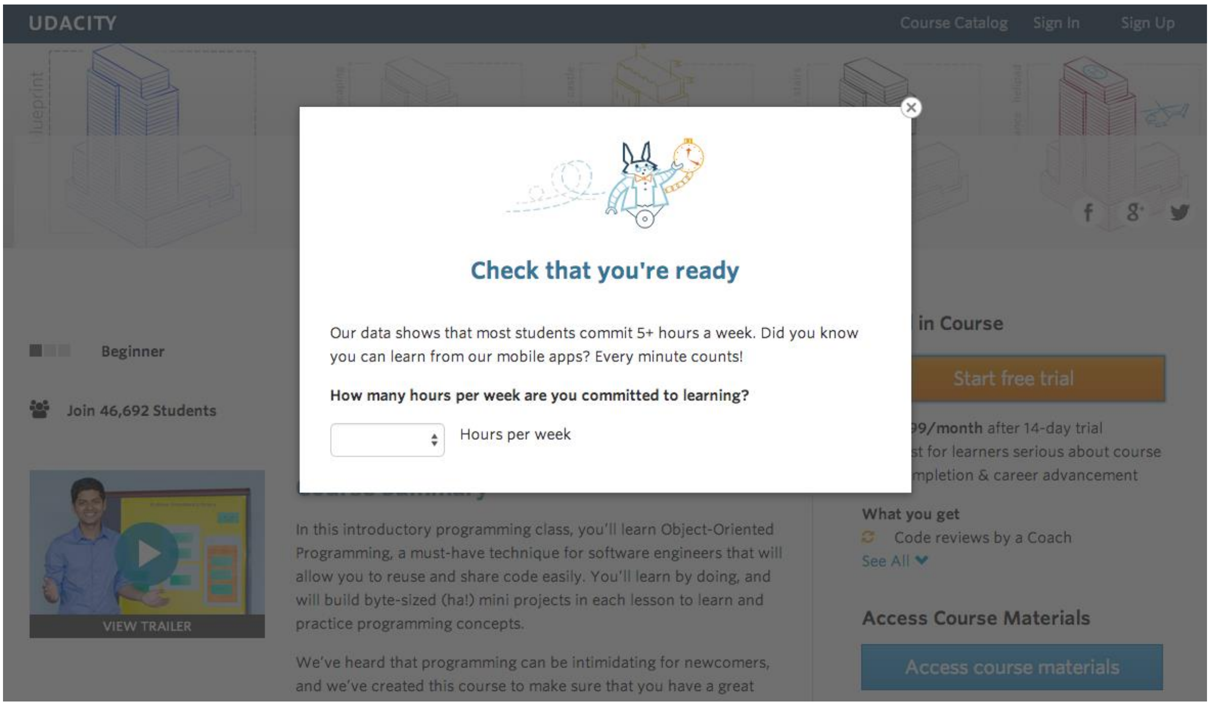
Udacity A/B testing course

Final project Report by Rishikesh Dhayarkar

Experiment Overview: Free Trial Screener

At the time of this experiment, Udacity courses currently have two options on the course overview page: "start free trial", and "access course materials". If the student clicks "start free trial", they will be asked to enter their credit card information, and then they will be enrolled in a free trial for the paid version of the course. After 14 days, they will automatically be charged unless they cancel first. If the student clicks "access course materials", they will be able to view the videos and take the quizzes for free, but they will not receive coaching support or a verified certificate, and they will not submit their final project for feedback.

In the experiment, Udacity tested a change where if the student clicked "start free trial", they were asked how much time they had available to devote to the course. If the student indicated 5 or more hours per week, they would be taken through the checkout process as usual. If they indicated fewer than 5 hours per week, a message would appear indicating that Udacity courses usually require a greater time commitment for successful completion, and suggesting that the student might like to access the course materials for free. At this point, the student would have the option to continue enrolling in the free trial, or access the course materials for free instead. This screenshot shows what the experiment looks like.



Objectives of this experiment:

- Lesser number dropouts from the free trail stage before the 14-day trial period.

- Improve overall student experience by providing a realistic view of course expectations
- Improve the capacity of coaches to support students who are likely to finish the course

Unit of diversion

The unit of diversion is a **cookie**, although if the student enrolls in the free trial, they are tracked by **user-id** from that point forward. The same user-id cannot enroll in the free trial twice. For users that do not enroll, their user-id is not tracked in the experiment, even if they were signed in when they visited the course overview page.

Choice of metric

These are some common metrics that one might consider while designing an A/B test to solve this problem.

1. **Number of cookies:** That is, number of unique cookies to view the course overview page

   The activity of viewing the course overview page is completely outside the area of experimentation. This metric remains independent of the experimental feature. Therefore, this can be used as an invariant metric to perform some sanity check about our experimental data.

