index

February 1, 2022

The GitHub link is:

`https://github.com/miladshiraniUCB/dsc-website-ab-testing-lab.git`

1 Website A/B Testing - Lab

1.1 Introduction

In this lab, you'll get another chance to practice your skills at conducting a full A/B test analysis. It will also be a chance to practice your data exploration and processing skills! The scenario you'll be investigating is data collected from the homepage of a music app page for audacity.

1.2 Objectives

You will be able to: * Analyze the data from a website A/B test to draw relevant conclusions * Explore and analyze web action data

1.3 Exploratory Analysis

Start by loading in the dataset stored in the file 'homepage_actions.csv'. Then conduct an exploratory analysis to get familiar with the data.

> Hints: * Start investigating the id column: * How many viewers also clicked? * Are there any anomalies with the data; did anyone click who didn't view? * Is there any overlap between the control and experiment groups? * If so, how do you plan to account for this in your experimental design?

[227]:

```python
#Your code here
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as stats

%matplotlib inline

sns.set_style('darkgrid')

df = pd.read_csv("homepage_actions.csv")
```

```
[228]: df.head()
```

```
[228]:                     timestamp      id       group action
       0  2016-09-24 17:42:27.839496  804196  experiment   view
       1  2016-09-24 19:19:03.542569  434745  experiment   view
       2  2016-09-24 19:36:00.944135  507599  experiment   view
       3  2016-09-24 19:59:02.646620  671993     control   view
       4  2016-09-24 20:26:14.466886  536734  experiment   view
```

```
[229]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8188 entries, 0 to 8187
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   timestamp  8188 non-null   object
 1   id         8188 non-null   int64
 2   group      8188 non-null   object
 3   action     8188 non-null   object
dtypes: int64(1), object(3)
memory usage: 256.0+ KB
```

```
[230]: df.isna().sum()
```

```
[230]: timestamp    0
       id           0
       group        0
       action       0
       dtype: int64
```

```
[231]: print(len(df))
```

```
8188
```

```
[232]: df["id"].nunique()
```

```
[232]: 6328
```

```
[233]: df["group"].value_counts()
```

```
[233]: control       4264
       experiment    3924
       Name: group, dtype: int64
```

```
[234]: df["action"].value_counts()
```

[234]:
```
view     6328
click    1860
Name: action, dtype: int64
```

People's `id` who viewed and clicked the add

[235]:
```
grouped = df.groupby(["id", "group"])["action"].count()
data = grouped.to_frame()
data.reset_index(inplace = True)
data.head()
dd = data.sort_values("action", ascending = False)
v_and_c = dd.loc[dd["action"] == 2]
v_and_c.reset_index(inplace = True)
v_and_c.drop(columns = ["index"], axis = 1, inplace = True)
```

```
/opt/anaconda3/envs/learn-env/lib/python3.8/site-
packages/pandas/core/frame.py:4163: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  return super().drop(
```

[236]:
```
view_set = set(df.loc[df["action"] == "view"]["id"])
click_set = set(df.loc[df["action"] == "click"]["id"])
view_click = view_set - click_set
click_view = click_set - view_set
```

[237]:
```
cc = df[df["group"] == "control"]["id"]# .drop_duplicates()
print(sum(cc.duplicated()))

ee = df[df["group"] == "experiment"]["id"]# .drop_duplicates()
print(sum(ee.duplicated()))
```

```
932
928
```

[238]:
```
v = len(view_set)
c = len(click_set)
v_c = len(v_and_c)
cont_v_c = len(v_and_c[v_and_c["group"] == "control"])
expt_v_c = len(v_and_c[v_and_c["group"] == "experiment"])
cont_v_not_c = (len(df[df["group"]=="control"]) -
                len(v_and_c[v_and_c["group"] == "experiment"]))
expt_v_not_c = (len(df[df["group"]=="experiment"]) -
                len(v_and_c[v_and_c["group"] == "experiment"]))
```

[239]:

```
print("Number of people who viewed                   : ", v)
print("Number of people who viewed and clicked       : ", c)
print("Number of people who viewed but did not click : ", len(view_click))
print("Number of people who clicked but did not view : ", len(click_view))
print("\n")
print("number of people in control group who viewed and clicked          : ",
      cont_v_c)

print("number of people in experiment group who viewed and clicked       : ",
      expt_v_c)

print("number of people in control group who viewed and did no click     : ",
      cont_v_not_c)

print("number of people in experiment group who viewed and did not click : ",
      expt_v_not_c)
```

```
Number of people who viewed                   :  6328
Number of people who viewed and clicked       :  1860
Number of people who viewed but did not click :  4468
Number of people who clicked but did not view :  0



number of people in control group who viewed and clicked          :  932
number of people in experiment group who viewed and clicked       :  928
number of people in control group who viewed and did no click     :  3336
number of people in experiment group who viewed and did not click :  2996
```

