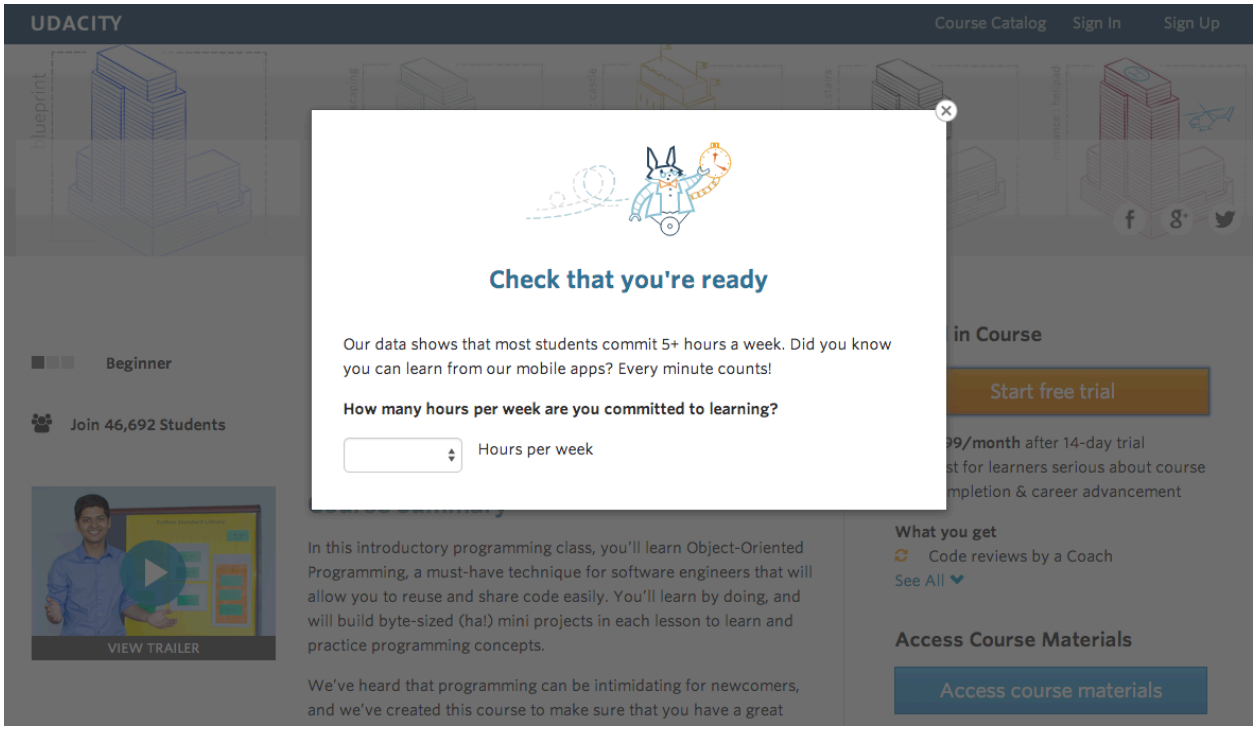
实验概述:免费试用筛选器

在本实验进行时，Udacity课程在课程概览页面上有两种选项:“开始免费试用”和“访问课程材料”。如果学生点击“开始免费试用”，他们将被要求输入信用卡信息，然后他们将被注册到课程付费版本的免费试用中。14天后，除非他们首先取消，否则他们将自动被收费。如果学生点击“访问课程材料”，他们将能够免费观看视频和参加测验，但他们将不会收到教练支持或经过验证的证书，也不会提交他们的最终项目以获得反馈。

在实验中，Udacity测试了一个变化，即如果学生点击“开始免费试用”，他们会被询问每周有多少时间可以投入到课程中。如果学生表示每周有5小时或更多时间，他们将像往常一样被引导通过结账流程。如果他们表示每周少于5小时，将出现一条消息，指出Udacity课程通常需要更大的时间承诺才能成功完成，并建议学生可能喜欢免费访问课程材料。此时，学生可以选择继续注册免费试用，或者改为免费访问课程材料。下面的截图显示了实验的外观:



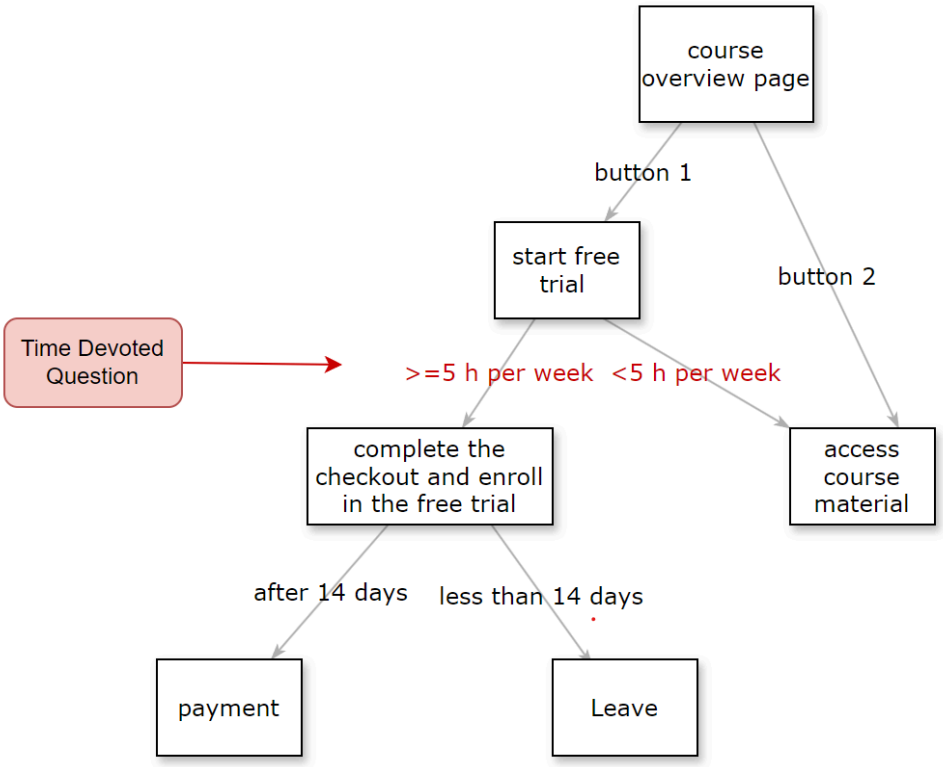
实验设计

1. 分流单位(由 **Udacity** 提供)

分流单位是cookie。如果学生注册了免费试用，他们将从那时起通过用户ID进行跟踪。但对于未注册的学生，即使他们在访问课程概览页面时已登录，他们也不会通过用户ID进行跟踪。同一个用户ID不能两次注册免费试用。

2. 初步假设

假设是“时间投入窗口”可能会为学生提前设定更清晰的期望，从而减少因时间不足而离开免费试用的沮丧学生的数量。此外，这一变化不会显著减少最终继续免费试用并完成课程的学生数量。如果假设成立，Udacity将进一步改善学生的整体体验，并提高教练支持可能完成课程的学生的能力。（由 Udacity 提供）



