```
from datascience import *
import matplotlib
path_data = 'https://www.inferentialthinking.com/data/'
matplotlib.use('Agg', warn=False)
%matplotlib inline
import matplotlib.pyplot as plots
plots.style.use('fivethirtyeight')
import numpy as np
np.set_printoptions(threshold=50)
```

```
/usr/local/lib/python3.7/dist-packages/datascience/tables.py:17: MatplotlibDeprecat
  matplotlib.use('agg', warn=False)
/usr/local/lib/python3.7/dist-packages/datascience/util.py:10: MatplotlibDeprecatio
  matplotlib.use('agg', warn=False)
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: MatplotlibDeprecati
  after removing the cwd from sys.path.
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=94

Enter your authorization code:
4/1AY0e-g7qDPCnxfEXCJCZYgCSmvORNkq9il1E4lLTCB9XGmHu2qU9EBgypsE
Mounted at /content/drive
```

```
from google.colab import files
uploaded = files.upload()
```

Choose Files healthcare-…ke-data.csv

- **healthcare-dataset-stroke-data.csv**(text/csv) - 316971 bytes, last modified: 1/26/2021 - 100% done

```
Saving healthcare-dataset-stroke-data.csv to healthcare-dataset-stroke-data.csv
```

```
# this path may be slightly different for you, be sure to copy the path correctly
stroke = Table.read_table("/content/drive/MyDrive/Colab Notebooks/healthcare-dataset-str
```

```
stroke.num_rows, stroke.num_columns
```

```
(5110, 12)
```

```
stroke.show(10)
```

id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_
9046	Male	67	0	1	Yes	Private	U
51676	Female	61	0	0	Yes	Self-employed	F
31112	Male	80	0	1	Yes	Private	F
						employed	

We are going to be studying the relationship between several health related variables. These variables will be both categorical and quantitative.

1. Exploratory Data Analysis

We first want to get an overall understanding of the dataset. Please perform the following tasks.

- Find the proportion of individuals in the data set who have hypertension and who have heart disease. It may be helpful to look at the data documentation. Please also find the proportion for both Male and Female.
- Find the average BMI and glucose level for the whole data set and for both Male and Female. The function `np.nanmean` may be helpful.

```
# make a male dataset and female dataset
males = stroke.where('gender', 'Male')
females = stroke.where('gender', 'Female')
```

```
# find the proportion who have hypertension
np.mean(stroke.column('hypertension'))
```

```
    0.0974559686888454
```

```
# find the proportions of males who have hypertension
np.mean(males.column('hypertension'))
```

```
    0.1049645390070922
```

```
# find the proportions of females who have hypertension
np.mean(females.column('hypertension'))
```

```
    0.09218436873747494
```

```
# find the proportion who have heart disease
np.mean(stroke.column('heart_disease'))
```

```
    0.05401174168297456
```

```
#find the proportion of males who have heart disease
np.mean(males.column('heart_disease'))
```

```
    0.07706855791962175
```

```
#find the proportion of females who have heart disease
```

```
np.mean(females.column('heart_disease'))
```

```
0.03774215096860387
```

```
# average bmi
np.nanmean(stroke.column('bmi'))
```

```
28.893236911794666
```

```
# average bmi male
np.nanmean(males.column('bmi'))
```

```
28.64793635007459
```

```
# average bmi female
np.nanmean(females.column('bmi'))
```

```
29.065757680358992
```

```
# average glucose level
np.mean(stroke.column('avg_glucose_level'))
```

```
106.1476771037182
```

```
# average glucose level male
np.mean(males.column('avg_glucose_level'))
```

```
109.08852009456265
```

```
# average glucose level female
np.mean(females.column('avg_glucose_level'))
```

```
104.05780895123581
```

2. A/B Test

Perform an A/B test to determine whether there is significant evidence that males have a higher `avg_glucose_level` than females. Let μ_1 and μ_2 be the true unknown population average glucose levels for males and females respectively. The null hypothesis is

$$H_0 : \mu_1 - \mu_2 = 0$$

and the alternative hypothesis is

$$H_1 : \mu_1 - \mu_2 > 0.$$

See the bulleted steps below, but basically, we can borrow some of the same ideas from 12.1 and 12.2 of the textbook. Essentially, we want to do the process where we shuffle the gender labels and recompute the difference in means for these shuffled labels. We then want to repeat this many times, which will give us an array of difference of mean values under the null hypothesis assumption that the mean for males is the same as the mean for females.

