AB_Testing

July 1, 2019

Udacity A/B Testing Project Hannah Hon

1. About Udacity's A/B Testing Course
2. Experiment Overview
3. Metric Choice

- 3.1 Invariate Metrics - Sanity Checks
- 3.2 Evaluation Metrics - Performance Indicators

4. Estimating the baseline values of metrics

- 4.1 Collecting estimators data
- 4.2 Estimating Standard Deviation

5. Experiment Sizing

- 5.1 Get Z-score critical value and Standard Deviations
- 5.2 Calculate Sample Size per Metric

6. Analyzing Collected Data

- 6.1 Loading collected data
- 6.2 Sanity Checks
- 6.3 Measuring effect size
- 6.4 Double check with Sign Tests

7. Conclusions & Recommendations

About Udacity's A/B Testing Course Udacity's A/B testing course is presented by Google and focuses on the design and analysis of A/B testing. This course included ethics and policy, choosing and characterizing metrics, designing experiment and analyzing results.

Experiment Overview

1. The experiment: Free Trial Screener
2. Udacity is a online teaching plarform with main business goal of maximizing course completion by students.
3. Current Condition:

- 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.

4. Experiment Condition:
    - 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.
5. Hypothesis:
   The hypothesis was that this might set clearer expectations for students upfront, thus reducing the number of frustrated students who left the free trial because they didn't have enough time—without significantly reducing the number of students to continue past the free trial and eventually complete the course. If this hypothesis held true, Udacity could improve the overall student experience and improve coaches' capacity to support students who are likely to complete the course.
6. 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.

Metric Choice We need two types of metrics for our experiment. The first one is **invariant metrics**, which is used for 'sanity checks'. This means that when we conduct the experiment (The way we present change to population and collect data) is not inherently wrong, we pick metrics we consider not to be affected by the experiment and later there is not significant difference between the results between the control and experiment groups. The second one is **evaluation metrics** , which are the metrics we expect to see changes and are related to the business goal. For each metric we have a Dmin, which is the minimal change that have practial significance to the business. A 5% Dmin means that any change that in the metrics that is less than 5% doesn't have practical meaning to the business.

1. Invariant metrics

- **Number of cookies**: That is, number of unique cookies to view the course overview page. (dmin=3000)
    - This metric will not be affected by the experiment. Since the number of cookies should be similar between experiment and control group, since they are randomly assigned.
- **Number of clicks**: That is, number of unique cookies to click the "Start free trial" button (which happens before the free trial screener is trigger). (dmin=240)

  - This metric should not be affected by the experiment, because at the point of clicking the button the experiment hasn't began yet.

- **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. (dmin=0.01)
  - This metric shouldn't be affected, because at the point of clicking the button the experiment hasn't began.

2. Evaluation metrics

- **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. (dmin= 0.01) The conversion rate is expected to decrease since the number of enrollment is expected to decrease.
- **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. (dmin=0.01) We expect the number of enrollment reduces and the number of payments remains the same, so this value will be higher for the experiment group.
- **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. (dmin= 0.0075) The number of clicks is expected to remain the same and the number of remain enrolled will approximately the same, so this metrics should be insignificant.

Estimating the baseline values Unique cookies to view course overview page per day: 40000
Unique cookies to click "Start free trial" per day: 3200
Enrollments per day: 660
Click-through-probability on "Start free trial": 0.08
Probability of enrolling, given click: 0.20625
Probability of payment, given enroll: 0.53
Probability of payment, given click 0.1093125

Calculating the standard deviation The standard deviation here is the standard deviation of the sampling distribution, which means standard error. The standard error of the current metrics depend on the sample size, and can be computed anlytically or empirically. The sample size of 5000 'cookies' visit the course view page. The baseline 'clicks' is 3200 * (5000/40000) = 400. The baseline enrollment is 660 * (5000/40000) = 82.5.