因此，我们可以根据提供的信息设定一些初始假设。然后，我们将在以后细化这些假设。

1. H0：更改对注册免费试用的学生数量没有影响
   H1：更改会减少注册免费试用的学生数量
2. H0：更改对学生退出免费试用的概率没有影响
   H1：更改会减少离开免费试用的学生的概率
3. H0：更改对学生至少进行一次付款的概率没有影响
   H1：更改会减少进行至少一次付款的学生的概率

3. 指标选择

Udacity提供了以下七个指标选择。

- **Cookie** 数量：即查看课程概览页面的独特 cookie 数量。(dmin=3000)
- 用户 **ID** 数量：即注册免费试用的用户数量。(dmin=50)
- 点击次数：即单击“开始免费试用”按钮（在触发免费试用筛选器之前发生）的独特 Cookie 数量。(dmin=240)
- 点击通过概率：即，单击“开始免费试用”按钮的唯一 Cookie 数量除以查看课程概述页面的独特 Cookie 数量。(dmin=0.01)
- 总转化率：即完成结账并注册免费试用的用户 ID 数量除以单击“开始免费试用”按钮的独特 Cookie 数量。(dmin= 0.01)
- 留存率：即在 14 天期限之后仍保持注册状态（至少进行一笔付款）的用户 ID 数量除以完成注册的用户 ID 数量。(dmin=0.01)
- 净转化率：即在 14 天期限后仍保持注册状态（至少进行一笔付款）的用户 ID 数量除以单击“开始免费试用”按钮所需的独特 Cookie 数量。(dmin=0.0075)

dmin表示每个指标的实际意义边界，即在业务上被认为是有意义的变化差异，以绝对值给出。

3.1 选择不变指标

不变指标是预计在控制组和实验组之间不会改变的指标，它们将用于健全性检查。

基于上面给出的指标，我们将使用这三个指标作为不变指标：

- **Cookie** 数量
- 点击次数
- 点击通过概率

因为需要在更改之前收集这三个指标的数据，所以它们应该保持不变。尽管点击通过概率涵盖了 Cookie 数量和点击次数，但在健全性检查期间它可能会出现意外变化，因此检查所有这三个指标是一个更安全的选择。

我们之所以不使用用户 ID数量是因为，我们只有注册免费试用的学生的用户 ID，因此我们无法保证对照和实验之间的分布相同并且不会受到更改的影响。

3.2 选择评估指标

对于评估指标，我们不仅关心统计显著性，更关心实际显著性(dmin)，因为它可能在统计上很重要，但在实践中不值得实施。

基于上面给出的指标，我们将把这三个指标作为A/B测试的评估指标。

- 总转化率：我们预计实验组这个指标会下降，因为时间问题会筛选出没有足够时间的学生，注册免费试用的学生数量会减少。
- 留存率：我们预计实验组这个指标会增加，因为问题已经筛选出了没有足够时间的学生，超过14天边界的学生比例将增加。
- 净转化：我们无法预期其方向，因为它是上述两个指标的产物，可能会有所不同。

3.3 假设的完善

根据我们选择的指标和最初的假设，我们可以完善我们的假设。

1. H0：总转化率（对照）= 总转化率（实验）
   H1：总转化率（对照）！= 总转化率（实验）
2. H0：留存率（对照）= 留存率（实验）
   H1：留存率（对照）！= 留存率（实验）
3. H0：净转化率（对照）= 净转化率（实验）
   H1：净转化率（对照）！= 净转化率（实验）

4. 计算指标变异性

基线值由Udacity提供：

每天查看课程概览页面的独特 cookie数量：	40000
每天单击“开始免费试用”的独特 cookie数量：	3200
每日注册量：	660
点击“开始免费试用”的点击通过概率：	0.08
点击后的注册概率：	0.20625
注册后的留存概率：	0.53
点击后的付款概率	0.1093125

由于每个评估指标都符合二项分布，并且分析单位与分流单位相同，我们可以分析性地计算标准差，不需要计算经验变异性。

此外，由于每种情况下的n相对较大，根据中心极限定理，样本均值的抽样分布接近正态分布，因此我们可以计算每个指标的抽样分布样本均值的标准差（简称标准误），作为每个指标标准差的一个很好的估计。（我们可以使用3标准差规则来检查n是否足够大。）（我们可以使用3-标准差 规则来检查 n 是否足够大。）

考虑到查看课程概览的cookie样本量为5000（由Udacity提供），我们可以使用以下公式计算每个指标的标准误：

$$SE = \sqrt{\frac{\hat{p}*(1-\hat{p})}{n}}$$

评估指标	标准误
总转化率	0.0202
留存率	0.0549
净转化率	0.0156

5. 规模

5.1 样本数量与功效

我们将显著性水平设为0.05，统计功效设为0.80（β为0.20）。。我们使用一个在线计算器 来计算所需的样本数量。

评估指标	样本量
总转化率	645,875
留存率	4,741,212
净转化率	685,325

根据上面的计算，为了检验这三个假设，我们需要4,741,212个用于查看课程概述页面的 cookie。

我们还需要决定是否在实验分析阶段使用Bonferroni校正。根据这篇论文，是否使用Bonferroni校正取决于研究的实际情况。当必须避免I型错误或者当我们进行大量测试而没有预先计划的假设时，应考虑使用。而对于这次分析，我们只有3个假设需要测试，并不需要对结果非常保守，所以我们决定不使用Bonferroni校正。

5.2 持续时间与暴露时间

从 Udacity 来看，考虑到每天有 40k 的页面浏览量，我们可以首先假设将 100% 的流量转移到这个实验中。在使用 100% 流量的情况下，运行此实验所需的天数为：

- 对于使用总转化率、留存率和净转化率的实验，我们需要：119天
- 对于使用总转化率和净转化率的实验，我们需要：17天

然而，如果我们将留存率作为指标之一，持续时间会太长而无法接受。因为我们不能在这个期间运行其他实验，这将产生机会成本。此外，如果变化使学生感到沮丧而降低了转化率，我们无法在4个多月内注意到这一点，这将损害我们的业务。因此，我们选择将50%的流量转移到这个实验中，并放弃留存率指标，因为留存率可以从其他两个指标中推断出来。

