoking.with_column('Shuffled weights', shuffled_weights)
simulated
```

[14]:
```
Maternal Smoker | Shuffled weights
False           | 112
False           | 96
True            | 129
True            | 160
False           | 91
False           | 100
False           | 92
False           | 119
True            | 107
False           | 131
… (1164 rows omitted)
```

[15]:
```
# Find difference between averages of the two groups (statistic)
simulated_diff = diff_between_group_means(simulated)
simulated_diff
```

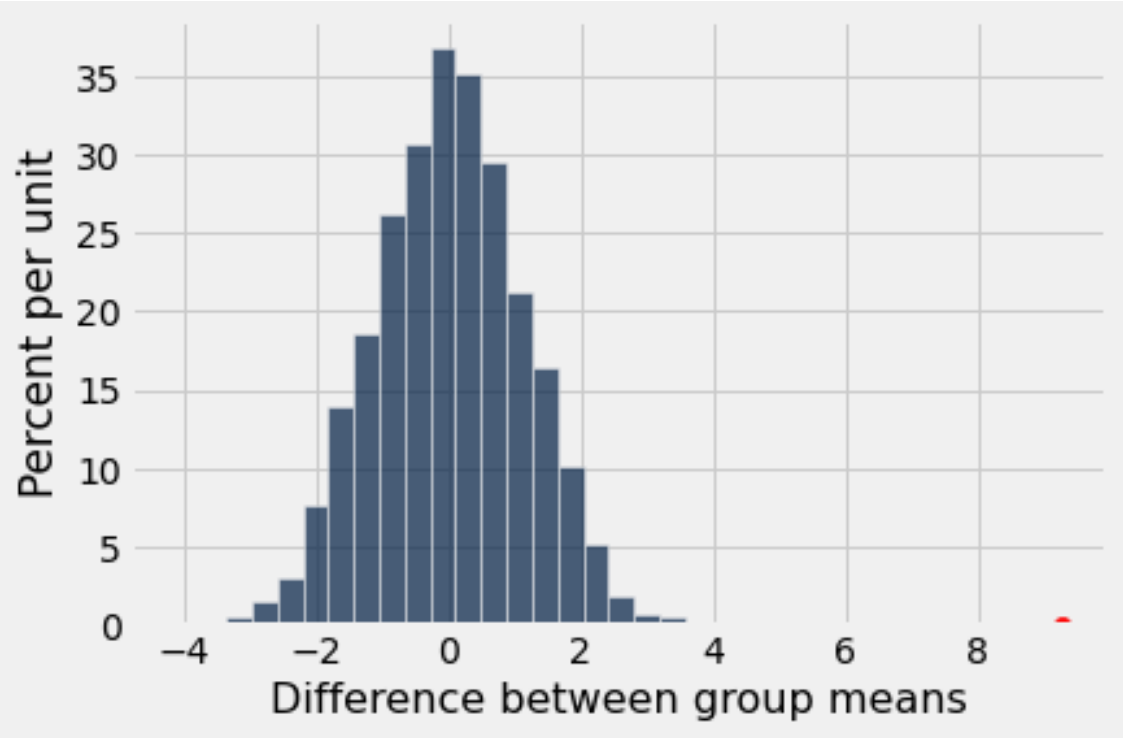
[15]:
```
0.6735865441747819
```

[16]:
```
# Repeat
diffs = make_array()
for i in np.arange(2000):
    shuffled_weights = weights.sample(with_replacement=False).column(0)
    simulated = smoking.with_column('Shuffled weights', shuffled_weights)
    diff = diff_between_group_means(simulated)
    diffs = np.append(diffs, diff)
```

```
diffs
```

[16]:
```
array([-0.1348721 ,  0.423179  ,  0.49830126, …, -1.38333257,
        0.86318083,  0.30512973])
```

[17]:
```
Table().with_column('Difference between group means', diffs).hist(bins=20)
plots.scatter(observed_diff, 0, color = 'red', s = 40);
```



0.2 Deflategate

[18]:
```
football = Table.read_table('deflategate.csv')
football.show()
```

```
<IPython.core.display.HTML object>
```

[19]:
```
combined = (football.column('Blakeman')+football.column('Prioleau'))/2
football = football.drop('Blakeman', 'Prioleau').with_column(
    'Combined',
    combined)
football.show()
```

```
<IPython.core.display.HTML object>
```

[20]:
```
np.ones(5)
```

[20]:
```
array([1., 1., 1., 1., 1.])
```

[21]:
```
initial_pressure = np.append(12.5 * np.ones(11), 13 * np.ones(4))
initial_pressure
```

[21]:
```
array([12.5, 12.5, 12.5, 12.5, 12.5, 12.5, 12.5, 12.5, 12.5, 12.5, 12.5,
       13. , 13. , 13. , 13. ])
```

[22]:
```
drop_values = initial_pressure - football.column(1)
```

[23]:
```
football = football.drop('Combined').with_column('Drop', drop_values)
```

[24]:
```
football.show()
```