- First, compute the observed difference in sample mean glucose level between males and females (males minus females).
- Then, use the random permutations method described in 12.1 to see how this difference in means will vary under the null hypothesis. The functions below will help.

```
obs_diff = np.mean(males.column('avg_glucose_level')) - np.mean(females.column('avg_gluc
obs_diff
```

```
     5.03071114332684
```

```
# these functions may be useful

def difference_of_means(table, label, group_label):
    reduced = table.select(label, group_label)
    means_table = reduced.group(group_label, np.average)
    means = means_table.column(1)
    return means.item(1) - means.item(0)

def one_simulated_difference(table, label, group_label):
    shuffled_labels = table.sample(with_replacement = False).column(group_label) #shuffl
    shuffled_table = table.select(label).with_column('Shuffled Label', shuffled_labels)
    return difference_of_means(shuffled_table, label, 'Shuffled Label')
```

```
one_simulated_difference(stroke, 'avg_glucose_level', 'gender')
```

```
     1.2499951439522192
```

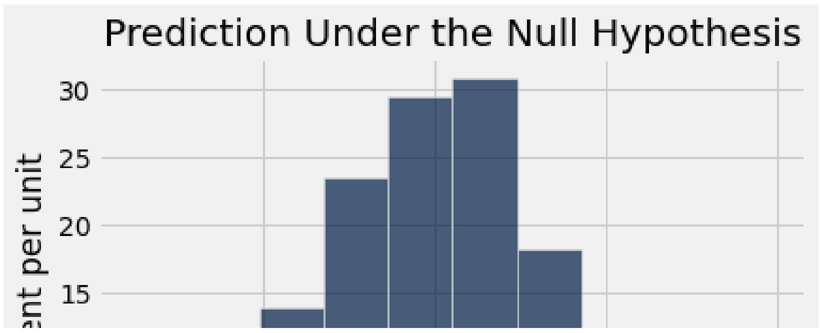
```
#create an array to store my simulated differences
differences = make_array()
# for some number of repition, create a new simulated difference and put it in the array
# put that array in its own table
repetitions = 1000
for i in np.arange(repetitions):
    new_difference = one_simulated_difference(stroke, 'avg_glucose_level', 'gender')
    differences = np.append(differences, new_difference)
```

```
obs_diff = difference_of_means(stroke, 'avg_glucose_level', 'gender')
obs_diff
```

```
     5.03071114332684
```

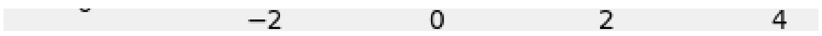
```
results = Table().with_column('Simulated Difference in Glucose Mean', differences).hist(
plots.title('Prediction Under the Null Hypothesis');

results
```



```
empirical_P = np.count_nonzero(differences >= obs_diff) / repetitions
empirical_P
```

```
    0.0
```



Please state the conclusion of your test with justification.

Our p-value calculates the number our simulated difference is greater than or equal to our observed difference. In this case, our p-value is 0, which implies that all our simulations are far less than (not as extreme) observed distribution. Hence is not a good representation of the null distribution (as the smaller the p-value more likely to reject the null hypothesis). In our graph, this is clearly represented as the closest mean is approximately around 4 while our observed distribution is 5.031. Our analysis is trying to examine the relationship with gender and its effect on glucose levels. Our null hypothesis takes both genders' (randomized) glucose levels means and concludes that there are no differences between the two. In contrast, the alternative equations show that gender has an effect on glucose levels ($H1{:}\mu1-\mu2>0.$).

3. Using the Bootstrap to Make a CI and Perform a Test

We will now make a 95% confidence interval for the true difference in average glucose level between males and females. Recall, to make confidence intervals we resample our sample, with replacement. We have only done this to make confidence intervals for single parameters, but now are interested in a single parameter. Here is how we will do this.

1. Using the `bootstrap_mean` function create an array of bootstrap sample mean `avg_glucose_level`s for both males and females. Make sure the number of repititions is the same for each gender!
2. To get the bootstrap differences, simply subtract the female bootstrap sample means from the male values. For example, if the bootstrap means are in `bstrap_means_male` and `bstrap_means_female`, we want `bstrap_means_male - bstrap_means_female`.
3. Find the middle 95% of these bootstrap values.