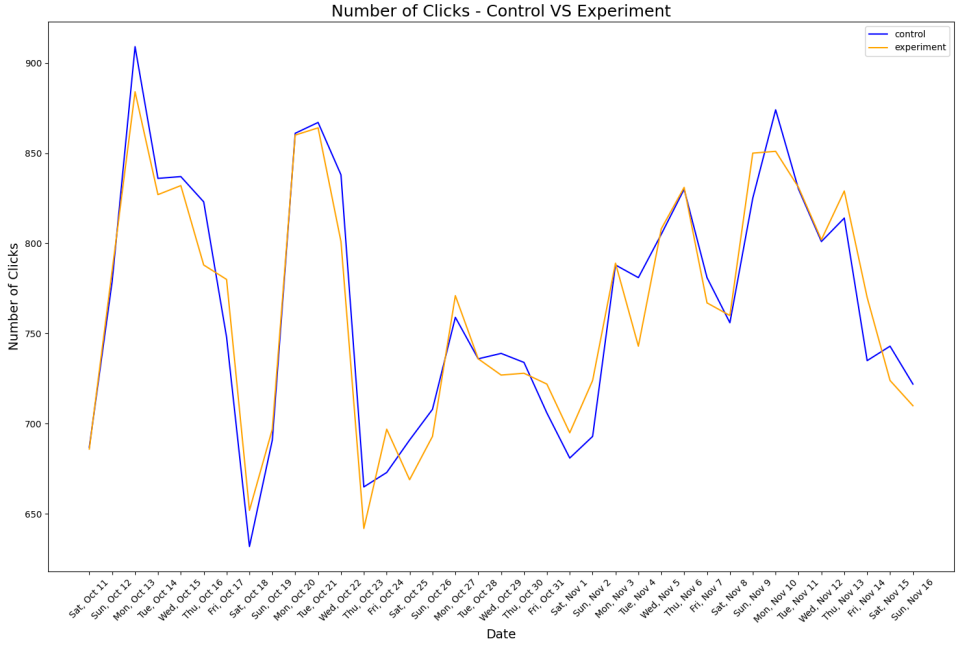
- 对于使用50%流量进行总转化率和净转化率的实验，我们需要：**34**天

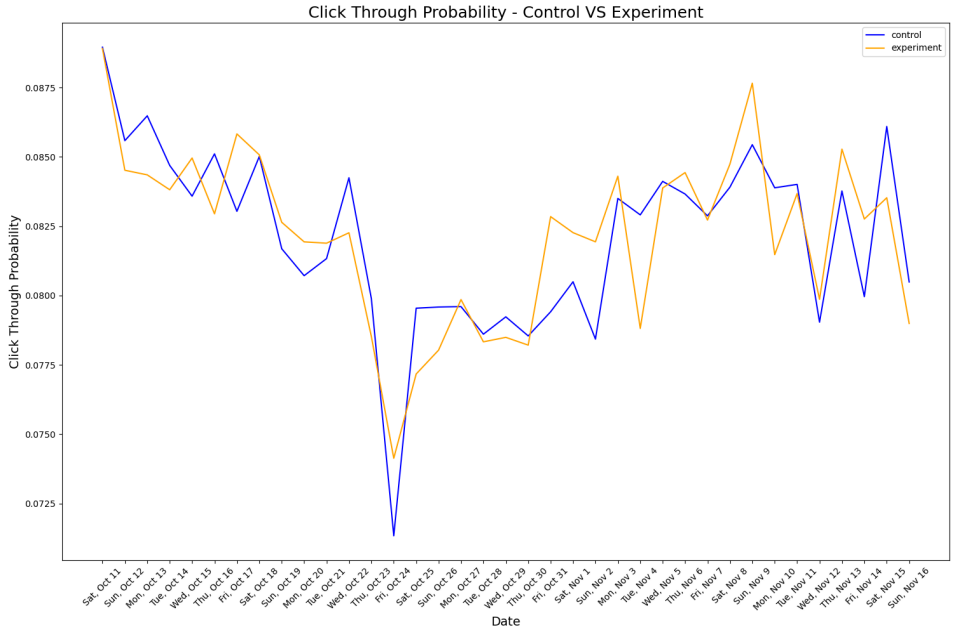
实验分析

1. 健全性检查

1.1 可视化

我们可以通过可视化不变指标来更清晰地比较控制组和实验组之间的差异。





我们可以看到，在10月24日的点击通过概率（CTP）上有大幅下降，这需要进一步调查。

1.2 健全性检查

根据Udacity提供的数据，我们在对照组中收集了345,543个cookie，在实验组中收集了344,660个cookie，因此总样本量为690,203。我们现在想要检查这些cookie是否已被随机分配到两组，这意味着两组之间不应存在显著差异。因此，观测值应该在置信区间内。

Cookies和点击数：因为cookie和点击应该被随机分配到控制组和实验组，所以进入任一组的概率是0.5。我们可以使用二项分布来模拟控制组中的cookie或者点击数，给定总样本量。此外，由于样本量相对较大（中心极限定理），我们使用 Z 检验来检查这些指标。

CTP：我们通常假设 CTP 服从正态分布。理想情况下，由于两个正态分布的自变量（假设独立）的方差线性性质，两组之间的 CTP 差值也遵循正态分布。由于我们还假设两组之间的方差相等，因此我们将使用合并标准误作为差异标准误差的更好估计。（不相等时使用韦尔奇 t 检验 ）

不变指标	置信区间上限	置信区间下限	观测值	结果
Cookies	0.4988	0.5012	0.5006	通过
点击次数	0.4959	0.5042	0.5005	通过
CTP差值	-0.0013	0.0013	0.0001	通过