```
<IPython.core.display.HTML object>
```

[25]:
```
means = football.group('Team', np.average)
means
```

[25]:
```
Team     | Drop average
Colts    | 0.46875
Patriots | 1.20227
```

[26]:
```
observed_difference = means.column(1).item(0) - means.column(1).item(1)
observed_difference
```

[26]:
```
-0.733522727272728
```

[27]:
```
def diff_between_means(tbl):
    means = tbl.group('Team', np.average).column(1)
    return means.item(0) - means.item(1)
```

[28]:
```
drops = football.select('Drop')
```

[29]:
```
shuffled_drops = drops.sample(with_replacement = False).column(0)
shuffled_drops
```

[29]:
```
array([0.425, 0.85 , 0.475, 0.65 , 1.8  , 1.65 , 0.275, 1.475, 1.375,
       0.475, 1.175, 0.725, 1.35 , 1.225, 1.175])
```

[30]:
```
simulated_football = football.with_column('Drop', shuffled_drops)
simulated_football.show(3)
```

```
<IPython.core.display.HTML object>
```

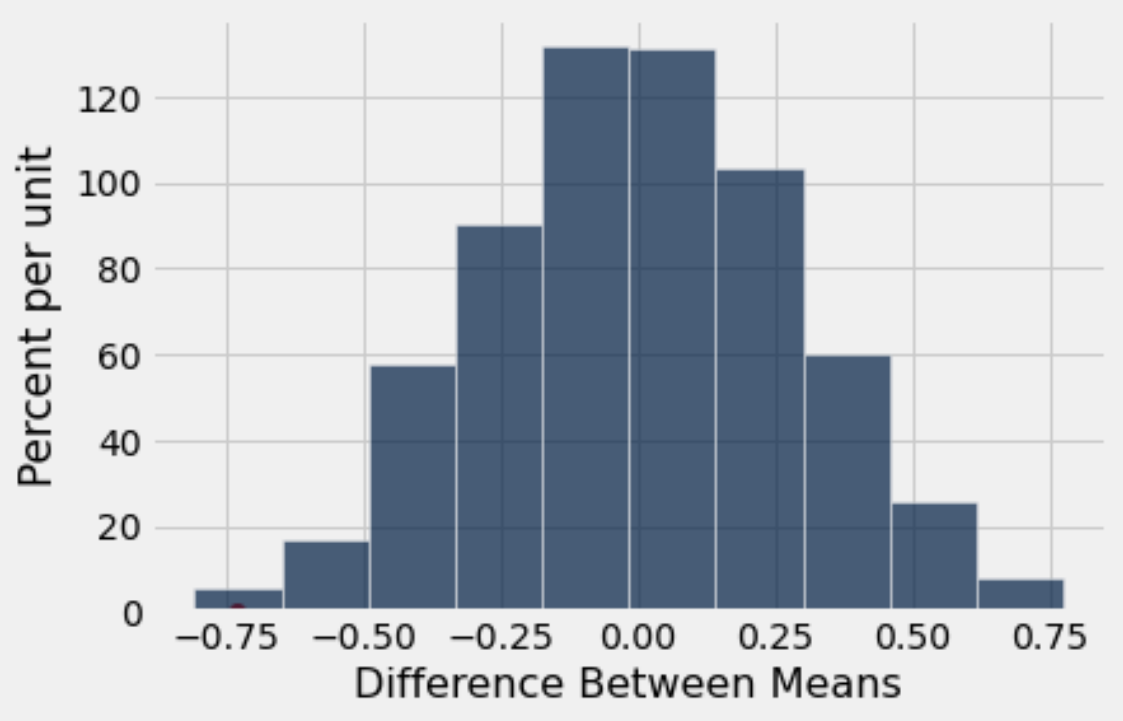
[31]:
```
diff_between_means(simulated_football)
```

[31]:
```
0.15284090909090997
```

[32]:
```
differences = make_array()

for i in np.arange(5000):
    shuffled_drops = drops.sample(with_replacement = False).column(0)
    simulated_football = football.with_column('Drop', shuffled_drops)
    new_diff = diff_between_means(simulated_football)
    differences = np.append(differences, new_diff)
```

[33]:
```
Table().with_column('Difference Between Means', differences).hist()
plots.scatter(observed_difference, 0, color='red', s=40);
```



[34]:
```
np.average(differences <= observed_difference)
```

[34]:
```
0.0034
```

Analyzing RCTs

[35]:
```
#See Inferential Thinking textbook Section 12.3
```

[36]:
```
bta = Table.read_table('bta.csv')
bta.show()
```