1. Gross conversion
   The baseline probability of gross conversion can be calculated by the number of users to enroll in a free trial divided by the number of cookies clicking the free trial, which means the probability of enrollment given a click. In this case, the unit of diversion(cookies), which we use to differentiate samples and assign them to control and experiment groups, is equal to the unit of analysis, that is the denominator of gross conversion to calculate Gross Conversion.

```
In [88]: import math as mt
         import numpy as np
```

```
import pandas as pd
from scipy.stats import norm
```

```
In [58]: #Let's place this estimators into a dictionary for ease of use later
         baseline = {"Cookies":40000,"Clicks":3200,"Enrollments":660,"CTP":0.08,"GConversion":(
                     "Retention":0.53,"NConversion":0.109313}
```

```
In [59]: #Scale The counts estimates
         baseline["Cookies"] = 5000
         baseline["Clicks"]=baseline["Clicks"]*(5000/40000)
         baseline["Enrollments"]=baseline["Enrollments"]*(5000/40000)
         baseline
```

```
Out[59]: {'Cookies': 5000,
          'Clicks': 400.0,
          'Enrollments': 82.5,
          'CTP': 0.08,
          'GConversion': 0.20625,
          'Retention': 0.53,
          'NConversion': 0.109313}
```

```
In [60]: GC={}
         GC["d_min"]=0.01
         GC["p"]=baseline["GConversion"]
         #p is given in this case - or we could calculate it from enrollments/clicks
         GC["n"]=baseline["Clicks"]
         GC["sd"]=round(mt.sqrt((GC["p"]*(1-GC["p"]))/GC["n"]),4)
         GC["sd"]
```

```
Out[60]: 0.0202
```

2. Retention
   The baseline probability for retention is the number of paying users(enrolled after 14 days) divided by number of total users, which means the probability of payment given enrollment. The sample size is the number of enrolled users. In this case, unit of diversion is not equal to the unit of analysis, so analytically estimate is not enough. Should use empirical estimate with the data we have.

```
In [61]: 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"]
```

```
Out[61]: 0.0549
```

3. Net conversion
   The baseline net conversion is number of user-id remained after the 14 day free trial divided by the number of cookies clicking the free trial, which means the probability of payment given a click. The sample size is the number of cookies that clicked. The unit of analysis and unit of diversion are equal to what we expect and analytically estimate is enough.

```
In [27]: 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"]
```

```
Out[27]: 0.0156
```

Calcuating sample size So now we know the metrics in the baseline, most importantly their variance. We can calculate their number of sample size we need for our experiment so our experiment will have enough statistical power and significance.

Given alpha = 0.05 and beta = 0.2(power), we need to estimate how many total pageviews(cookies that view the course overview pages) are required for our experiment. Then we will assign these pageviews to control and experiment group.

To calculate the minimum sample size for control and experiment groups, which provide the probability of Type I Error alpha, Power 1- beta, detectable effect d and baseline conversion rate of p.

H0: Pcont = Pexp, Ha: Pcont != Pexp is

n=(Z1/2sd1+Z1sd2)$^{2/d}$2

sd1=sqrt(p(1p)+p(1p))

sd2=sqrt(p(1p)+(p+d)(1(p+d))

Type I error: = 0.05 Power: (1-)

Detectable change: Dmin = Baseline conversion rate: P

What we need to calculate:

1. Z scores for 1- /2 and for 1- 2. Get the two standard deviations, that is for both the baseline and for expected changed rate.

```
In [28]: #Inputs: required alpha value (alpha should already fit the required test)
         #Returns: z-score for given alpha
         def get_z_score(alpha):
             return norm.ppf(alpha)

         # Inputs p-baseline conversion rate which is our estimated p and d-minimum detectable
         def get_sds(p,d):
             sd1=mt.sqrt(2*p*(1-p))
             sd2=mt.sqrt(p*(1-p)+(p+d)*(1-(p+d)))
             sds=[sd1,sd2]
             return sds

         # Inputs:sd1-sd for the baseline,sd2-sd for the expected change,alpha,beta,d-d_min,p-
         # Returns: the minimum sample size required per group according to metric denominator
         def get_sampSize(sds,alpha,beta,d):
             n=pow((get_z_score(1-alpha/2)*sds[0]+get_z_score(1-beta)*sds[1]),2)/pow(d,2)
             return n
```