2. **Number of user-ids:** That is, number of users who enroll in the free trial.

   This metric is responsive to the experiment but it does not explain if we are achieving the desired objective - reduction in dropout from free trail stage.

3. **Number of clicks:** That is, number of unique cookies to click the "Start free trial" button (which happens before the free trial screener is triggered).

   This another example of a metric that is independent of the experimental feature. Even this can be used an invariant metric.

4. **Click-through-probability:** That is, number of unique cookies to click the "Start free trial" button divided by number of unique cookies to view the course overview page.

   This metric operates outside the area of experimentation and can be used as an invariant metric.

5. **Gross conversion:** That is, number of user-ids to complete checkout and enroll in the free trial divided by number of unique cookies to click the "Start free trial" button.

   This metric has the same denominator for both control and experiment groups. The numerator for the control group will be higher than the numerator for the experiment group. If the feature works in the intended manner, the magnitude of gross conversion should reduce.

6. **Retention:** That is, number of user-ids to remain enrolled past the 14-day boundary (and thus make at least one payment) divided by number of user-ids to complete checkout.

This metric perfectly captures our objective – reduction in dropout numbers. Retention for control group would be low. This is because it includes people who do not satisfy the time commitment criterion. Such students might get frustrated and drop out of the course before the 14-day trial period. Retention for experimental group would be high because this group includes people who are aware of the time commitments and therefore do not dropout after the 14-day trial period.

7. **Net conversion:** That is, number of user-ids to remain enrolled past the 14-day boundary (and thus make at least one payment) divided by the number of unique cookies to click the "Start free trial" button.

   This metric is similar to Gross Conversion metric, the only difference is that here we consider users who made it past the 14-day period instead of users who only completed the checkout and enrolled in the free trail. The trend is similar to Gross Conversion metric.

Above metrics can be classified into 3 groups:

- **Invariant metrics** - Number of cookies, Number of clicks, Click-through-probability
- **Evaluation metrics** – Retention, Net conversion, Gross conversion
- **Not suitable metrics** - Number of user-ids

Invariant metrics

Metric name	**Formula**	**Symbol**	**d_min**
Number of cookies	# unique daily cookies on page	Ck	3000
Number of clicks	# unique daily cookies who clicked	Cl	240
Click-through-probability	Cl/Ck	CTP	0.01

Evaluation metrics

Metric name	**Formula**	**Symbol**	**d_min**
Retention	Paid/Enrolled	Retention	0.01
Net conversion	Paid/Cl	Net conversion	0.0075
Gross conversion	Enrolled/Cl	Gross conversion	0.01

Find standard deviation from the given baseline values

Metric	Description	Baseline Value
Number of cookies	Unique cookies to view course overview page per day:	40000
Number of clicks	Unique cookies to click "Start free trial" per day:	3200
Number of enrollments	Enrollments per day:	660
CTP	Click-through-probability on "Start free trial":	0.08
Gross Conversion	Probability of enrolling, given click:	0.20625
Retention	Probability of payment, given enroll:	0.53
Net Conversion	Probability of payment, given click	0.1093125

Find standard deviation (SD) for all evaluation metrics

These baseline values are for 40000 cookies, the question asks us to use 5000 cookies as our sample size. Scale all metrics for 5000 cookies.

Original baseline data for 40000 cookies

```
[1]
baseline = {"Cookies":40000,
            "Clicks":3200,
            "Enrollments":660,
            "CTP":0.08,
            "GConversion":0.20625,
            "Retention":0.53,
            "NConversion":0.109313}
```

Scaled baseline data for 5000 cookies

```
baseline["Cookies"] = 5000
baseline["Clicks"]=baseline["Clicks"]*(5000/40000)
baseline["Enrollments"]=baseline["Enrollments"]*(5000/40000)
baseline
```

```
{'CTP': 0.08,
 'Clicks': 400.0,
 'Cookies': 5000,
 'Enrollments': 82.5,
 'GConversion': 0.20625,
 'NConversion': 0.109313,
 'Retention': 0.53}
```

Next, calculate the standard deviation for all evaluation metrics. We have a binomial distribution for all evaluation metrics and we can approximate the binomial with a normal distribution since the sample size of 5000 is large enough.

SD for Gross Conversion – 0.0202

The analytical estimate is good enough because the unit of diversion (cookies) is the same as unit of analysis (cookies).

SD for Net Conversion – 0.0156

The analytical estimate is good enough because the unit of diversion (cookies) is the same as unit of analysis (cookies).

SD for Retention – 0.0549

The analytical estimate might not be good enough because the unit of diversion (cookies) is not the same as unit of analysis (enrolled, measured in user_id).