```
<IPython.core.display.HTML object>
```

[37]:
```
bta = Table.read_table('bta.csv')
bta.show()
```

```
<IPython.core.display.HTML object>
```

[38]:
```
bta.group('Group', sum)
```

[38]:
```
Group     | Result sum
Control   | 2
Treatment | 9
```

[39]:
```
bta.group('Group', np.average)
```

[39]:
```
Group     | Result average
Control   | 0.125
Treatment | 0.6
```

[40]:
```
observed_outcomes = Table.read_table('observed_outcomes.csv')
observed_outcomes.show()
```

```
<IPython.core.display.HTML object>
```

[41]:
```
bta.group('Group', np.average).column(1)
```

[41]:
```
array([0.125, 0.6  ])
```

[42]:
```
abs(0.125 - 0.6)
```

[42]:
```
0.475
```

[43]:
```
def distance_between_group_proportions(tbl):
    proportions = tbl.group('Group', np.average).column(1)
    return abs(proportions.item(1) - proportions.item(0))
```

[44]:
```
observed_distance = distance_between_group_proportions(bta)
observed_distance
```

[44]:
```
0.475
```

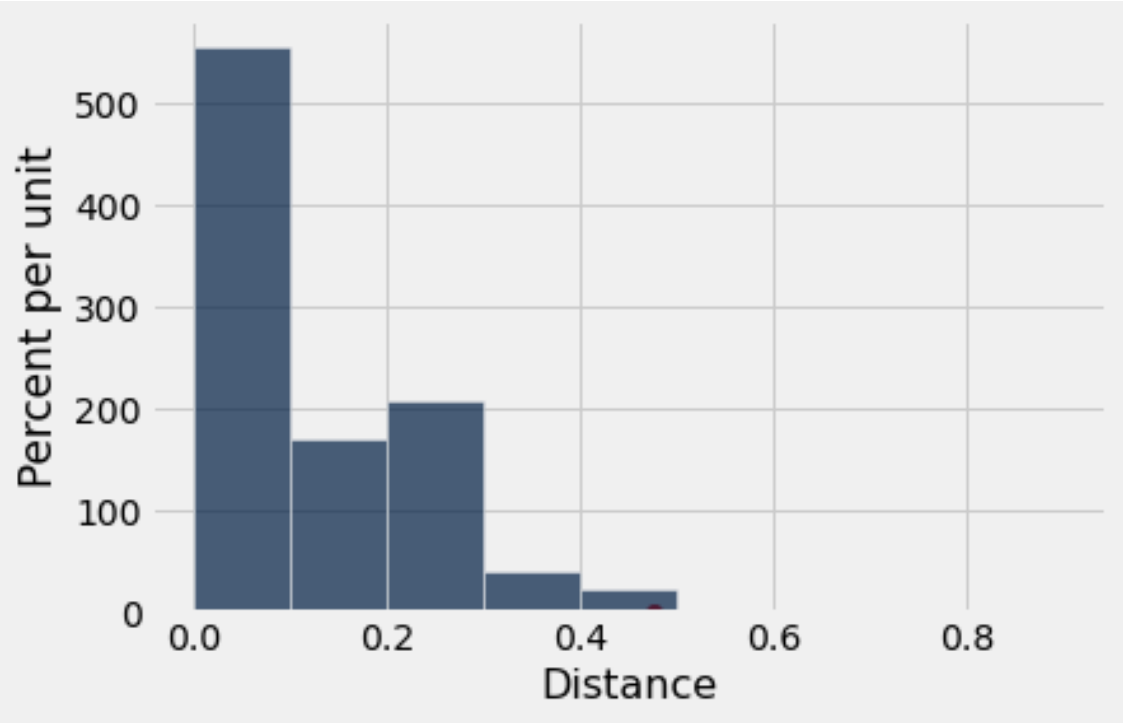
[45]:
```
labels = bta.select('Group')
results = bta.select('Result')
```

[46]:
```
# Repeat
distances = make_array()
for i in np.arange(2000):
    shuffled_results = results.sample(with_replacement=False).column(0)
    simulated = labels.with_column('Shuffled results', shuffled_results)
    distance = distance_between_group_proportions(simulated)
    distances = np.append(distances, distance)

distances
```

[46]:
```
array([0.3       , 0.0875    , 0.0875    , …, 0.3       , 0.04166667,
       0.17083333])
```

[47]:
```
Table().with_column('Distance', distances).hist(bins = np.arange(0, 1, 0.1))
plots.scatter(observed_distance, 0, color='red', s=40);
```



[48]:
```
np.average(distances >= observed_distance)
```

[48]:
```
0.011
```

[ ]:
```

```