4.Use the bootsrap confidence inteerval to test the hypothesis from question 2, that males and females have the same average glucose level in the population. Hint: If 0 is not in your CI, then 0 is not a plausible value for the true difference in means.

```
def bootstrap_mean(original_sample, label, replications):

    """Returns an array of bootstrapped sample means:
      Returns an array of bootstrapped sample means:
    original_sample: table containing the original sample
    label: label of column containing the variable
    replications: number of bootstrap samples
    """

    just_one_column = original_sample.select(label)
    means = make_array()
    for i in np.arange(replications):
        bootstrap_sample = just_one_column.sample()
        resampled_mean = np.mean(bootstrap_sample.column(0))
        means = np.append(means, resampled_mean)

    return means
```

```
# run for males
reps = 1000
male_bstrap = bootstrap_mean(males, 'avg_glucose_level', reps)
```

```
# run for females
reps = 1000
female_bstrap = bootstrap_mean(females, 'avg_glucose_level', reps)
```

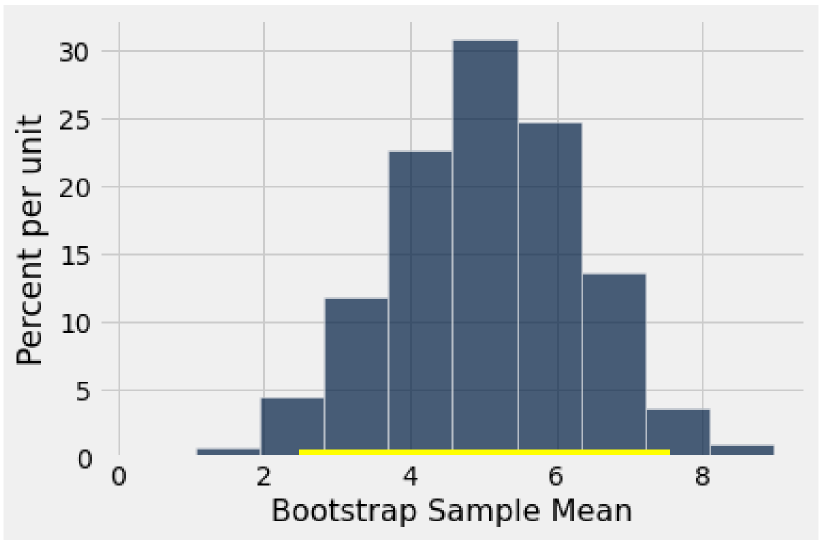
```
bstrap = male_bstrap - female_bstrap
bstrap
```

```
array([6.62300733, 6.49958698, 4.75207146, ..., 5.81161954, 5.05688362,
       5.84193263])
```

```
left = percentile(2.5, bstrap)
right = percentile(97.5, bstrap)
left, right
```

```
(2.487181630400272, 7.5616890092240965)
```

```
resampled_means = Table().with_column('Bootstrap Sample Mean', bstrap)
resampled_means.hist()
plots.plot(make_array(left, right), make_array(0, 0), color= 'yellow', lw=8);
```



The range of the 95% confidence interval is from approximetly 2.5 to 7.6. And since the range does not have 0 within it we can reject the null hypothesis (null hypothesis is the mean of each gender is equal).

4. Bootstrap CI for Difference in Proportion

We now are interested in rural versus urban and its effect on heart disease.

- First, find the percentage of the full data set that lives in urban areas.
- Make one table for urban dwellers and one for only rural dwellers.
- Compute the difference in heart disease proportion among urban and rural dwellers.
- Repeat a similar bootstrapping procedure to generate a 95% confidence interval for the difference in heart disease proportion between urban and rural. Report the interval, and perform a hypothesis test to determine if there is significant evidence that the proportion is higher in urban areas. Be sure to state the null and alternative hypotheses and a p-value. **The null hypothesis should be** *The proporiton of urban dwellers with heart diesease is the same as the proportion of rural dwellers with heart disease.*
- Change the confidence level to 90% and 99%. Write a sentence for each of these confidence levels descirbing how the size of the interval changes and how the conclusion of the hypotheseis test changes.

```
# urban
urban = stroke.where('Residence_type', 'Urban')
```

```
# percentage of urban
percentage_urban = np.mean(stroke.column('Residence_type') == 'Urban')* 100
percentage_urban
```

```
    50.80234833659492
```

```
# table urban
urban_table = stroke.where('Residence_type', 'Urban')
urban_table
```

id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_t
9046	Male	67	0	1	Yes	Private	U

```
# table rural
rural_table = stroke.where('Residence_type', are.equal_to('Rural'))
rural_table
```

