A/B Testing the Udacity Website

Exercise 1

Begin by importing Udacity's data on user behavior by going to http://www.github.com/nickeubank/MIDS_Data/ and using the udacity_AB_testingfolder, or by clicking here. Note that there are TWO datasets for this test – one for the control data (users who saw the original design), and one for treatment data (users who saw the experimental design). Udacity decided to show their test site to 1/2 of visitors, so there are roughly the same number of users appearing in each dataset (though this is not a requirement of AB tests).

In [1]:
```
import pandas as pd
import numpy as np
from scipy import stats
```

In [2]:
```
control_data = pd.read_csv("./data/control_data.csv")
experiment_data = pd.read_csv("./data/experiment_data.csv")
```

In [3]:
```
experiment_data.head()
```

Out[3]:

	Date	Pageviews	Clicks	Enrollments	Payments
0	Sat, Oct 11	7716	686	105.0	34.0
1	Sun, Oct 12	9288	785	116.0	91.0
2	Mon, Oct 13	10480	884	145.0	79.0
3	Tue, Oct 14	9867	827	138.0	92.0
4	Wed, Oct 15	9793	832	140.0	94.0

In [4]:
```
control_data.head()
```

Out[4]:

	Date	Pageviews	Clicks	Enrollments	Payments
0	Sat, Oct 11	7723	687	134.0	70.0
1	Sun, Oct 12	9102	779	147.0	70.0
2	Mon, Oct 13	10511	909	167.0	95.0
3	Tue, Oct 14	9871	836	156.0	105.0
4	Wed, Oct 15	10014	837	163.0	64.0

Exercise 2

Explore the data. Can you identifying the unit of observation of the data (e.g. what is represented by each row)?

> Here, each row indicates the performance of ~~a course~~ on a specific day. For each

> row, it contains five properties (columns).
>
> - Date: Date
> - Pageviews: Number of unique **cookies** to view the course overview page that day.
> - Clicks: Number of **unique cookies** to click the course overview page that day.
> - Enrollments: Number of **user-ids to** enroll in the free trial that day.
> - Payments: Number of **user-ids** who who enrolled on that day to remain enrolled for 14 days and thus make a payment.
>
> One thing ~~two~~ notice is that unite of pageviews and clicks is different from enrollments and payments. The first two properties' unit is unique cookies while the last two properties are user-ids.

Exercise 3

The easiest way to analyze this data is to stack it into a single dataset where each observation is a day-treatment-arm (so you should end up with two rows per day, one for those who are in the treated groups, and one for those who were in the control group). Note that currently nothing in the data identifies whether a given observation is a treatment group observation or a control group observation, so you'll want to make sure to add a "treatment" indicator variable.

The variables in the data are:

- Pageviews: number of unique users visiting homepage
- Clicks: number of those users clicking "Start Free Trial"
- Enrollments: Number of people enrolling in trial
- Payments: Number of people who eventually pay for the service

In [5]:

```
control_data['Treatment'] = 0
experiment_data['Treatment'] = 1

data = pd.concat([control_data, experiment_data]).sort_values("Date")

# Convert Date to standard np datetime, assume default year is 2017
data['Date'] = pd.to_datetime(data['Date'] , format='%a, %b %d') \
                 .apply(lambda x: x.replace(year=2017))

data.sort_values(['Date', 'Treatment'], inplace = True)
data.head(n=6)
```

Out[5]:

	Date	Pageviews	Clicks	Enrollments	Payments	Treatment
0	2017-10-11	7723	687	134.0	70.0	0
0	2017-10-11	7716	686	105.0	34.0	1
1	2017-10-12	9102	779	147.0	70.0	0
1	2017-10-12	9288	785	116.0	91.0	1
2	2017-10-13	10511	909	167.0	95.0	0
2	2017-10-13	10480	884	145.0	79.0	1

Check is there unmatched record

In [6]:
```
sum(data.groupby("Date")["Pageviews"].count() == 1)
```

Out[6]: 0

0 means all records are pairs wised, one comes from the control group, and another comes from the treatment group.

