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

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

Birth Weights

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

Out[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)

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

Out[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)

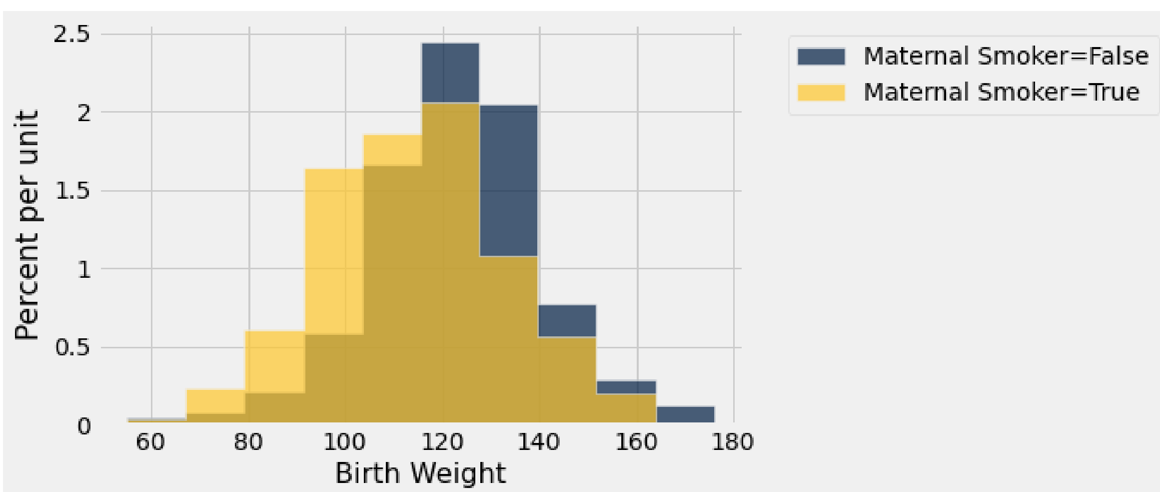
In [4]:
```python
smoking_and_birthweight.group('Maternal Smoker')
```

Out[4]:

Maternal Smoker	count
False	715
True	459

In [5]:
```python
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))
```



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

Out[6]:

Maternal Smoker	Birth Weight average
False	123.085
True	113.819

In [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)
```

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

Out[8]: 9.266142572024918

In [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
```

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

Out[10]:

Birth Weight
120
113
128
108
136
138
132
120
143
140

... (1164 rows omitted)

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

Out[11]:

Maternal Smoker
False
False
True
True
False
False
False
False
True
False

... (1164 rows omitted)

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

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

Out[13]:
```
array([138, 167, 102, ..., 131, 121, 112])
```

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

Out[14]:

Maternal Smoker	Shuffled weights
False	138
False	167
True	102
True	77
False	78
False	143
False	152
False	115
True	121
False	139

... (1164 rows omitted)

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

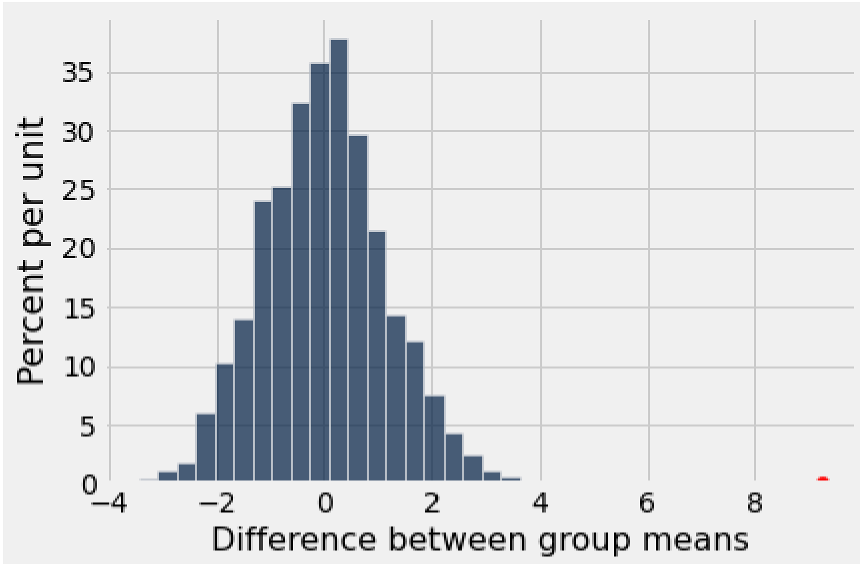
Out[15]:
```
0.995539101421457
```

In [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
```

Out[16]:
```
array([ 2.24042232, -0.51763792, -1.5550406 , ...,  0.09764919,
        0.423179  ,  1.02057986])
```

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



Deflategate

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

Team	Blakeman	Prioleau
Patriots	11.5	11.8
Patriots	10.85	11.2
Patriots	11.15	11.5
Patriots	10.7	11
Patriots	11.1	11.45
Patriots	11.6	11.95
Patriots	11.85	12.3
Patriots	11.1	11.55
Patriots	10.95	11.35
Patriots	10.5	10.9
Patriots	10.9	11.35
Colts	12.7	12.35

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

Team	Combined
Patriots	11.65
Patriots	11.025
Patriots	11.325
Patriots	10.85
Patriots	11.275
Patriots	11.775
Patriots	12.075
Patriots	11.325
Patriots	11.15
Patriots	10.7
Patriots	11.125
Colts	12.525
Colts	12.525
Colts	12.725
Colts	12.35

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

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

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

Out[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. ])
```