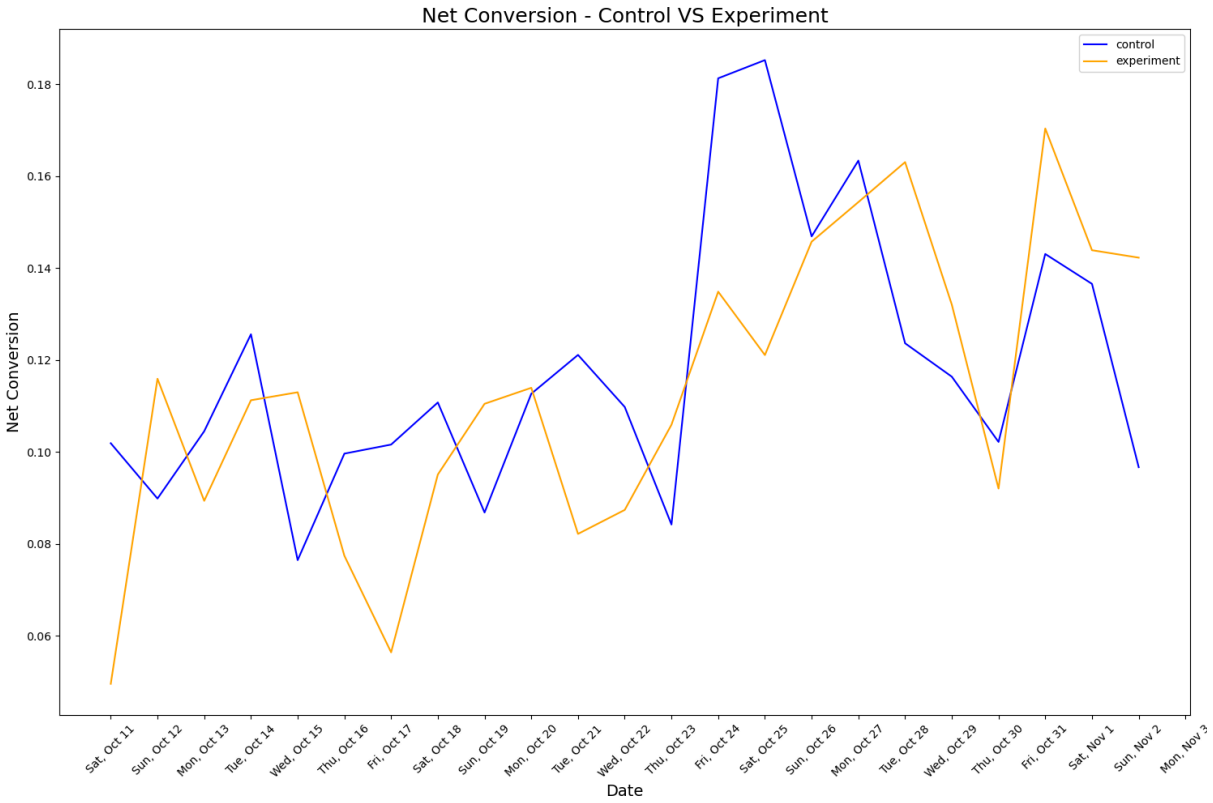
根据 Z 检验结果，不变指标的健全性检查已全部通过。

2. 结果分析

2.1 效应大小测试

2.1.1 可视化

与之前一样，我们将实验指标可视化，以便更清楚地比较两组之间的情况。



我们还可以观察到，在10月24日的两个指标上都有一个巨大的峰值。基于我们在CTP指标上的观察，我们可以推断出当天的点击数量相对较少。因此，我们建议Udacity团队查看所以流量，而不仅仅是这部分流量，以找出原因。

2.1.2 实际和统计显著性检查

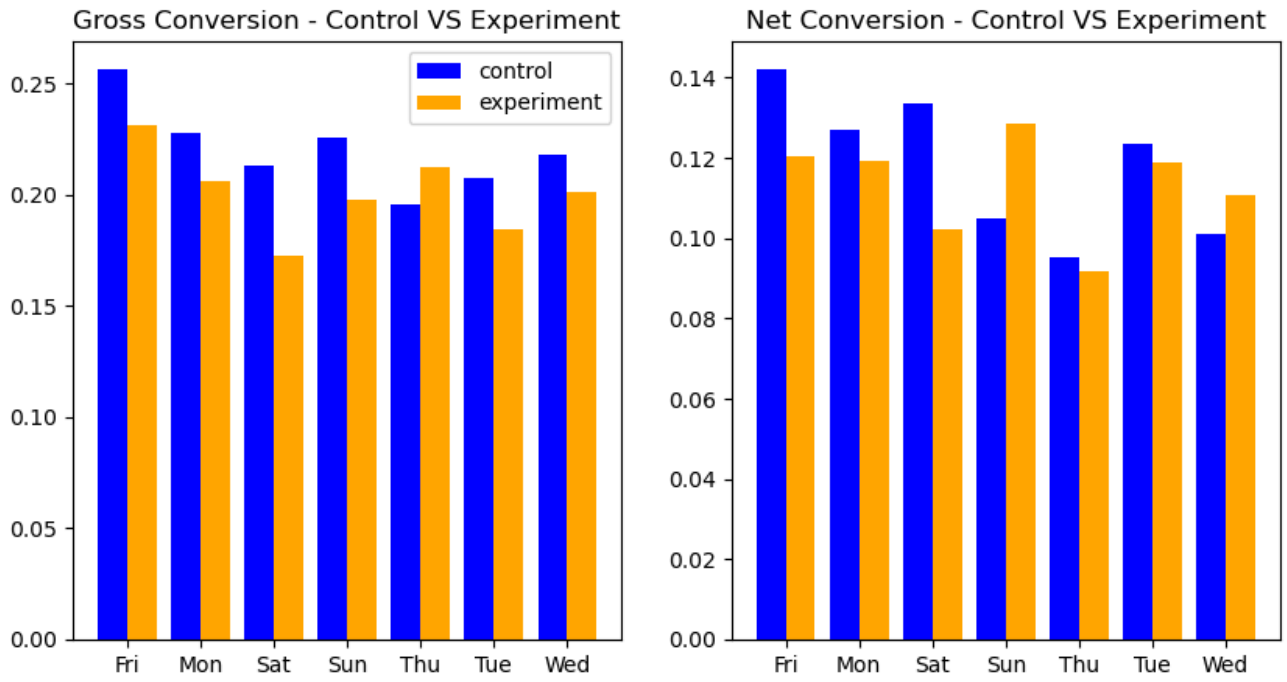
两个转化率指标都是比例，与CTP类似，所以我们可以像检查CTP一样检查它们差异的显著性。此外，我们将检查它们的统计和实际意义，因为实际意义决定了是否实施这一变化。(注意:我们需要计算观察值周围的置信区间，并检查置信区间是否包含0，而不是像以前我们对不变指标那样计算。)

由于学生在14天试用期后才会进行支付，因此最后14天内的支付和注册都是空值。我们需要685,325个cookie来获得一个稳健的测试结果，但我们只有423,525个cookie，因为没有最后14天的数据，所以数据不够。因此，我们建议延长实验周期以获取更多数据。

评估指标	置信区间上限	置信区间下限	观测值	统计显著结果	最小	实际显著结果
总转化率差值	-0.0291	-0.0120	-0.0205	通过	-0.01	通过
净转化率差值	-0.0116	0.0019	-0.0049	失败	0.0075	失败

2.2 周分析

在某些情况下，进行符号检验可以帮助我们逐日检查数据，但它要求来自两个总体的样本是相关的或配对的，这违反了我们 A/B 检验的假设。因此，我们进行了周分析，而不是符号检验，试图在我们的数据中找到一些规律。



从上图可以看出，周四的总转化率实验效果较差。至于净转化率，仅在周日和周三有效。

2.3 结果总结

1. 对于总转化率，我们观察到大约2%的下降，这与我们在假设阶段的预期相符。而且2%的下降在统计和实际上都是显著的。此外，效果在星期六很明显，但在星期四几乎没有影响。
2. 对于净转化率，观察值约为-0.5%，与我们预期的方向不符，并且在两个显著性测试中都失败了。此外，正如周分析所示，我们只在星期三和星期日观察到增加，但在其他五天都观察到减少。

3. 建议

这个改变确实对减少沮丧的学生并将他们引导到免费课程材料方面有影响。但Udacity业务的整体目标是增加收入，因此用户是否支付更为重要。在这种情况下，尽管我们在总转化率上观察到了显著，但我们仍然不能决定推出这个改变，因为净转化率更为关键，但其在统计和实际上都不

显著。然而，由于我们没有收集足够的数据来测试这两个指标，我们需要进一步的实验来确认我们的结果。

后续实验

如果我们延长了实验周期，而净转化率的测试仍然失败，我们可以进行其他实验，试图减少早期取消（在14天免费试用期结束并触发支付之前的取消）。考虑的方面有两个：注册前和注册后。

- 注册前：我们可以重复使用本次测试中所做的实验设计，只需对免费试用筛选器进行一些修改，看看是否会提高总转化率和净转化率。例如，我们可以在筛选器上添加一些鼓励措施，这将使学生更有动力完成课程。比如“ 完成课程，你离梦想又近了一步！”，这可以帮助学生更能坚持完成课程。其次，对于那些选择不注册课程的人，我们可以添加一个按钮，将他们引导至免费课程材料页面。此按钮改善了用户体验，并鼓励了更多学生访问免费材料。更改后，我们还可以测试一下总转化率和净转化率，分别会下降和上升。