[6]
```python
import math as mt
```

```python
GC={}
GC["d_min"]=0.01
GC["p"]=baseline["GConversion"]
GC["n"]=baseline["Clicks"]
GC["sd"]=round(mt.sqrt((GC["p"]*(1-GC["p"]))/GC["n"]),4)
GC["sd"]
```

```
0.0202
```

[8]
```python
R={}
R["d_min"]=0.01
R["p"]=baseline["Retention"]
R["n"]=baseline["Enrollments"]
R["sd"]=round(mt.sqrt((R["p"]*(1-R["p"]))/R["n"]),4)
R["sd"]
```

```
0.0549
```

[9]
```python
NC={}
NC["d_min"]=0.0075
NC["p"]=baseline["NConversion"]
NC["n"]=baseline["Clicks"]
NC["sd"]=round(mt.sqrt((NC["p"]*(1-NC["p"]))/NC["n"]),4)
NC["sd"]
```

```
0.0156
```

Calculate the required sample size

Next step is to calculate the required sample for each of our evaluation metrics. That is, we need to find the required number of pageviews(cookies) for the control and experiment groups. The sample size is calculated using **Alpha** = 0.05(significance level) and **Beta** = 0.2 as parameters.

The equation for calculating the sample size ‘n’ is given by,

$$n = \frac{(Z_{1-\frac{\alpha}{2}} sd_1 + Z_{1-\beta} sd_2)^2}{d^2}$$, with:

$$sd_1 = \sqrt{p(1-p)+p(1-p)}$$

$$sd_2 = \sqrt{p(1-p)+(p+d)(1-(1-(p+d))}$$

Here **sd1** and **sd2** are standard deviations for the baseline conversion rate and the expected changed conversion rate. **‘p’** is the baseline conversion rate and **‘d’** is the minimum detectable change (d_min).

The following code snippet is an implementation of the above equations. **Scipy.stats** is used for calculating z-scores.

Calculate the required sample size

```
from scipy.stats import norm

def calc_sd1_and_sd2(p,d):
    sd1=mt.sqrt(2*(p*(1-p)))
    sd2=mt.sqrt((p*(1-p)) + (p+d)*(1-(p+d)))
    x=[sd1,sd2]
    return x

def calc_z_score(alpha):
    return norm.ppf(alpha)

def calc_sampSize(sds,alpha,beta,d):
    n = pow((calc_z_score(1-alpha/2) * sds[0] + calc_z_score(1-beta) * sds[1]), 2) / pow(d,2)
    return n
```

For Gross Conversion:

We need 25835 clicks get a minimum detectable difference of 0.01 for gross conversion metric.

```
GC["SampSize"]=round(calc_sampSize(calc_sd1_and_sd2(GC["p"],GC["d"]),0.05,0.2,GC["d"]))
GC["SampSize"]
```

```
25835
```

From the baseline dataset, we saw that 5000 cookies generate 400 enrollments, we have to find the number of cookies that can generate 25835 enrollments. Note -: 400/5000 = 0.08.

```
GC["SampSize"]=round(calc_sampSize(calc_sd1_and_sd2(GC["p"],GC["d"]),0.05,0.2,GC["d"]))
GC["SampSize"]=round(GC["SampSize"]/0.08*2)
print(f'Required sample size for gross conversion metric = {GC["SampSize"]}')
```

```
Required sample size for gross conversion metric = 645875
```

For Net Conversion:

```
[59] R["SampSize"]=round(calc_sampSize(calc_sd1_and_sd2(R["p"],R["d"]),0.05,0.2,R["d"]))
     R["SampSize"]=R["SampSize"]/0.08/0.20625*2
     print(f'Required sample size for retention metric = {R["SampSize"]}')
```

```
Required sample size for retention metric = 4737818.181818182
```

For Retention:

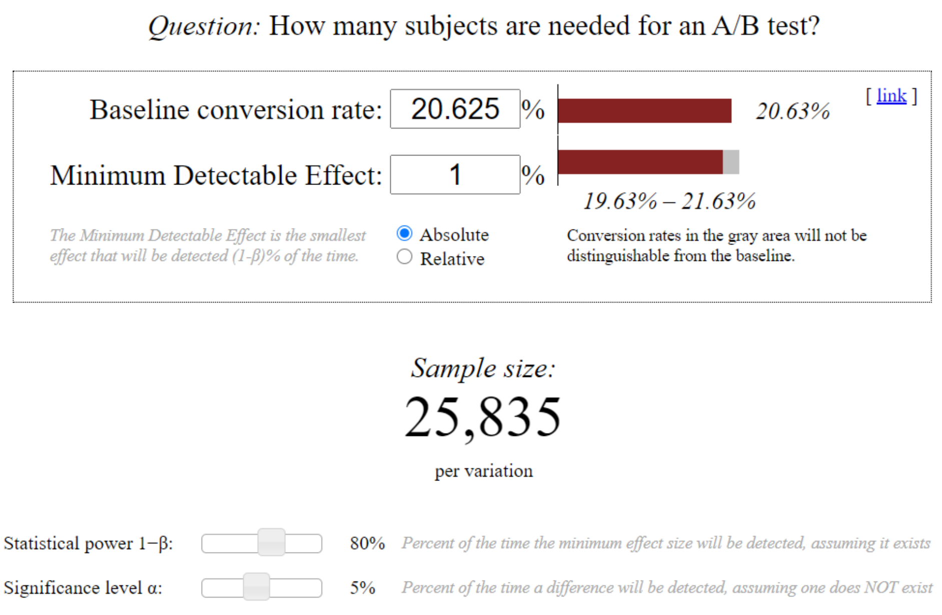
```
[60] NC["SampSize"]=round(calc_sampSize(calc_sd1_and_sd2(NC["p"],NC["d"]),0.05,0.2,NC["d"]))
     NC["SampSize"]=NC["SampSize"]/0.08*2
     print(f'Required sample size for net conversion metric = {NC["SampSize"]}')
```

```
Required sample size for net conversion metric = 685325.0
```

Sample sizes can also be calculated using this tool (https://www.evanmiller.org/ab-testing/sample-size.html). The results are approximately the same.

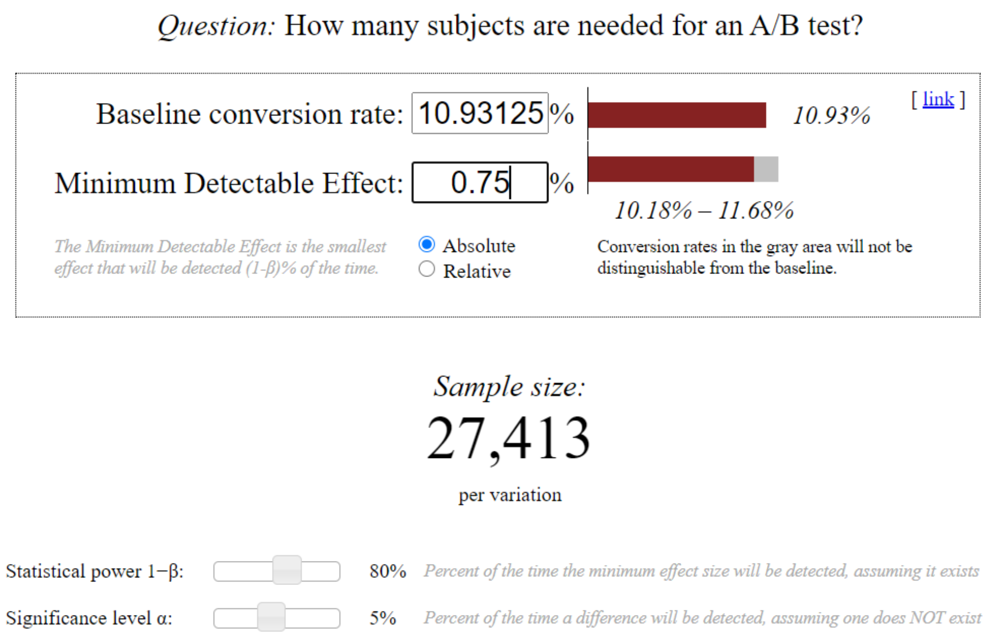
For gross conversion:

- Baseline Conversion: 20.625%
- Minimum Detectable Effect: 1%
- Sample Size = 25,835 enrollments/group
- Total sample size = (25,835 *2) = 51,670 enrollments
- Enrollments or clicks /Pageview: 400/5000 = 0.08
- Pageviews Required = 645,875