Gross Conversion

```
In [29]: # Let's get an integer value for simplicity
         GC["SampSize"]=round(get_sampSize(get_sds(GC["p"],GC["d_min"]),0.05,0.2,GC["d_min"]))
         GC["SampSize"]
```

```
Out[29]: 25835.0

In [30]: GC

Out[30]: {'d_min': 0.01, 'p': 0.20625, 'n': 400.0, 'sd': 0.0202, 'SampSize': 25835.0}
```

This means we need at least 25,835 cookies who click the Free Trial button - per group! That means that if we got 400 clicks out of 5000 pageviews (400/5000 = 0.08) -> So, we are going to need GC["SampSize"]/0.08 = 322,938 pageviews, again ; per group! Finally, the total amount of samples per the Gross Conversion metric is:

```
In [31]: GC["SampSize"]=round(GC["SampSize"]/0.08*2)
         GC["SampSize"]

Out[31]: 645875.0
```

Retention

```
In [36]: # Getting a nice integer value
         R["SampSize"]=round(get_sampSize(get_sds(R["p"],R["d_min"]),0.05,0.2,R["d_min"]))
         R["SampSize"]

Out[36]: 39087.0
```

This means that we need 39,087 users who enrolled per group! We have to first convert this to cookies who clicked, and then to cookies who viewed the page, then finally to multiply by two for both groups.

```
In [37]: R["SampSize"]=round((R["SampSize"]/0.08/0.20625*2),2)
         R["SampSize"]

Out[37]: 4737818.18
```

This takes us as high as over 4 million page views total, this is practically impossible because we know we get about 40,000 a day, this would take well over 100 days. This means we have to drop this metric and not continue to work with it because results from our experiment (which is much smaller) will be biased.

Net Conversion

```
In [38]: # Getting a nice integer value
         NC["SampSize"]=round(get_sampSize(get_sds(NC["p"],NC["d_min"]),0.05,0.2,NC["d_min"]))
         NC["SampSize"]

Out[38]: 27413.0
```

So, needing 27,413 cookies who click per group takes us all the way up to:

```
In [39]: NC["SampSize"]=NC["SampSize"]/0.08*2
         NC["SampSize"]

Out[39]: 685325.0
```

We are all the way up to 685,325 cookies who view the page. This is more than what was needed for Gross Conversion, so this will be our number. Assuming we take 80% of each days pageviews, the data collection period for this experiment (the period in which the experiment is revealed) will be about 3 weeks.

Analyzing Collected Data Finally, the moment we've all been waiting for, after so much preparation we finally get to see what this experiment will prove! The data is presented as two spreadsheets. I will load each spreadshot into a pandas dataframe.

1. Loading collected data

```
In [40]: control = pd.read_csv('Final Project Results - Control.csv')
         experiment = pd.read_csv('Final Project Results - Experiment.csv')
         control.head()
```

Out[40]:

	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

2 Sanity Checks

First thing we have to do before even beginning to analyze this experiment's results is sanity checks. These checks help verify that the experiment was conducted as expected and that other factors did not influence the data which we collected. This also makes sure that data collection was correct.

We have 3 Invariant metrics::

Number of Cookies in Course Overview Page Number of Clicks on Free Trial Button Free Trial button Click-Through-Probability Two of these metrics are simple counts like number of cookies or number of clicks and the third is a probability (CTP). We will use two different ways of checking whether these obsereved values are like we expect (if in fact the experiment was not damaged.