- 注册后：当学生在观看在线课程或做作业时，如果他们在某个课程或某个作业上花费的时间超过平均时间时，我们可以添加一个弹出窗口，询问他们是否需要教学习辅导。这可以帮助提高留存率，因为学生可以更容易地学习。

    - 实验设计：对于已经注册的学生，随机将他们分配到实验组和控制组。
    - 转移单位：用户 ID
      理由：因为我们只考虑那些已经注册了该课程的人，所以我们可以使用用户 ID 来 跟踪它们。
    - 假设：
      H0：弹窗对离开免费试用的学生数量没有影响。
      H1：弹窗减少了离开免费试用的学生数量。
      理由：有了及时的学习辅导，学生就不太可能因课程的难度而感到沮丧，因此更有可能在试用期结束后留下来。
    - 不变指标：用户 ID
      理由：用户 ID是本次测试的分流单位，是我们可以根据变化前收集到的信息。
    - 评估指标：留存率
      理由：留存率是指用在 14 天期限后仍保持注册状态（至少进行一笔付款）的用户 ID 数量除以完成结账的用户 ID 数量得到的指标。

由于通过了健全性检查，如果第一个实验中的总转化率和净转化率，或第二个实验中的留存率在统计和实际上都显著，并且资源和成本对于 Udacity 来说是可以接受的，我们就可以推出这个更改。

相关参考：
https://github.com/zyellieyan/AB-Testing-Project?tab=readme-ov-file
https://www.kaggle.com/code/mariusmesserschmied/udacity-a-b-testing-final-course-project/notebook

附录一一 Python 代码实现

1. For count metrics, SE = root (p*(1-p)/n); For proportion metrics, SE_pooled = root (s**2/n + s**2/n, s = root (p*(1-p))
2. n is large enough, use Z test
3. 14 days data is missing, recalculate sample size

Visualization for invariant variables

In [25]:
```
import pandas as pd
```

In [26]:
```
ctl = pd.read_csv('D:\\pandas\\YUQI\\Udacity\\Final Project Results - Control.csv')
ctl.head()
```

Out[26]:

	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

In [28]:
```
ctl.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 37 entries, 0 to 36
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Date         37 non-null     object 
 1   Pageviews    37 non-null     int64  
 2   Clicks       37 non-null     int64  
 3   Enrollments  23 non-null     float64
 4   Payments     23 non-null     float64
dtypes: float64(2), int64(2), object(1)
memory usage: 1.6+ KB
```

In [29]:
```
exp = pd.read_csv('D:\\pandas\\YUQI\\Udacity\\Final Project Results - Experiment.csv')
exp.head()
```

Out[29]:

	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 [30]:
```
exp.info()
```

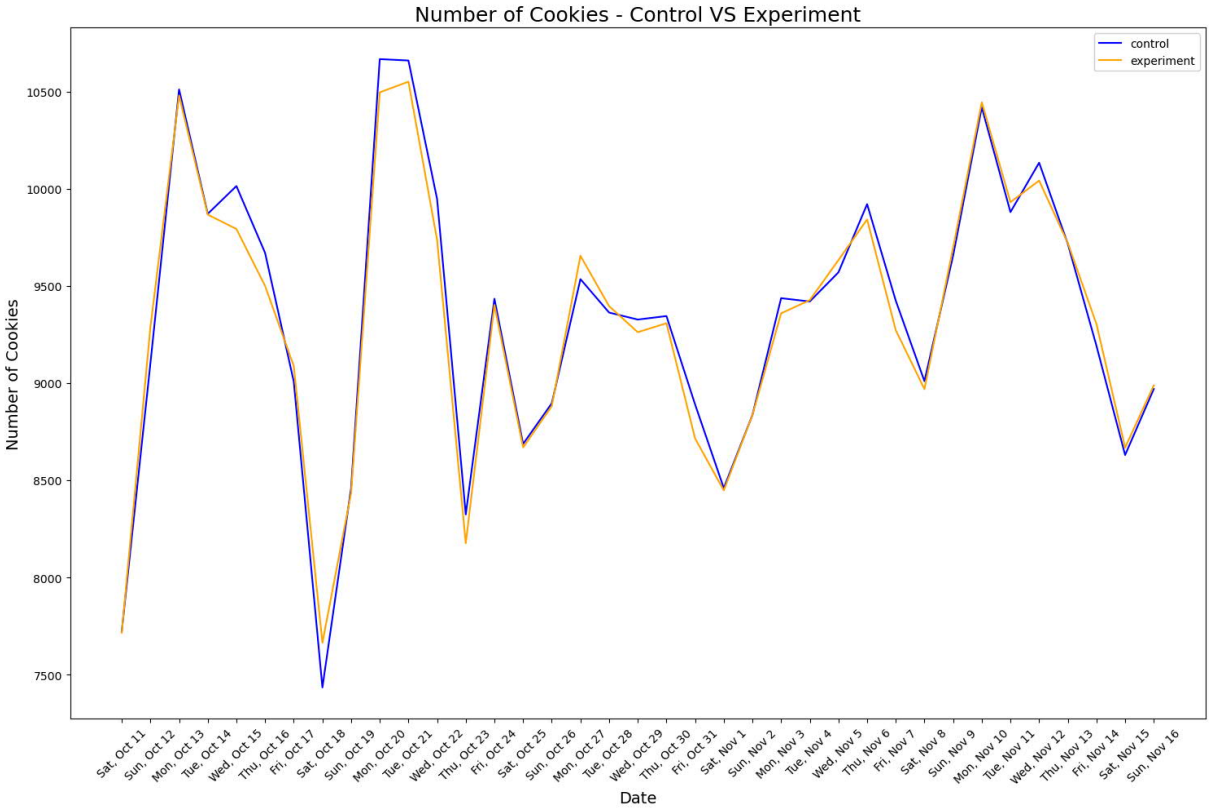
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 37 entries, 0 to 36
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Date         37 non-null     object 
 1   Pageviews    37 non-null     int64  
 2   Clicks       37 non-null     int64  
 3   Enrollments  23 non-null     float64
 4   Payments     23 non-null     float64
dtypes: float64(2), int64(2), object(1)
memory usage: 1.6+ KB
```

1. visualize invariant metrics to get a clearer comparison

In [31]:
```
from matplotlib import pyplot as plt
filepath = 'D:\\pandas\\YUQI\\Udacity\\'
```

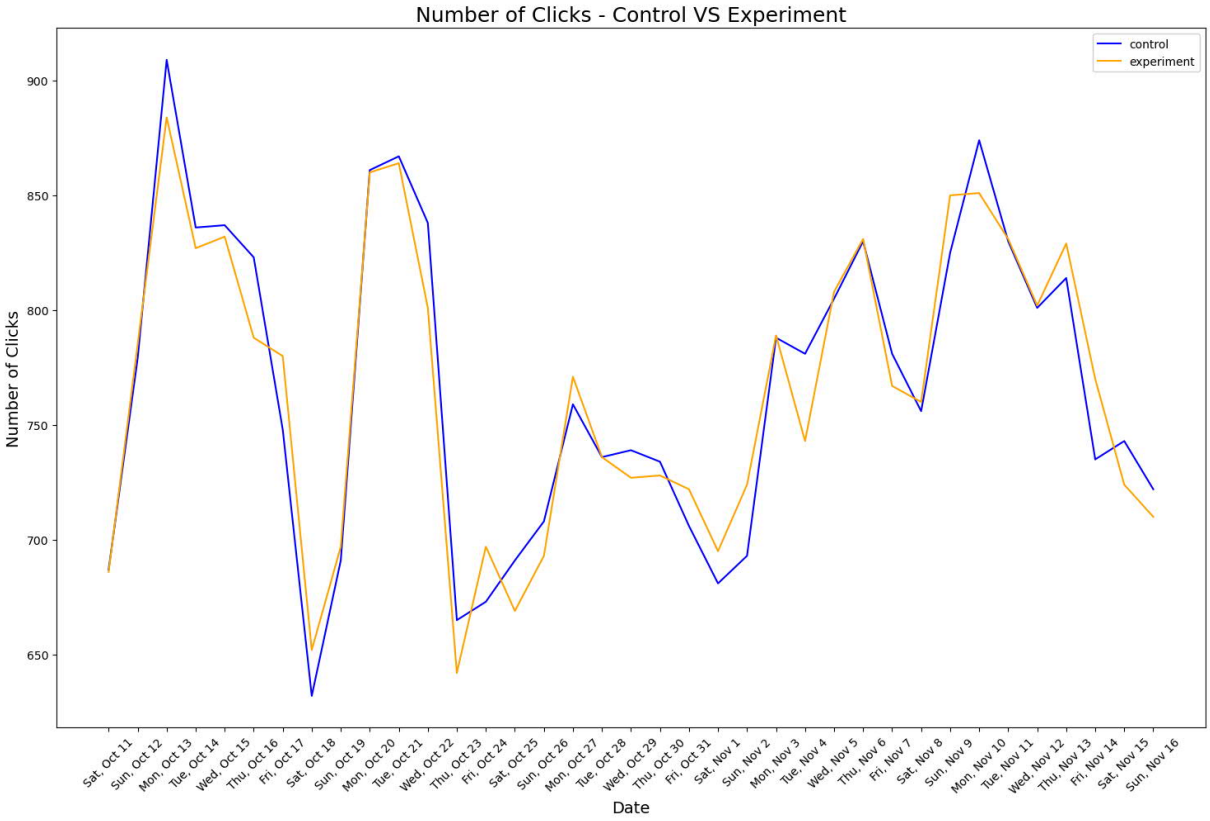
1.1 pageview

In [32]:
```
plt.figure(figsize=(18,11))
plt.plot(ctl['Date'], ctl['Pageviews'], color = 'blue', label = 'control')
plt.plot(exp['Date'], exp['Pageviews'], color = 'orange', label =  'experiment')
plt.xticks(rotation = 45)
plt.xlabel('Date', fontsize = 14)
plt.ylabel('Number of Cookies', fontsize = 14)
plt.title('Number of Cookies - Control VS Experiment', fontsize = 18)
plt.legend()
plt.savefig(filepath+'pageviews')
plt.show()
```



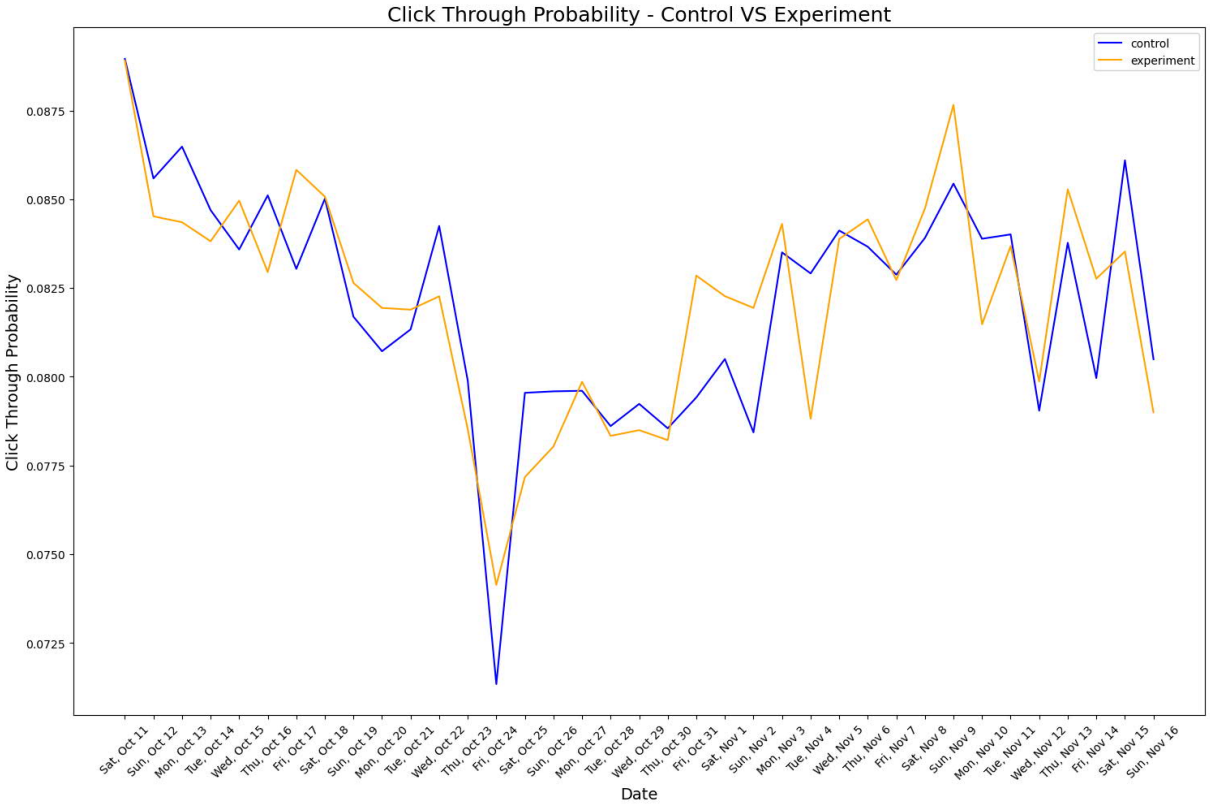
1.2 clicks

In [33]:
```
plt.figure(figsize=(18,11))
plt.plot(ctl['Date'], ctl['Clicks'], color = 'blue', label = 'control')
plt.plot(exp['Date'], exp['Clicks'], color = 'orange', label =  'experiment')
plt.xticks(rotation = 45)
plt.legend()
plt.xlabel('Date', fontsize = 14)
plt.ylabel('Number of Clicks', fontsize = 14)
plt.title('Number of Clicks - Control VS Experiment', fontsize = 18)
plt.savefig(filepath+'Clicks')
plt.show()
```



1.3 CTP

In [34]:
```
plt.figure(figsize=(18,11))
plt.plot(ctl['Date'], ctl['Clicks']/ctl['Pageviews'], color = 'blue', label = 'control')
plt.plot(exp['Date'], exp['Clicks']/exp['Pageviews'], color = 'orange', label =  'experiment')
plt.xticks(rotation = 45)
plt.xlabel('Date',fontsize = 14)
plt.ylabel('Click Through Probability',fontsize = 14)
plt.title('Click Through Probability - Control VS Experiment', fontsize = 18)
plt.legend()
plt.savefig(filepath+'CTP')
plt.show()
```



we can see there is a significant drop on Oct 23. We will look into that later.

In [35]:
```
import numpy as np
from scipy import stats
```

2. Define a Function for SE of pageviews and clicks, a Function for pooled SE of CTP

In [36]:
```
def SE(n,p) -> float:
    # for count metrics
    # n: sample size in total
    # p: probability
    return (p*(1-p)/n)**0.5