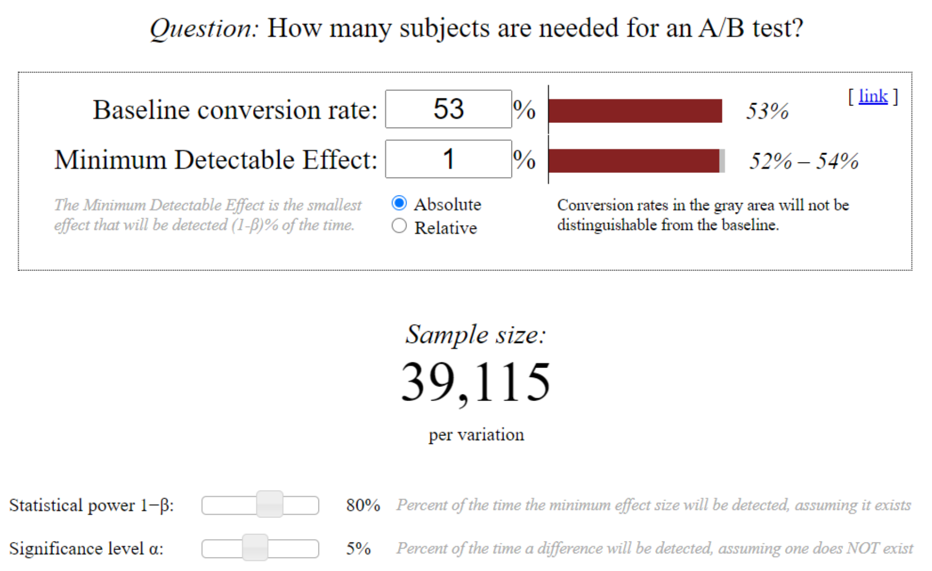
For net conversion:

- Baseline Conversion: 10.93125%
- Minimum Detectable Effect: 0.75%
- Sample size = 27,413 enrollments/group
- Total sample size = 54,826 enrollments
- Enrollments or clicks /pageview: 400/5000 = 0.08
- Pageviews Required = 685,325



For retention:

- Baseline Conversion: 53%
- Minimum Detectable Effect: 1%
- Sample size = 39,115 enrollments/group
- Total sample size = 78,230 enrollments
- Enrollments or clicks/pageview: 82.5/5000 = 0.0165
- Pageviews Required = 78,230/0.0165 = 4,741,212



Sanity checks on the collected data

The data for you to analyze is here. This data contains the raw information needed to compute the above metrics, broken down day by day. Note that there are two sheets within the spreadsheet - one for the experiment group, and one for the control group.
The meaning of each column is:

- **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. (Note that the date for this column is the start date, that is, the date of enrollment, rather than the date of the payment. The payment happened 14 days later. Because of this, the enrollments and payments are tracked for 14 fewer days than the other columns.)

The first step in analysing this dataset is to perform sanity checks on the collected values. These checks verify if the collected data was not affected by unseen or unknown factors/variables. Invariant metrics are used to perform these checks.

We have three invariant metrics,

- Number of Cookies in Course Overview Page
- Number of Clicks on Free Trial Button
- Free Trial button Click-Through-Probability

Number of Cookies in Course Overview Page

If the experiment was performed correctly, this metric should result in approximately the same value for both the groups. Let's count the number of cookies diverted into each of the groups and see if there is a significant difference. If there is a significant difference then this metric is biased towards one of the groups and therefore, we can conclude that there was some unseen factor which was not accounted during the data collection process.

```
pageviews_cont = df_control['Pageviews'].sum()
pageviews_exp = df_expt['Pageviews'].sum()
total = pageviews_cont + pageviews_exp
print("number of pageviews in control group:", pageviews_cont)
print("number of Pageviewsin experiment group:" ,pageviews_exp)
```

```
number of pageviews in control group: 345543.0
number of Pageviewsin experiment group: 344660
```

The total number of pageviews collected in both the groups is almost equal. To confirm this, we need to generate a confidence interval for the total number of pageviews. Ideally the sum of pageviews in both the groups should be exactly the same, i.e exactly 50% of datapoints should fall in one of the groups and the other 50% should fall into the remaining one.

Each data point gets diverted into one of two buckets – control or experiment. So, we are performing a bunch of Bernoulli trails, each trail results in a binomial distribution. As the number of data points increase binomial distribution can be approximated with a normal distribution (central limit theorem).

We wish to check if the observed number of samples in one group divided by the total number of samples in both the group is very close to 0.5. We need to build a confidence interval around 0.5. To build a confidence interval we first need to calculate the standard deviation.

First let's consider the control group,

```
[23]
p_hat = round(pageviews_cont/(total), 4)
p_hat
```

```
0.5006
```

So, p_hat for control group is 0.5006. For the sanity check to hold true this value must be within the confidence interval.

```python
p = 0.5
alpha = 0.05

p_hat = round(pageviews_cont/(pageviews_total), 4)
sd = mt.sqrt((p*(1-p))/(pageviews_total))
margin = round(calc_z_score(1-(alpha/2))*sd,4)
print("Confidence interval for pageviews in the control group is between",\
      p - margin,"and",p + margin)
```

```
Confidence interval for pageviews in the control group is between 0.4988 and 0.5012
```

We can see that 0.5006 is in fact present within the confidence interval. This tells us that pageview counts in the experiment were diverted into the right buckets without any bias. The same can be done for the experiment group.