Exercise 4

Given the outcomes of interest to Udacity, what outcomes do you want to measure? (In the language of the Potential Outcomes Framework, what are your Y variables?). Add these to your data.

> The Gross Conversion and Net Conversion are the most important metrics to care about, because they are associated with income.
>
> - $NetConv = \frac{Payments}{Clicks}$
> - $GrossConv = \frac{Enrollments}{Clicks}$

In [7]:
```
data['NetConv'] = data['Payments'] / data['Clicks']
data['GrossConv'] = data['Enrollments'] / data['Clicks']
data.head()
```

Out[7]:

	Date	Pageviews	Clicks	Enrollments	Payments	Treatment	NetConv	GrossConv
0	2017-10-11	7723	687	134.0	70.0	0	0.101892	0.195051
0	2017-10-11	7716	686	105.0	34.0	1	0.049563	0.153061
1	2017-10-12	9102	779	147.0	70.0	0	0.089859	0.188703
1	2017-10-12	9288	785	116.0	91.0	1	0.115924	0.147771
2	2017-10-13	10511	909	167.0	95.0	0	0.104510	0.183718

Exercise 5

Whenever you are working with experimental data, the first thing you want to do is verify that users actually were randomly sorted into the two arms of the experiment. In this data, half of users were supposed to be shown the old version of the site and half were supposed to see the new version.

Pageviews tells you how many unique users visited the welcome site we are experimenting on. Pageviews is what is sometimes called an “invariant” variable, meaning that it shouldn’t vary across treatment arms – after all, people have to visit the site before they get a chance to see the treatment, so there’s no way that being assigned to treatment or control should affect the number of pageviews assigned to each group.

“Invariant” variables are also an example of what are known as a “pre-treatment” variable, because pageviews are determined before users are manipulated in any way. That makes it analogous to gender or age in experiments where you have demographic data – a person's age and gender are determined before they experience any manipulations, so the value of any pre-treatment attributes should be the same across the two arms of our experiment. This is what is called "checking for balance." If pre-treatment attributes aren't balanced, then we know our attempt to randomly assign people to different groups failed.

To test the quality of the randomization, calculate the average number of pageviews for the treated group and for the control group. Do they look similar?

In [8]:

```
print("Average Pageviews of Treatment Group is {:.3f}".format(
    np.mean(data[data['Treatment'] == 1]["Pageviews"])))
print("Average Pageviews of Control Group is {:.3f}".format(
    np.mean(data[data['Treatment'] == 0]["Pageviews"])))
```

```
Average Pageviews of Treatment Group is 9315.135
Average Pageviews of Control Group is 9339.000
```

> According to above data, the two groups pageviews are very similar.

Exercise 6

"Similar" is a tricky concept – obviously, we expect some differences across groups since users were randomly divided across treatment arms. The question is whether the differences between groups are larger than we'd expect to emerge given our random assignment process. To evaluate this, let's use a ttest to test the statistical significance of the differences we see.

If you're using R, you can just use the t.test function.

If you're using Python, you can use the ttest function from scipy, which you can import as from scipy.stats import ttest_ind.

Note: Remember that scipy functions don't accept pandas objects, so you have to pass the numpy vectors underlying your data with the .values operator (e.g. df.my_column.values).

Does the difference in pageviews look statistically significant?

In [9]:

```
test_res = stats.ttest_ind(data[data['Treatment'] == 1]["Pageviews"],
                           data[data['Treatment'] == 0]["Pageviews"])

print("The p-value of Treatment t-test is {:.3f}".format(test_res.pvalue))
```

```
The p-value of Treatment t-test is 0.888
```

> Because the p-value is bigger than 0.05, which mean we cannot say the two group's Pageviews average is not identical. In another words, the the difference in pageviews does not significant.

Exercise 7

Pageviews is not the only pre-treatment variable in this data. What other measure is pre-treatment? Review the description of the experiment if you're not sure.