def pooled_SE(s_con, s_exp,n_con, n_exp) -> float:
    # for proportion metrics with different variance between 2 groups
    # s_con: estimated standard deviation of CTP in control group (use frequency as probability)
    # s_exp: estimated standard deviation of CTP in experiment group (root p*(1-p))
    # n_con: sample size of the denominator of the metric in control group
    # n_exp: sample size of the denominator of the metric in experiment group
    return (s_con**2/n_con + s_exp**2/n_exp)**0.5
```

Sanity check for invariant metrics

In [37]:
```
results = pd.DataFrame(columns=['metrics', 'CI_left', 'CI_right', 'observed', 'sanity check'])
results['metrics'] = ['Number of Cookies', 'Number of Clicks', 'Difference_CTP']
results
```

Out[37]:

	metrics	CI_left	CI_right	observed	sanity check
0	Number of Cookies	NaN	NaN	NaN	NaN
1	Number of Clicks	NaN	NaN	NaN	NaN
2	Difference_CTP	NaN	NaN	NaN	NaN

In [38]:
```
n_cookie = ctl['Pageviews'].sum()+exp['Pageviews'].sum()
n_click = ctl['Clicks'].sum()+exp['Clicks'].sum()
fq_con = ctl['Clicks'].sum()/ctl['Pageviews'].sum()
fq_exp = exp['Clicks'].sum()/exp['Pageviews'].sum()
observed = [ctl['Pageviews'].sum()/n_cookie,
            ctl['Clicks'].sum()/n_click,
            (fq_exp - fq_con)]

results['observed'] = [round(x, 4) for x in observed]
results
```

Out[38]:

	metrics	CI_left	CI_right	observed	sanity check
0	Number of Cookies	NaN	NaN	0.5006	NaN
1	Number of Clicks	NaN	NaN	0.5005	NaN
2	Difference_CTP	NaN	NaN	0.0001	NaN

1. Calculate CI = mean +- Z*SE

In [39]:
```
# se
se_cookie = SE(n_cookie, 0.5) # the probability of assigning to control is 0.5
se_click = SE(n_click, 0.5)
s_con = (fq_con*(1-fq_con))**0.5
s_exp = (fq_exp*(1-fq_exp))**0.5
se_pooled = pooled_SE(s_con, s_exp, ctl['Pageviews'].sum(), exp['Pageviews'].sum())

# CI
alpha = 0.05
se = [se_cookie, se_click, se_pooled]
mean = [0.5, 0.5, 0]
ci_left = []
ci_right = []
for i, j in zip(mean, se):
    ci_left.append(i - stats.norm.ppf(1-alpha/2)*j)
    ci_right.append(i + stats.norm.ppf(1-alpha/2)*j)

results['CI_left'] = ci_left
results['CI_right'] = ci_right
results
```

Out[39]:

	metrics	CI_left	CI_right	observed	sanity check
0	Number of Cookies	0.498820	0.501180	0.5006	NaN
1	Number of Clicks	0.495885	0.504115	0.5005	NaN
2	Difference_CTP	-0.001296	0.001296	0.0001	NaN

2. Sanity Check

In [40]:
```
sanity_check = []
for i in [0,1,2]:
    if ci_left[i] <= observed[i] <= ci_right[i]:
        sanity_check.append('pass')
    else:
        sanity_check.append('not pass')
results['sanity check'] = sanity_check
results
```

Out[40]:

	metrics	CI_left	CI_right	observed	sanity check
0	Number of Cookies	0.498820	0.501180	0.5006	pass
1	Number of Clicks	0.495885	0.504115	0.5005	pass
2	Difference_CTP	-0.001296	0.001296	0.0001	pass

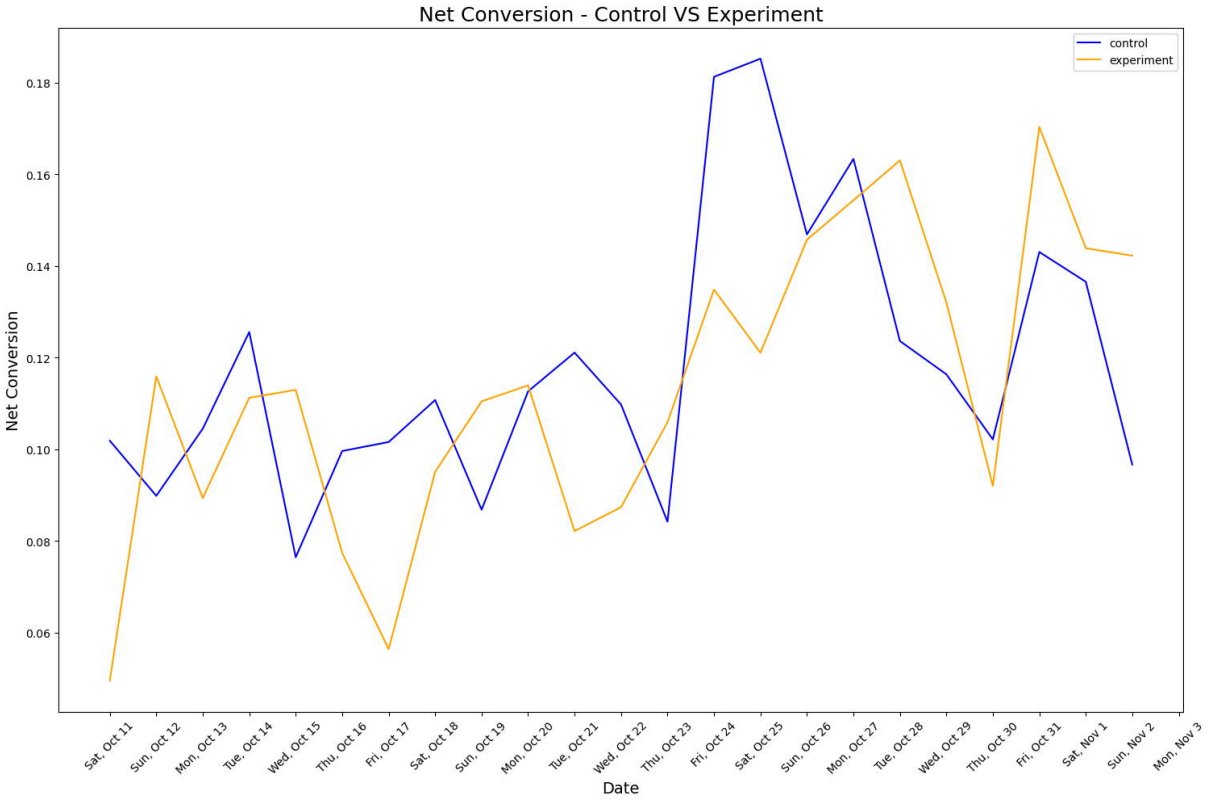
Visualize the experiment metrics

1.1 Gross Conversion

In [41]:
```
plt.figure(figsize=(18,11))
plt.plot(ctl['Date'], ctl['Enrollments']/ctl['Clicks'], color = 'blue', label = 'control')
plt.plot(exp['Date'], exp['Enrollments']/exp['Clicks'], color = 'orange', label =  'experiment')
plt.xticks(rotation = 45)
plt.xlabel('Date',fontsize = 14)
plt.ylabel('Gross Conversion',fontsize = 14)
plt.title('Gross Conversion - Control VS Experiment', fontsize = 18)
plt.legend()
plt.savefig(filepath+'Gross Conversion')
plt.show()
```

1.2 Net Conversion

In [42]:
```
plt.figure(figsize=(18,11))
plt.plot(ctl['Date'], ctl['Payments']/ctl['Clicks'], color = 'blue', label = 'control')
plt.plot(exp['Date'], exp['Payments']/exp['Clicks'], color = 'orange', label =  'experiment')
plt.xticks(rotation = 45)
plt.xlabel('Date',fontsize = 14)
plt.ylabel('Net Conversion',fontsize = 14)
plt.title('Net Conversion - Control VS Experiment', fontsize = 18)
plt.legend()
plt.savefig(filepath+'Net Conversion')
plt.show()
```



Practical and Statistical Significance check for experiment metrics

In [43]:
```
sig_check = pd.DataFrame(columns=['metrics', 'CI_left', 'CI_right', 'observed value', 'dmin', 'stat_check', 'prac
```

In [44]:
```
sig_check['metrics'] = ['Diff_GC', 'Diff_NC']
sig_check['dmin'] = [0.01, 0.0075]
sig_check
```

Out[44]:

	metrics	CI_left	CI_right	observed value	dmin	stat_check	prac_check
0	Diff_GC	NaN	NaN	NaN	0.0100	NaN	NaN
1	Diff_NC	NaN	NaN	NaN	0.0075	NaN	NaN

In [45]:
```
ctl['Pageviews'].iloc[:23].sum()+exp['Pageviews'].iloc[:23].sum()
```

Out[45]:
```
423525
```

We need 685,325 cookies to obtain a robust test result. But since payment can only be made after 14-day trial, we don't have the last 14 day data and thus we only have 423,525 cookies to check the Net Conversion metric, which is not enough

In [46]:
```
# observed value
gc_con = ctl['Enrollments'].iloc[:23].sum()/ctl['Clicks'].iloc[:23].sum()
gc_exp = exp['Enrollments'].iloc[:23].sum()/exp['Clicks'].iloc[:23].sum()