Sanity Checks for differences between counts Number of cookies who viewed the course overview page - Starting from this simpel invariant metric, we want to count the total amount of cookie pageviews we diverted to each group and see if there is a significant difference int he amount of cookies. A significant difference will imply a biased experiment that we should not rely on it's results.

```
In [41]: pageviews_cont=control['Pageviews'].sum()
         pageviews_exp=experiment['Pageviews'].sum()
         pageviews_total=pageviews_cont+pageviews_exp
         print ("number of pageviews in control:", pageviews_cont)
         print ("number of Pageviews in experiment:" ,pageviews_exp)
```

```
number of pageviews in control: 345543
number of Pageviewsin experiment: 344660
```

We want to make sure this two numbers are not statistically different:
We expect the number of pageviews in control is 50% of the total pageviews of both groups. To get the distribution, I'm going to use a binomial distribution. A binomial random variable will be the number of successes we can expect to get out of N experiments, given the probabilities of a single success.

We can approximate our binomial distribution with normal distribution, using central limit theorem. The mean of our distribution is p and standard deviation is sqrt(p(1-p)/n). We want to know whether our observed p is significantlly different tha p = 0.5. We should calculate the margin of error acceptable at a 95% confidence level.

ME = Z(1-alpha/2) *sd

The confidence interval is CI = (p_hat - ME, p_hat + ME)

```
In [52]: p = 0.5
         alpha = 0.05
         p_hat = round(pageviews_cont/(pageviews_total),4)
         sd=mt.sqrt(p*(1-p)/(pageviews_total))
         ME = norm.ppf(1-alpha/2)*sd
         print(p_hat)
         print('The confidence interval is between:', round(p_hat - ME,4) ,'and', round(p_hat +
```

```
0.5006
The confidence interval is between: 0.4994 and 0.5018
```

From the result we can see that 0.5006 is inside the range os 0.4994 and 0.5018, which means that there is no significant difference in pageviews between control and experiment groups.

Number of cookies who clicked the Free Trial Button

```
In [56]: clicks_cont = control['Clicks'].sum()
         clicks_exp = experiment['Clicks'].sum()
         clicks_total = clicks_cont + clicks_exp
         p_hat=round(clicks_cont/clicks_total,4)
         sd=mt.sqrt(p*(1-p)/clicks_total)
         ME=round(get_z_score(1-(alpha/2))*sd,4)
         print(p_hat)
         print ("The confidence interval is between",p-ME,"and",p+ME)
```

```
0.5005
The confidence interval is between 0.4959 and 0.5041
```

From the result we can see that 0.5005 is inside the range os 0.4959 and 0.5041, which means that there is no significant difference in clicks between control and experiment groups.

Click-through-probability of the Free Trial Button In this case, we want to make sure the proportion of clicks given a pageview (our observed CTP) is about the same in both groups (since this was not expected to change due to the experiment). In order to check this out we will calculate the CTP in each group and calculate a confidence interval for the expected difference between them.

We expect to see no difference between between CTPcont and CTPexp, with an acceptable margin of error, dictated by our confidence interval. The changes we should notice are for the calculation of the standard error - which in this case is a pooled standard error.

SDpool = sqrt(P_pool_hat(1-p_pool_hat(1/Ncont + 1/Nexp)))

Ppool = (Xcont + Xexp)/(Ncont + N exp)

```
In [69]: ctp_cont=clicks_cont/pageviews_cont
         ctp_exp=clicks_exp/pageviews_exp
         d_hat=round(ctp_exp-ctp_cont,4)
         p_pooled=clicks_total/pageviews_total
         sd_pooled=mt.sqrt(p_pooled*(1-p_pooled)*(1/pageviews_cont+1/pageviews_exp))
         ME=round(get_z_score(1-(alpha/2))*sd_pooled,4)
         print(round(d_hat,4))
         print ("The confidence interval is between",0-ME,"and",0+ME)

0.0001
The confidence interval is between -0.0013 and 0.0013
```

From the result we can see that 0.0001 is inside the range os -0.0013 and 0.0013, which means that there is no significant difference in pageviews between control and experiment groups.