[26]

```python
p_hat = round(pageviews_exp/(total), 4)
p_hat
```

```
0.4994
```

```python
p = 0.5
alpha = 0.05

p_hat = round(pageviews_exp/(pageviews_total), 4)
sd = mt.sqrt((p*(1-p))/(pageviews_total))
margin = round(calc_z_score(1-(alpha/2))*sd,4)
print("Confidence interval for pageviews in the experiment group is between",\
      p - margin,"and",p + margin)
```

```
Confidence interval for pageviews in the experiment group is between 0.4988 and 0.5012
```

0.4994 lies within the confidence interval 0.4988 to 0.5012.

We have to repeat the same procedure for the other two invariant metrics.

Number of Clicks on Free Trial Button

[94]

```python
clicks_cont = df_control['Clicks'].sum()
clicks_exp = df_expt['Clicks'].sum()
clicks_total = clicks_cont + clicks_exp

p_hat = round(clicks_cont/clicks_total,4)
sd = mt.sqrt(p*(1-p)/clicks_total)
margin = round(calc_z_score(1-(alpha/2))*sd,4)
print("The confidence interval is between",p-margin,"and",p+margin)
```

```
The confidence interval is between 0.4959 and 0.5041
```

```python
p_hat = round(clicks_cont/clicks_total,4)
p_hat
```

```
0.5005
```

```
clicks_cont = df_control['Clicks'].sum()
clicks_exp = df_expt['Clicks'].sum()
clicks_total = clicks_cont + clicks_exp

sd = mt.sqrt(p*(1-p)/clicks_total)
margin = round(calc_z_score(1-(alpha/2))*sd,4)
print("The confidence interval is between",p-margin,"and",p+margin)
```

```
The confidence interval is between 0.4959 and 0.5041
```

```
[99]
p_hat = round(clicks_exp/clicks_total,4)
p_hat
```

```
0.4995
```

Following the procedure mentioned above, we can conclude that even the second invariant metric is unbiased.

Now for the last invariant metric,

Click-through-probability of the Free Trial Button

The only difference here is we are comparing two ratios instead of counts, the ratio of clicks to pageviews. Since clicking on the free trail button is outside the experimental feature, we expect this metric to be the same for both the groups.

```
[32]
d_expected = 0

ctp_cont = clicks_cont/pageviews_cont
ctp_exp = clicks_exp/pageviews_exp
p_pooled = clicks_total/total
sd_pooled = mt.sqrt(p_pooled*(1-p_pooled)*(1/pageviews_cont+1/pageviews_exp))
margin = round(calc_z_score(1-(alpha/2))*sd_pooled,4)
print("The confidence interval is between",d_expected-margin,"and",d_expected+margin)
```

```
The confidence interval is between -0.0013 and 0.0013
```

```
[33]
d_hat = round(ctp_exp-ctp_cont,4)
d_hat
```

```
0.0001
```

The difference between both the ratios is within the confidence interval. Therefore, even this metric is unbiased.

At this point all three of our invariant metrics have passed the sanity check to prove that the data collection procedure was unbiased and without any unseen variables.

Examining effect size

Now that all our invariant metrics have passed the sanity check, the next, step is to check if the evaluation metrics show significant difference between the two groups.

In other words, we need to check if the evaluation metrics show a **statistically significant** and a **practically significant** (the difference is big enough to be beneficial to the company) difference between the two groups.

A metric is statistically significant if the confidence interval does not include 0 (that is, you can be confident there was a change), and it is practically significant if the confidence interval does not include the practical significance boundary.

Note: We need to consider only the first 23 rows of our dataset because there are no observations recorded for the remaining days. This is because the payment happened 14 days later. Because of this, the enrollments and payments are tracked for 14 fewer days than the other columns.

For gross conversion metric

Calculate the pooled standard deviation, critical z-score, and the confidence interval.

[129]

```
clicks_cont = df_control['Clicks'][:23].sum()
clicks_exp = df_expt['Clicks'][:23].sum()
```

```
enrollments_cont = df_control["Enrollments"].sum()
enrollments_exp = df_expt["Enrollments"].sum()

GC_cont = enrollments_cont/clicks_cont
GC_exp = enrollments_exp/clicks_exp

GC_pooled = (enrollments_cont + enrollments_exp)/(clicks_cont + clicks_exp)
GC_sd_pooled = mt.sqrt((GC_pooled*(1-GC_pooled))*((1/clicks_cont) + (1/clicks_exp)))

GC_ME = round(calc_z_score(1-alpha/2)*GC_sd_pooled, 4)
GC_diff = round(GC_exp-GC_cont, 4)

print("The change due to the experiment is",GC_diff*100,"%")
print("Confidence Interval: [",round(GC_diff-GC_ME, 4),",",round(GC_diff+GC_ME, 4),"]")
```

```
The change due to the experiment is -2.06 %
Confidence Interval: [ -0.0292 , -0.012 ]
```

We can see that gross conversion reduced by almost 2%. This indicates that the experiment did work as expected. The result is statistically significant because the confidence interval does not contain zero. And it's practically significant because the entire confidence interval is more than the minimum detectable change (d_min) of 1%.