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

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

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

Team	Drop
Patriots	0.85
Patriots	1.475
Patriots	1.175
Patriots	1.65
Patriots	1.225
Patriots	0.725
Patriots	0.425
Patriots	1.175
Patriots	1.35
Patriots	1.8
Patriots	1.375
Colts	0.475
Colts	0.475
Colts	0.275
Colts	0.65

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

Out[25]:

Team	Drop average
Colts	0.46875
Patriots	1.20227

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

Out[26]: -0.733522727272728

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

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

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

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

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

Team	Drop
Patriots	1.175
Patriots	1.8
Patriots	0.275

... (12 rows omitted)

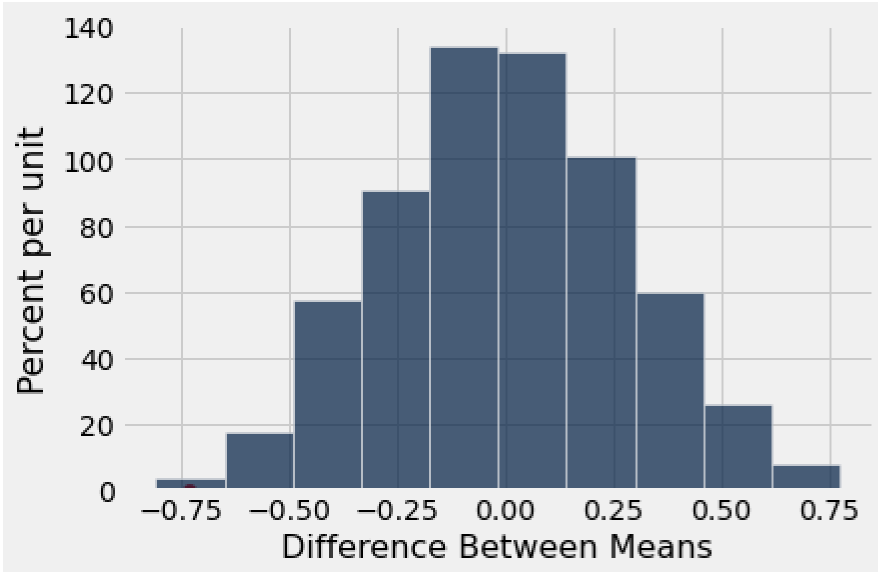
In [31]:
```
diff_between_means(simulated_football)
```

Out[31]: -0.017613636363636553

In [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)
```

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



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

Out[34]: 0.0012

Analyzing RCTs

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

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

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

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

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

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

Out[38]:

Group	Result sum
Control	2
Treatment	9

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

Out[39]:

Group	Result average
Control	0.125
Treatment	0.6

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

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

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

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

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

Out[42]:
```
0.475
```

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

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

Out[44]:
```
0.475
```

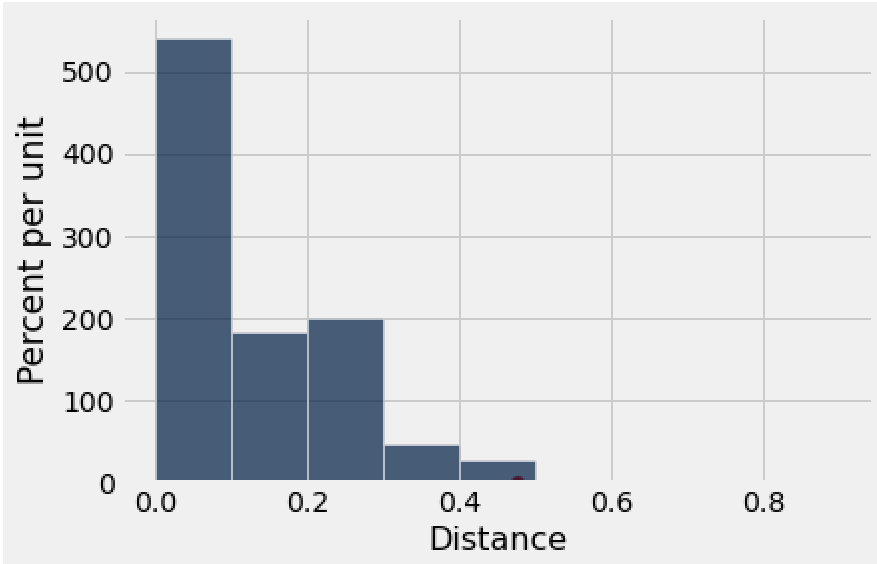
In [45]:
```
labels = bta.select('Group')
results = bta.select('Result')
```

In [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
```

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

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



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

Out[48]: 0.01

In [ ]:
```
```