Examining effect size Then we should look at the changes between the control and experiment groups according to our evaluation metrics to make sure there is a difference, which means that there is statistically significant and most importantly practically significant (the difference is "big" enough to make the experimented change beneficial to the company).

For each evaluation metric, we should calculate the difference between the values from both groups. Then, we compute the confidence interval for that difference and test whether or not this confidence interval is both statistically and practically significant.

Gross Conversion

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

Important: The given spreadsheet lists pageviews and clicks for 39 days, while it only lists enrollments and payments for 23 days. So, when working with enrollments and payments we should notice using only the corresponding pageviews and clicks, and not all of them.

```
In [72]: #count the total clicks from complete records only
         clicks_cont = control['Clicks'].loc[control['Enrollments'].notnull()].sum()
         clicks_exp = experiment['Clicks'].loc[experiment['Enrollments'].notnull()].sum()

         #Gross Conversion - number of enrollments divided by number of clicks
         enroll_cont = control['Enrollments'].sum()
         enroll_exp = experiment['Enrollments'].sum()

         GC_cont = enroll_cont/clicks_cont
         GC_exp = enroll_exp/clicks_exp
         GC_pooled = (enroll_cont + enroll_exp)/(clicks_cont + clicks_exp)
         GC_sd_pooled = mt.sqrt(GC_pooled *(1-GC_pooled)*(1/clicks_cont+1/clicks_exp))
         GC_ME=round(get_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: [",GC_diff-GC_ME,",",GC_diff+GC_ME,"]")
         print ("The change is statistically significant if the CI doesn't include 0. In that
```

```
The change due to the experiment is -2.06 %
Confidence Interval: [ -0.0292 , -0.012 ]
The change is statistically significant if the CI doesn't include 0. In that case, it is practi
```

According to this result there was a change due to the experiment, that change was both statistically and practically significant. We have a negative change of 2.06%, when we were willing to accept any change greater than 1%. This means the Gross Conversion rate of the experiment group (the one exposed to the change, i.e. asked how many hours they can devote to studying) has decreased as expected by 2% and this change was significant. This means less people enrolled in the Free Trial after due to the pop-up.

Net Conversion

The hypothesis is the same as before just with net conversion instead of gross. At this point we expect the fraction of payers (out of the clicks) to decrease as well.

```
In [81]: ##net conversion - number of payments divided by number of clicks
         payment_cont = control['Payments'].sum()
         payment_exp = experiment['Payments'].sum()

         NC_cont = payment_cont/clicks_cont
         NC_exp = payment_exp/clicks_exp
         NC_pooled = (payment_cont + payment_exp)/(clicks_cont + clicks_exp)
         NC_sd_pooled = mt.sqrt(NC_pooled * (1-NC_pooled)*(1/clicks_cont + 1/clicks_exp))
         NC_ME=round(get_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: [",NC_diff-NC_ME,",",NC_diff+NC_ME,"]")
         print ("The change is statistically significant if the CI doesn't include 0. In that c
```

```
The change due to the experiment is -0.49 %
Confidence Interval: [ -0.0116 , 0.0018000000000000004 ]
The change is statistically significant if the CI doesn't include 0. In that case, it is practi
```

In this situation the 0.0005 decrease is not in the confidence interval, which means that this is not statistically significant and practially significant.

Double check with Sign Tests In a sign test we get another angle at analyzing the results we got - we check if the trend of change we observed (increase or decrease) was evident in the daily data. We are going to compute the metric's value per day and then count on how many days the metric was lower in the experiment group and this will be the number of succssesses for our binomial variable. Once this is defined we can look at the proportion of days of success out of all the available days.