For Net conversion

For net conversion the confidence interval does include zero and it includes the d_min of 0.0075, which indicates that the difference is neither statistically significant not practically significant.

```python
payments_cont=df_control["Payments"].sum()
payments_exp=df_expt["Payments"].sum()

NC_cont=payments_cont/clicks_cont
NC_exp=payments_exp/clicks_exp

NC_pooled=(payments_cont+payments_exp)/(clicks_cont+clicks_exp)
NC_sd_pooled=mt.sqrt(NC_pooled*(1-NC_pooled)*(1/clicks_cont+1/clicks_exp))

NC_ME=round(calc_z_score(1-alpha/2)*NC_sd_pooled,4)
NC_diff=round(NC_exp-NC_cont,4)

print("The change due to the experiment is",NC_diff*100,"%")
print("Confidence Interval: [",round(NC_diff-NC_ME, 4),",",round(NC_diff+NC_ME,4),"]")
```

```
The change due to the experiment is -0.49 %
Confidence Interval: [ -0.0116 , 0.0018 ]
```

Verifying our results with sign tests

Sign tests involve counting the number of times the desired pattern is observed in the dataset. For gross and net conversion metrics.

Based on the way we have defined gross and net conversion metrics we expect our experiment to decrease both metrics. To perform sign tests, we need to count the number of days on which decrease in gross and net conversion was observed.

For easier data manipulation let's join the experimental and control datasets.

```python
[43]
full = df_control.join(other=df_expt,how="inner",lsuffix="_cont",rsuffix="_exp")
full=full.loc[full["Enrollments_cont"].notnull()]
full.count()
```

```
Date_cont          23
Pageviews_cont     23
Clicks_cont        23
Enrollments_cont   23
Payments_cont      23
Date_exp           23
Pageviews_exp      23
Clicks_exp         23
Enrollments_exp    23
Payments_exp       23
dtype: int64
```

Divide total enrollments per day by clicks per day to get gross conversion per day. To get net conversion per day divide total payments by clicks per day.

```python
import numpy as np

x=full['Enrollments_cont']/full['Clicks_cont']
y=full['Enrollments_exp']/full['Clicks_exp']
full['GC'] = np.where(x>y,1,0)

z=full['Payments_cont']/full['Clicks_cont']
w=full['Payments_exp']/full['Clicks_exp']
full['NC'] = np.where(z>w,1,0)
full.head()
```

	Date_cont	Pageviews_cont	Clicks_cont	Enrollments_cont	Payments_cont	Date_exp	Pageviews_exp	Clicks_exp	Enrollments_exp	Payments_exp	GC	NC
0	Sat, Oct 11	7723.0	687.0	134.0	70.0	Sat, Oct 11	7716	686	105.0	34.0	1	1
1	Sun, Oct 12	9102.0	779.0	147.0	70.0	Sun, Oct 12	9288	785	116.0	91.0	1	0
2	Mon, Oct 13	10511.0	909.0	167.0	95.0	Mon, Oct 13	10480	884	145.0	79.0	1	1
3	Tue, Oct 14	9871.0	836.0	156.0	105.0	Tue, Oct 14	9867	827	138.0	92.0	1	1
4	Wed, Oct 15	10014.0	837.0	163.0	64.0	Wed, Oct 15	9793	832	140.0	94.0	1	0

New columns ‘GC’ and ‘NC’ are populated with binary values to indicate the days on which the gross and net conversion were lower in the experiment group.

Count all occurrences where gross conversion and net conversion were lower in the experiment than the values in the control group.

```python
GC_x=full.GC[full["GC"]==1].count()
NC_x=full.NC[full["NC"]==1].count()
n=full.NC.count()
print("No. of cases for GC:",GC_x,'\n',
      "No. of cases for NC:",NC_x,'\n',
      "No. of total cases",n)
```

```
No. of cases for GC: 19
 No. of cases for NC: 13
 No. of total cases 23
```

So, the experiment worked on 19 days w.r.t to GC metric and it worked on 13 days w.r.t to NC metric out of a total 23 days.

Now, we need to check how likely it is for us to observe these successes. To do this conduct another hypothesis test.