Making new columns to know the group of people who clicked the add

[240]:

```
# data = pd.DataFrame([])

# data["id"] = df["id"]
# data["control"] = df["group"].apply( lambda x: 1 if x == "control" else 0)
# data["experiment"] = df["group"].apply( lambda x: 1 if x == "experiment" else 0)

# data["control_click"] = ((df["action"] == "click").astype(int) *
#                         (df["group"]   == "control").astype(int))

# data["experiment_click"] = ((df["action"] == "click").astype(int) *
#                           ( df["group"]   == "experiment").astype(int))
```

1.4 Creating New DataFrame for Control and Experimental groups

[292]:
```
## From GitHub

df["count"] = 1
control = df[df["group"] == "control"].pivot(index = "id",
                                             columns = "action",
                                             values = "count")
control.reset_index(inplace = True)
control.fillna(value = 0,inplace = True)
control
```

[292]:
```
action      id  click  view
0       182994    1.0   1.0
1       183089    0.0   1.0
2       183248    1.0   1.0
3       183515    0.0   1.0
4       183524    0.0   1.0
…          …    …    …
3327    936786    0.0   1.0
3328    937003    0.0   1.0
3329    937073    0.0   1.0
3330    937108    0.0   1.0
3331    937217    1.0   1.0

[3332 rows x 3 columns]
```

[295]:
```
control_mean_click = control.click.mean()
control_mean_std   = control.click.std()
```

[296]:
```
df["count"] = 1
experiment = df[df["group"] == "experiment"].pivot(index = "id",
                                             columns = "action",
                                             values = "count")
experiment.reset_index(inplace = True)
experiment.fillna(value = 0,inplace = True)
experiment
```

[296]:
```
action      id  click  view
0       182988    0.0   1.0
1       183136    0.0   1.0
2       183141    1.0   1.0
3       183283    0.0   1.0
4       183389    0.0   1.0
…          …    …    …
2991    935382    0.0   1.0
2992    935576    0.0   1.0
```

```
2993    935742    1.0   1.0
2994    936129    0.0   1.0
2995    937139    1.0   1.0

[2996 rows x 3 columns]
```

1.5 My Analysis

1.5.1 First Method, by using equations:

[297]:

```
control_mean_click = control.click.mean()
control_std_click   = control.click.std()

experiment_mean_click = experiment.click.mean()
experiment_std_click   = experiment.click.std()

z_num = (control_mean_click - experiment_mean_click)
z_denum = np.sqrt( control_std_click**2 / (len(control) - 1) +
                   experiment_std_click**2 / (len(experiment) -1 ))
z = (z_num/z_denum)

print(z)
pval = 1 - stats.norm.sf(z)
print(pval)
```

```
-2.615023686946102
0.004461063385910569
```

1.5.2 Second Method by Using one tailed two samples t-test

[298]:

```
results = stats.ttest_ind(control.click, experiment.click, equal_var = False)
print("t-score : ", results.statistic)
print("p-value : ", results.pvalue/2)
```

```
t-score :  -2.615440020788211
p-value :  0.004466402814337101
```

1.6 Conduct a Statistical Test

Conduct a statistical test to determine whether the experimental homepage was more effective than that of the control group.

[291]:

```
# Your code here
### Check the Solution in GitHub
```

1.7 Verifying Results

One sensible formulation of the data to answer the hypothesis test above would be to create a binary variable representing each individual in the experiment and control group. This binary variable would represent whether or not that individual clicked on the homepage; 1 for they did and 0 if they did not.

The variance for the number of successes in a sample of a binomial variable with n observations is given by:

1.8 $n \bullet p(1-p)$

Given this, perform 3 steps to verify the results of your statistical test: 1. Calculate the expected number of clicks for the experiment group, if it had the same click-through rate as that of the control group. 2. Calculate the number of standard deviations that the actual number of clicks was from this estimate. 3. Finally, calculate a p-value using the normal distribution based on this z-score.

1.8.1 Step 1:

Calculate the expected number of clicks for the experiment group, if it had the same click-through rate as that of the control group.

[ ]:

```
#Your code here
### Check the Solution in GitHub
```

1.8.2 Step 2:

Calculate the number of standard deviations that the actual number of clicks was from this estimate.

[ ]:

```
#Your code here
### Check the Solution in GitHub
```

1.8.3 Step 3:

Finally, calculate a p-value using the normal distribution based on this z-score.

[ ]:

```
#Your code here
### Check the Solution in GitHub
```

1.8.4 Analysis:

Does this result roughly match that of the previous statistical test?

> Comment: **Your analysis here**

1.9 Summary

In this lab, you continued to get more practice designing and conducting AB tests. This required additional work preprocessing and formulating the initial problem in a suitable manner. Additionally, you also saw how to verify results, strengthening your knowledge of binomial variables, and reviewing initial statistical concepts of the central limit theorem, standard deviation, z-scores, and their accompanying p-values.