> Another pre-treatment variable in this data are:
>
> - Clicks
> - Click-through-probability (CTP) of the Free Trial Button. The CTP is cauculated by:$CPT = \frac{Pageviews}{Clicks}$

Exercise 8

Check if the other pre-treatment variable is also balanced.

a. Clicks

In [10]:
```
test_res = stats.ttest_ind(data[data['Treatment'] == 1]["Clicks"],
                           data[data['Treatment'] == 0]["Clicks"])

print("The p-value of Clicks t-test is {:.3f}".format(test_res.pvalue))
```

```
The p-value of Clicks t-test is 0.926
```

> According to the result of the t-test, there is no evidence that the average of Clicks of the two groups is not identical. Hence, the variable of Clicks is balanced.

b. CTP

In [11]:
```
data["CPT"] = data["Pageviews"] / data["Clicks"]
data.head()
```

Out[11]:

	Date	Pageviews	Clicks	Enrollments	Payments	Treatment	NetConv	GrossConv	CPT
0	2017-10-11	7723	687	134.0	70.0	0	0.101892	0.195051	11.241630
0	2017-10-11	7716	686	105.0	34.0	1	0.049563	0.153061	11.247813
1	2017-10-12	9102	779	147.0	70.0	0	0.089859	0.188703	11.684211
1	2017-10-12	9288	785	116.0	91.0	1	0.115924	0.147771	11.831847
2	2017-10-13	10511	909	167.0	95.0	0	0.104510	0.183718	11.563256

In [12]:
```
test_res = stats.ttest_ind(data[data['Treatment'] == 1]["CPT"],
                           data[data['Treatment'] == 0]["CPT"])

print("The p-value of CPT t-test is {:.3f}".format(test_res.pvalue))
```

```
The p-value of CPT t-test is 0.924
```

> According to the result of the t-test, there is no evidence that the average of CPT of the two groups is not identical. Hence, the variable of CPT is balanced.

Summary

All pre-treatment variables are balanced. It shows that the data ~~is sanity~~ for future analysis.

Exercise 9

Now that we've established we have good balance (meaning we think randomization was likely successful), we can evaluate the effects of the experiment. Test whether the two metrics you picked have different average values in the control group and treatment group. Because we've randomized, this is a consistent estimate of the Average Treatment Effect of Udacity's website change.

Did Udacity achieve their goal?

Note: You may discover some issues with your data. Can you figure out what's going on, and adjust?

In [13]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 74 entries, 0 to 36
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Date         74 non-null     datetime64[ns]
 1   Pageviews    74 non-null     int64
 2   Clicks       74 non-null     int64
 3   Enrollments  46 non-null     float64
 4   Payments     46 non-null     float64
 5   Treatment    74 non-null     int64
 6   NetConv      46 non-null     float64
 7   GrossConv    46 non-null     float64
 8   CPT          74 non-null     float64
dtypes: datetime64[ns](1), float64(5), int64(3)
memory usage: 5.8 KB
```

In [14]:
```
# There are null data in the dateset, drop them
full_data = data.dropna()
```

In [15]:
```
test_res = stats.ttest_ind(full_data[full_data['Treatment'] == 1]["NetConv"],
                           full_data[full_data['Treatment'] == 0]["NetConv"])

print("The p-value of Net Conversion t-test is {:.3f}".format(test_res.pvalue))

test_res = stats.ttest_ind(full_data[full_data['Treatment'] == 1]["GrossConv"],
                           full_data[full_data['Treatment'] == 0]["GrossConv"])

print("The p-value of Gross Conversion t-test is {:.3f}".format(test_res.pvalue))
```

```
The p-value of Net Conversion t-test is 0.593
The p-value of Gross Conversion t-test is 0.131
```

Summary :

According to p-value of t-test, the different average values in the control group and treatment group are not ~~**statistically significant**~~ Hence, Udacity haven't achieve their goal in this expeirment.

Exercise 10

One of the magic things about experiments is that all you have to do is compare averages to get an average treatment effect. However, you can do other things to try and increase the statistical power of your experiments, like add controls in a linear regression model.

As you likely know, a bivariate regression is exactly equivalent to a t-test, so let's start by re-estimating the effect of treatment on payments-per-click using a linear regression. Can you replicate the results from your t-test? They shouldn't just be close – they should be numerically equivalent (i.e. exactly the same to the limits of floating point number precision).

In [16]:

```
import statsmodels.api as sm
import statsmodels.formula.api as smf

smf.ols("GrossConv ~ Treatment", full_data).fit().summary()
```

Out[16]:

OLS Regression Results

Dep. Variable:	GrossConv	**R-squared:**	0.051
Model:	OLS	**Adj. R-squared:**	0.030
Method:	Least Squares	**F-statistic:**	2.371
Date:	Tue, 09 Feb 2021	**Prob (F-statistic):**	0.131
Time:	09:24:56	**Log-Likelihood:**	77.613
No. Observations:	46	**AIC:**	-151.2
Df Residuals:	44	**BIC:**	-147.6
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	0.2204	0.010	23.084	0.000	0.201	0.240
Treatment	-0.0208	0.013	-1.540	0.131	-0.048	0.006

Omnibus:	6.181	**Durbin-Watson:**	0.677
Prob(Omnibus):	0.045	**Jarque-Bera (JB):**	6.094
Skew:	0.850	**Prob(JB):**	0.0475
Kurtosis:	2.460	**Cond. No.**	2.62

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [17]:

```
smf.ols("NetConv ~ Treatment", full_data).fit().summary()
```

Out[17]:

OLS Regression Results

Dep. Variable:	NetConv	**R-squared:**	0.007
Model:	OLS	**Adj. R-squared:**	-0.016
Method:	Least Squares	**F-statistic:**	0.2903

Date:	Tue, 09 Feb 2021	**Prob (F-statistic):**	0.593
Time:	09:24:56	**Log-Likelihood:**	95.810
No. Observations:	46	**AIC:**	-187.6
Df Residuals:	44	**BIC:**	-184.0
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	0.1183	0.006	18.403	0.000	0.105	0.131
Treatment	-0.0049	0.009	-0.539	0.593	-0.023	0.013

Omnibus:	0.968	**Durbin-Watson:**	1.165
Prob(Omnibus):	0.616	**Jarque-Bera (JB):**	0.985
Skew:	0.316	**Prob(JB):**	0.611
Kurtosis:	2.662	**Cond. No.**	2.62

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Summary : The linear regression gives p-values of GrossConv (0.131) and NetConv (0.593) are **exactly** the same with the results given by t-test.

Exercise 11

Now add indicator variables for the day of each observation. Do the standard errors on your treatment variable change? If so, in what direction?

You should have found that your standard errors decreased by about 20% – this is why, although just comparing means works, if you have additional variables you should add them as covariates in your analysis. Moreover, in other settings you may find this effect is even larger – the date indicators we added to our data are perfectly balanced between treatment and control, so we aren't adding a lot of data to the model by adding them as variables. As we'll see in later exercises, adding variables like "gender" or "age" (which will never be perfectly balanced across treatment and control) will help even more.

In [18]:
```
smf.ols("NetConv ~ Treatment + Date", full_data).fit().summary()
```

Out[18]:

OLS Regression Results

Dep. Variable:	NetConv	**R-squared:**	0.743
Model:	OLS	**Adj. R-squared:**	0.475
Method:	Least Squares	**F-statistic:**	2.770
Date:	Tue, 09 Feb 2021	**Prob (F-statistic):**	0.00991
Time:	09:24:56	**Log-Likelihood:**	126.94

No. Observations:	46	**AIC:**	-205.9
Df Residuals:	22	**BIC:**	-162.0
Df Model:	23		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.0782	0.016	4.885	0.000	0.045	0.111
Date[T.Timestamp('2017-10-12 00:00:00')]	0.0272	0.022	1.226	0.233	-0.019	0.073