Null hypothesis: no difference between metrics of the two groups

Alternate hypothesis: metrics from the experiment group are lower than the metrics of control group

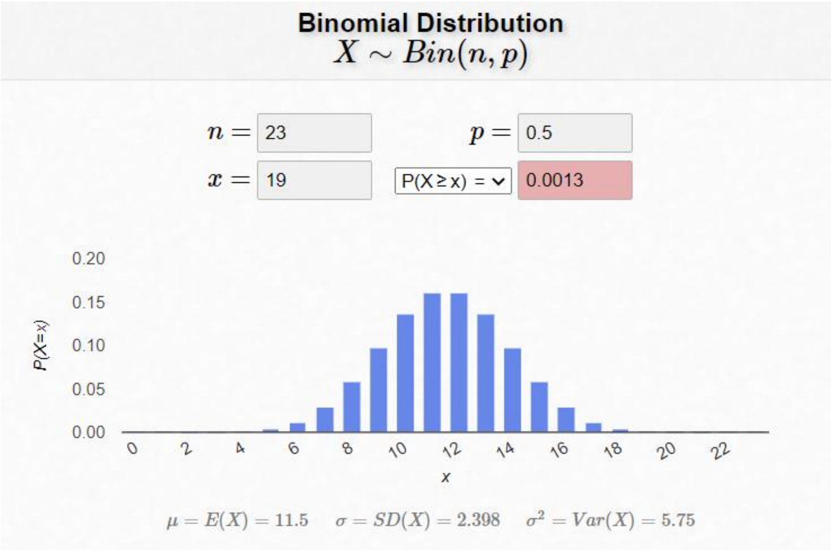
Let’s define a random variable ‘S’ – total number of successes. Number of successes can range from 0 to 23 (total days).

For GC:

We need to find P(S>=19) by using the binomial distribution. We consider a binomial distribution with n=23 trails, p=0.5, x=19.

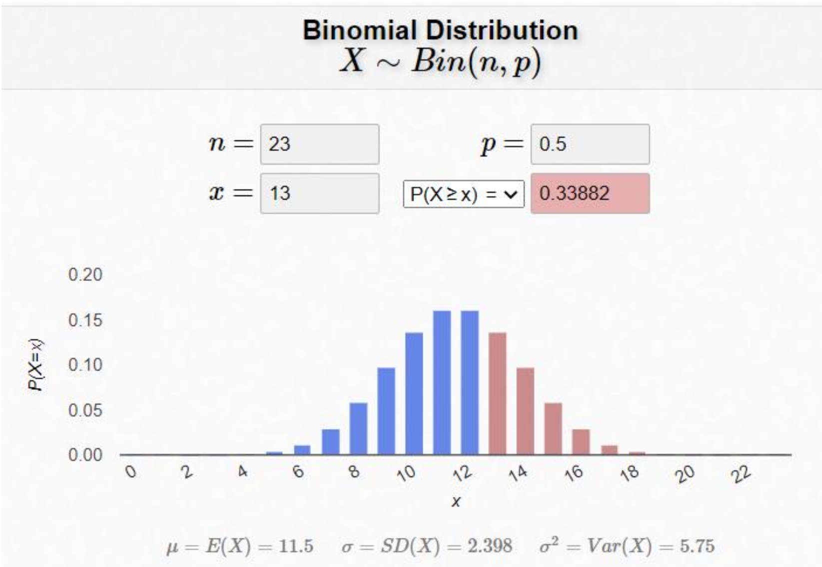
- p=0.5 for random assignment case for two outcomes, 0 or 1
- x is the number of successes

We get P(S>=19) as 0.0013 which is less than the significance value of 0.05. Reject null hypothesis. This implies, the probability of 19 or more successes is very unlikely which means the experiment worked as desired.



For NC:

We need to find P(S>=13) by using the binomial distribution. We consider a binomial distribution with n=23 trails, p=0.5, x=13.



We get P(S>=13) as 0.3388 which is greater than the significance value of 0.05. Fail to reject null hypothesis. This implies, the probability of 13 or more successes is very likely which means the experiment did not work as desired.

From the sign tests we get the same conclusions as we got from our effect size calculation: the change in Gross conversion was indeed significant, while the change in Net conversion was not.

Conclusion and recommendation

This experiment was designed to determine if filtering students as a function of study time commitment would improve the overall student experience and the coaches' capacity to support students who are likely to complete the course, without significantly reducing the number of students who continue past the free trial.

A statistically and practically significant decrease in Gross Conversion was observed from the results of both sign test and effect size calculation test. But there was no significant difference observed in Net Conversion for either of the tests.

This shows that decrease in enrollment is not strongly associated to increase in the number of students staying for the requisite 14 days such that the at least one payment gets triggered. Based on this I would recommend not to launch the feature at this stage. More tests and data collection should be pursued.