Examining Racial Discrimination in the US Job Market

Background

Racial discrimination continues to be pervasive in cultures throughout the world. Researchers examined the level of racial discrimination in the United States labor market by randomly assigning identical résumés to black-sounding or white-sounding names and observing the impact on requests for interviews from employers.

Data

In the dataset provided, each row represents a resume. The 'race' column has two values, 'b' and 'w', indicating black-sounding and white-sounding. The column 'call' has two values, 1 and 0, indicating whether the resume received a call from employers or not.

Note that the 'b' and 'w' values in race are assigned randomly to the resumes when presented to the employer.

Exercises

You will perform a statistical analysis to establish whether race has a significant impact on the rate of callbacks for resumes.

Answer the following questions **in this notebook below and submit to your Github account**.

1. What test is appropriate for this problem? Does CLT apply?
2. What are the null and alternate hypotheses?
3. Compute margin of error, confidence interval, and p-value. Try using both the bootstrapping and the frequentist statistical approaches.
4. Write a story describing the statistical significance in the context or the original problem.
5. Does your analysis mean that race/name is the most important factor in callback success? Why or why not? If not, how would you amend your analysis?

You can include written notes in notebook cells using Markdown:

- In the control panel at the top, choose Cell > Cell Type > Markdown
- Markdown syntax: http://nestacms.com/docs/creating-content/markdown-cheat-sheet (http://nestacms.com/docs/creating-content/markdown-cheat-sheet)

Resources

- Experiment information and data source: http://www.povertyactionlab.org/evaluation/discrimination-job-market-united-states (http://www.povertyactionlab.org/evaluation/discrimination-job-market-united-states)
- Scipy statistical methods: http://docs.scipy.org/doc/scipy/reference/stats.html (http://docs.scipy.org/doc/scipy/reference/stats.html)
- Markdown syntax: http://nestacms.com/docs/creating-content/markdown-cheat-sheet (http://nestacms.com/docs/creating-content/markdown-cheat-sheet)
- Formulas for the Bernoulli distribution: https://en.wikipedia.org/wiki/Bernoulli_distribution (https://en.wikipedia.org/wiki/Bernoulli_distribution)

In [4]:
```
import pandas as pd
import numpy as np
```

In [5]:

In [34]:
```
# number of callbacks for black-sounding names
len(data)
```

Out[34]: 4870

In [7]:

...

Your answers to Q1 and Q2 here

From the analysis below we can see that the proportion of callbacks is 8.0% overall, but 9.65% for whites and 6.45% for blacks.

The conventional wisdom is that the Central Limit Theorem may be applied with a sample size of 30 or more for a symmettric distibution, and a sample size of 300 or more for very heavily skewed distribution.
Since our sample sizes are each in the thousands, this condition is clearly satified.

The Null Hypothesis is always that "nothing is going on", and therefore that the proportion of callbacks for the White and Black groups are identical. A single "pooled proportion is therefore calculated, and the observed difference in sample means (9.65% - 6.45%) is divided by the standard error of the difference ion means, computed also using the pooled proportion of 8.0%.
This standard error (SE) of the difference in sample means may be computed as:
(0.08*0.92/2435 + 0.08*0.92/2435)**0.5,
which is equal to 0.0077.

The derived z-score is therfore 4.11. It is helpful for me personally (as a former statistician) to make the computation the long way as with the various packages the exact methodology is somewhat ambigous and hence a basis of comparison, and a form of crosscheck is provided.In fact both packages (statsmodels and scipy.stats) generate th same z-score, so that appears to be confirmatory of the methodology employed.

The 99% confidence interval is 0.08 +/- 2.33 * SE, so

The Alternative hypothesis is that there is discrimination. Intuitively, this should b a one-tailed test (namely that

In [ ]:
```
# Your solution to Q3 here
```

In [39]:
```
w = data[data.race=='w']
b = data[data.race=='b']
len(w), len(b), w['call'].tail(50), b['call'].tail(50)# shows the sample sizes are eac
```

...

In [23]:
```
w_callback_prop= sum(data[data.race=='w'].call)/len(w)
b_callback_prop= sum(data[data.race=='b'].call)/len(b)
overall_callback_prop= (sum(data[data.race=='w'].call) + sum(data[data.race=='b'].call
```

Out[23]: (0.09650924024640657, 0.06447638603696099, 0.08049281314168377)

In [57]:
```
# the standard error of the difference in sample means is the square root of the sum o
# (assuming that the two samples are independent)
SE=  (0.08*0.92/2435 + 0.08*0.92/2435)**0.5
SE
z_value= (0.0965-0.0645)/SE
z_value
```

Out[57]: 0.007775072049819011

In [60]:
```
SE # equal to 0.0077
# 95% Confidence interval derivation:
Upper = overall_callback_prop + 1.96*SE
Lower = overall_callback_prop - 1.96*SE
```

Out[60]: (0.06525367192403851, 0.09573195435932903)

In [55]:
```
# Derivation of P-value using the statsmodels package
import numpy as np
from statsmodels.stats.proportion import proportions_ztest
counts = np.array([sum(data[data.race=='w'].call), sum(data[data.race=='b'].call)])

n_obs = np.array([len(w), len(b)])
counts,n_obs

stat, pval = proportions_ztest(counts, n_obs)
```

0.000040

In [56]:
```
# Derivation of P-value using the scipy.stats package
from scipy.stats import ttest_ind
# cat1 = my_data[my_data['Category']=='cat1']
# cat2 = my_data[my_data['Category']=='cat2']
ttest_ind(w['call'], b['call'])
```

Out[56]: Ttest_indResult(statistic=4.1147052908617514, pvalue=3.9408021031288859e-05)

In [61]:
```
# Bootstrap Approach
# draw_bs_reps()
def draw_bs_reps(data, func, size=1):
    """Draw bootstrap replicates."""
    return func(np.random.choice(data,size=len(data)))
    # Initialize array of replicates: bs_replicates
    bs_replicates = np.empty(size)

    # Generate replicates
    for i in range(size):
        bs_replicates[i] = bootstrap_replicate_1d(data, func)
```

In [65]:
```
white = w['call']
black = b['call']
call_concat= np.concatenate((white, black))
mean_call = np.mean(call_concat)  # Compute mean for all callbacks
empirical_diff_means = np.mean(white) - np.mean(black)
w_shifted = white - np.mean(white) + mean_call  # Generate shifted arrays
b_shifted = black - np.mean(black) + mean_call
bs_replicates_white = draw_bs_reps(w_shifted, np.mean, size=10000)#Compute 10000 boots
bs_replicates_black = draw_bs_reps(b_shifted, np.mean, size=10000)
bs_replicates = bs_replicates_white - bs_replicates_black   # Get replicates of differ

p = np.sum(bs_replicates > empirical_diff_means) /10000 # len(bs_replicates) # Compute
print('p-value =', p) # From which we note that with the Bootstrap approach the P-valu
```

```
('p-value =', 0)
```

The statistical significance of the race/name variable was investigated using two statistical packages, and
also by a formal calculation following conventional statistical methodologies.
The same z-score (of 4.11) and P-value (of 0.00004) was obtained in each case.

Accordingly, one may infer that in this study, race had a statistically significant impact on the rate of callbacks

This analysis does NOT in itself prove that race/name is the most important factor in callback success (although it is clearly a highly statistically significant factor). The correct way to investigate this would be to build a Classification Model with the target variable being 'callback', for example using logistic regression. The t-value for the 'race' variable could then be compared to the t-value for other parameters, in

Your answers to Q4 and Q5 here