Data Preparation

```
In [82]: #let's first create the dataset we need for this:
         # start by merging the two datasets
         full=control.join(other=experiment,how="inner",lsuffix="_cont",rsuffix="_exp")
         #Let's look at what we got
         full.count()
```

```
Out[82]: Date_cont           37
         Pageviews_cont      37
         Clicks_cont         37
         Enrollments_cont    23
         Payments_cont       23
         Date_exp            37
         Pageviews_exp       37
         Clicks_exp          37
         Enrollments_exp     23
         Payments_exp        23
         dtype: int64
```

```
In [83]: #now we only need the complete data records
         full=full.loc[full["Enrollments_cont"].notnull()]
         full.count()
```

```
Out[83]: 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
```

```
In [84]: # Perfect! Now, derive a new column for each metric, so we have it's daily values
         # We need a 1 if the experiment value is greater than the control value=
         x=full['Enrollments_cont']/full['Clicks_cont']
         y=full['Enrollments_exp']/full['Clicks_exp']
         full['GC'] = np.where(x<y,1,0)
         # The same now for net conversion
         z=full['Payments_cont']/full['Clicks_cont']
         w=full['Payments_exp']/full['Clicks_exp']
         full['NC'] = np.where(z<w,1,0)
         full.head()
```

```
Out[84]:      Date_cont  Pageviews_cont  Clicks_cont  Enrollments_cont  Payments_cont  \
         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

               Date_exp  Pageviews_exp  Clicks_exp  Enrollments_exp  Payments_exp  GC  \
         0  Sat, Oct 11           7716         686            105.0          34.0   0
         1  Sun, Oct 12           9288         785            116.0          91.0   0
```

```
2  Mon, Oct 13          10480          884          145.0          79.0   0
3  Tue, Oct 14           9867          827          138.0          92.0   0
4  Wed, Oct 15           9793          832          140.0          94.0   0

   NC
0   0
1   1
2   0
3   0
4   1
```

```
In [85]: 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: 4
 No. of cases for NC: 10
 No. of total cases 23
```

Building a Sign Test

We can forget all about this part and just use an online sign test calculator, but for me that is just no fun - so I will implement the calculations behind it. What we want to do after we count the amount of days in which the experiment group had a higher metric value than that of the control group, is to see if that number is likely to be seen again in a new experiment (significance). We assume the chance of a day like this is random (50% chance to happen) and then use the binomial distribution with p=0.5 and the number of experiments (days) to tell us the probability of this happening according to a random chance. So, according to the binomial distribution with p=0.5 and n= total number of days; we want to now the probability of x days being a success (higher metric value in experiment). Because we are doing a two-tailed test we want to double this probability and once we have we can call it the pvalue and compare it to our . If the pvalue is greater than the the result is not significant and vice-versa.

p(successes)=n!x!(nx)!px(1p)nx

Recall that a pvalue is the probability of observing a test statistic as or more extreme than that observed. If we observed 2 days like that, the pvalue for the test is: pvalue=P(x<=2). We only need to remember the following: P(x<=2)=P(0)+P(1)+P(2).

```
In [86]: #first a function for calculating probability of x=number of successes
         def get_prob(x,n):
             p=round(mt.factorial(n)/(mt.factorial(x)*mt.factorial(n-x))*0.5**x*0.5**(n-x),4)
             return p
         #next a function to compute the pvalue from probabilities of maximum x
         def get_2side_pvalue(x,n):
             p=0
             for i in range(0,x+1):
```

```
            p=p+get_prob(i,n)
        return 2*p
```

Finally, to conduct the sign test itself: we will calculate the p-value for each metric, using the counts GC_x,NC_x and n and the function get_2side_pvalue.

```
In [87]: print ("GC Change is significant if",get_2side_pvalue(GC_x,n),"is smaller than 0.05")
         print ("NC Change is significant if",get_2side_pvalue(NC_x,n),"is smaller than 0.05")
```

```
GC Change is significant if 0.0026000000000000003 is smaller than 0.05
NC Change is significant if 0.6774 is smaller than 0.05
```

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.

Conclusions & Recommendations At this point, once we have seen that the actual underlying goal we had was not reached (increase fraction of paying users by asking them in advance if they have the time to invest in the course), we can only recommend to not continue with change. It may have caused a change in Gross conversion, but it didn't for net conversion.

```
In [ ]:
```