Date[T.Timestamp('2017-10-13 00:00:00')]	0.0212	0.022	0.957	0.349	-0.025	0.067
Date[T.Timestamp('2017-10-14 00:00:00')]	0.0427	0.022	1.927	0.067	-0.003	0.089
Date[T.Timestamp('2017-10-15 00:00:00')]	0.0190	0.022	0.857	0.401	-0.027	0.065
Date[T.Timestamp('2017-10-16 00:00:00')]	0.0128	0.022	0.578	0.569	-0.033	0.059
Date[T.Timestamp('2017-10-17 00:00:00')]	0.0033	0.022	0.148	0.884	-0.043	0.049
Date[T.Timestamp('2017-10-18 00:00:00')]	0.0272	0.022	1.228	0.233	-0.019	0.073
Date[T.Timestamp('2017-10-19 00:00:00')]	0.0229	0.022	1.035	0.312	-0.023	0.069
Date[T.Timestamp('2017-10-20 00:00:00')]	0.0376	0.022	1.696	0.104	-0.008	0.084
Date[T.Timestamp('2017-10-21 00:00:00')]	0.0259	0.022	1.170	0.255	-0.020	0.072
Date[T.Timestamp('2017-10-22 00:00:00')]	0.0229	0.022	1.032	0.313	-0.023	0.069
Date[T.Timestamp('2017-10-23 00:00:00')]	0.0193	0.022	0.873	0.392	-0.027	0.065
Date[T.Timestamp('2017-10-24 00:00:00')]	0.0823	0.022	3.717	0.001	0.036	0.128
Date[T.Timestamp('2017-10-25 00:00:00')]	0.0774	0.022	3.495	0.002	0.031	0.123
Date[T.Timestamp('2017-10-26 00:00:00')]	0.0706	0.022	3.186	0.004	0.025	0.117
Date[T.Timestamp('2017-10-27 00:00:00')]	0.0831	0.022	3.752	0.001	0.037	0.129
Date[T.Timestamp('2017-10-28 00:00:00')]	0.0676	0.022	3.052	0.006	0.022	0.114
Date[T.Timestamp('2017-10-29 00:00:00')]	0.0485	0.022	2.188	0.040	0.003	0.094
Date[T.Timestamp('2017-10-30 00:00:00')]	0.0214	0.022	0.965	0.345	-0.025	0.067
Date[T.Timestamp('2017-10-31 00:00:00')]	0.0810	0.022	3.655	0.001	0.035	0.127
Date[T.Timestamp('2017-11-01 00:00:00')]	0.0645	0.022	2.911	0.008	0.019	0.110
Date[T.Timestamp('2017-11-02 00:00:00')]	0.0437	0.022	1.974	0.061	-0.002	0.090
Treatment	-0.0049	0.007	-0.750	0.461	-0.018	0.009

Omnibus:	4.114	**Durbin-Watson:**	2.921
Prob(Omnibus):	0.128	**Jarque-Bera (JB):**	1.796
Skew:	-0.000	**Prob(JB):**	0.407
Kurtosis:	2.032	**Cond. No.**	27.3

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

> Yes Treatment's standard error decrease about 20% from 0.009 to 0.007. And its p-value also decrase a little.

Exercise 12

Given your results, what would you tell Udacity about their trial?

> With the result, I would suggest to Udacity that it is better not to perform that modification because there is no statistically significant difference between control and treatment groups.

Exercise 13

As a last exercise, instead of adding indicators for each date, add indicators for day of the week (e.g. Monday, Tuesday, etc.).

(This is just for data manipulation practice!)

In [21]:
```
import calendar

data["DoW"] = [calendar.day_name[x.weekday() ] for x in data["Date"] ]
data.head()
```

Out[21]:

	Date	Pageviews	Clicks	Enrollments	Payments	Treatment	NetConv	GrossConv	CPT	
0	2017-10-11	7723	687	134.0	70.0	0	0.101892	0.195051	11.241630	W
0	2017-10-11	7716	686	105.0	34.0	1	0.049563	0.153061	11.247813	W
1	2017-10-12	9102	779	147.0	70.0	0	0.089859	0.188703	11.684211	
1	2017-10-12	9288	785	116.0	91.0	1	0.115924	0.147771	11.831847	
2	2017-10-13	10511	909	167.0	95.0	0	0.104510	0.183718	11.563256	

> The day of week is stored in **DoW** column.