id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_t
51676	Female	61	0	0	Yes	Self-employed	F
31112	Male	80	0	1	Yes	Private	F
1665	Female	79	1	0	Yes	Self-employed	F
53882	Male	74	1	1	Yes	Private	F
27419	Female	59	0	0	Yes	Private	F
12109	Female	81	1	0	Yes	Private	F
12095	Female	61	0	1	Yes	Govt_job	F
58202	Female	50	1	0	Yes	Self-employed	F
70630	Female	71	0	0	Yes	Govt_job	F
64778	Male	82	0	1	Yes	Private	F

```
# run for urban
urban_mean = np.mean(urban_table.column('heart_disease'))
urban_mean
```

```
0.054699537750385205
```

```
# run for rural
rural_mean = np.mean(rural_table.column('heart_disease'))
rural_mean
```

```
0.053301511535401754
```

```
bstrap_residence = urban_mean - rural_mean
bstrap_residence
```

```
0.0013980262149834513
```

```
differences = make_array()
repetitions = 1000
for i in np.arange(repetitions):
  new_differences = one_simulated_difference(stroke, 'heart_disease', 'Residence_type')
  differences = np.append(differences, new_differences)
```

```
def bootstrap_mean(original_sample, label, replications):

    """Returns an array of bootstrapped sample means:
    original_sample: table containing the original sample
    label: label of column containing the variable
    replications: number of bootstrap samples
    """

    just_one_column = original_sample.select(label)
    means = make_array()
    for i in np.arange(replications):
        bootstrap_sample = just_one_column.sample()
        resampled_mean = np.mean(bootstrap_sample.column(0))
        means = np.append(means, resampled_mean)

    return means


reps = 500
urban_bstrap = bootstrap_mean(urban_table, 'heart_disease', reps)
urban_bstrap
```

```
array([0.05624037, 0.05431433, 0.05046225, ..., 0.05469954, 0.05392912,
       0.04853621])
```

```
reps = 500
rural_bstrap = bootstrap_mean(rural_table, 'heart_disease', reps)
rural_bstrap
```

```
array([0.05568815, 0.05409706, 0.04534606, ..., 0.0548926 , 0.04574383,
       0.04256165])
```

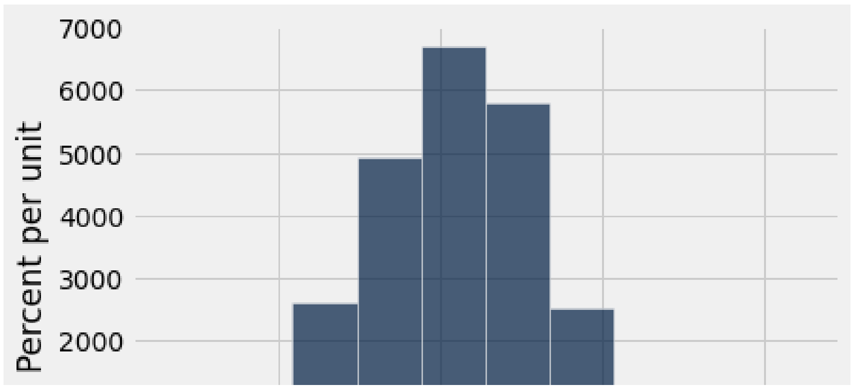
```
bstrap_1 = urban_bstrap - rural_bstrap
bstrap_1
```

```
array([ 0.00055222,  0.00021727,  0.00511619, ..., -0.00019306,
        0.00818529,  0.00597455])
```

```
left = percentile(2.5, bstrap_1)
right = percentile(97.5, bstrap_1)
left, right
```

```
(-0.010694195708960484, 0.013477683677109262)
```

```
resampled_means = Table().with_column('Bootstrap Sample Mean', bstrap_1)
resampled_means.hist()
plots.plot(make_array(left, right), make_array(0, 0), color= 'yellow', lw=8);
```



```
left = percentile(5.0, bstrap)
right = percentile(95.0, bstrap)
left, right
```

```
    (-0.008730462261872803, 0.011727546080929845)
```

```
left = percentile(0.5, bstrap)
right = percentile(99.5, bstrap)
left, right
```

```
    (-0.015278998471425963, 0.01797882551088327)
```

Summary of the Intervals: The difference in the 95% confidence interval range is 0.02417187939. At the same time, the 90% confidence interval got slightly narrower as the difference was 0.02045800834. The 99% confidence interval is 0.03325782398, which is wider than both CI. From this, we can conclude that a higher confidence interval has a wider range. Since our null hypothesis denotes that our two proportions are equal, the difference in proportions is 0. All intervals above include 0; hence we can conclude our null hypothesis might be true (as it is a plausible value). In contrast, the alternative is that there is a difference in the proportions between the urban and rural in terms of heart disease.