nc_con = ctl['Payments'].iloc[:23].sum()/ctl['Clicks'].iloc[:23].sum()
nc_exp = exp['Payments'].iloc[:23].sum()/exp['Clicks'].iloc[:23].sum() # calculate from 15th day
## may not have enough sample
diff_GC = gc_exp - gc_con
diff_NC = nc_exp - nc_con
sig_check['observed value'] = [diff_GC, diff_NC]

# SE & pooled SE
s_GC_con = (gc_con*(1-gc_con))**0.5
s_GC_exp = (gc_exp*(1-gc_exp))**0.5

s_NC_con = (nc_con*(1-nc_con))**0.5
s_NC_exp = (nc_exp*(1-nc_exp))**0.5

pooled_SE_GC = pooled_SE(s_GC_con, s_GC_exp, ctl['Clicks'].iloc[:23].sum(), exp['Clicks'].iloc[:23].sum())
pooled_SE_NC = pooled_SE(s_NC_con, s_NC_exp, ctl['Clicks'].iloc[:23].sum(), exp['Clicks'].iloc[:23].sum())

SE_pooled = [pooled_SE_GC, pooled_SE_NC]
```

1. Calculate CI = observed_value- Z*pooled_se

In [47]:
```
CI_left = []
CI_right = []
for i in range(2):
    CI_left.append(sig_check['observed value'][i] - stats.norm.ppf(1-alpha/2)*SE_pooled[i])
    CI_right.append(sig_check['observed value'][i] + stats.norm.ppf(1-alpha/2)*SE_pooled[i])

sig_check['CI_left'] = CI_left
sig_check['CI_right'] = CI_right

sig_check
```

Out[47]:

	metrics	CI_left	CI_right	observed value	dmin	stat_check	prac_check
0	Diff_GC	-0.029120	-0.011990	-0.020555	0.0100	NaN	NaN
1	Diff_NC	-0.011604	0.001857	-0.004874	0.0075	NaN	NaN

2. Check stats and prac significance

In [48]:
```
stat = []
prac = []
for i in range(2):
    if CI_left[i] <= 0 <= CI_right[i]:
        stat.append('not statistically significant')
    else:
        stat.append('statistically significant')

for i in range(2):
    if abs(sig_check['observed value'][i]) <= sig_check['dmin'][i]:
        prac.append('not practically significant')
    else:
        prac.append('practically significant')

sig_check['stat_check'] = stat
sig_check['prac_check'] = prac
```

In [49]:
```
sig_check
```

Out[49]:

	metrics	CI_left	CI_right	observed value	dmin	stat_check	prac_check
0	Diff_GC	-0.029120	-0.011990	-0.020555	0.0100	statistically significant	practically significant
1	Diff_NC	-0.011604	0.001857	-0.004874	0.0075	not statistically significant	not practically significant

Weekly analysis

In [73]:
```
df = pd.DataFrame(columns=['week_day', 'GC_con', 'GC_exp', 'NC_con', 'NC_exp'])
```

In [74]:
```
df['week_day'] = ctl['Date'].str[:3]
```

In [75]:
```
df['GC_con'] = ctl['Enrollments']/ctl['Clicks']
df['GC_exp'] = exp['Enrollments']/exp['Clicks']
df['NC_con'] = ctl['Payments']/ctl['Clicks']
df['NC_exp'] = exp['Payments']/exp['Clicks']
```

In [76]:
```
df.head()
```

Out[76]:

	week_day	GC_con	GC_exp	NC_con	NC_exp
0	Sat	0.195051	0.153061	0.101892	0.049563
1	Sun	0.188703	0.147771	0.089859	0.115924
2	Mon	0.183718	0.164027	0.104510	0.089367
3	Tue	0.186603	0.166868	0.125598	0.111245
4	Wed	0.194743	0.168269	0.076464	0.112981

In [81]:
```
weekday = df.groupby('week_day').mean()
weekday
```

Out[81]:

week_day	GC_con	GC_exp	NC_con	NC_exp
Fri	0.256180	0.231078	0.141981	0.120545
Mon	0.227446	0.206066	0.126848	0.119222
Sat	0.213220	0.172846	0.133614	0.102404
Sun	0.225736	0.198024	0.105066	0.128601
Thu	0.195392	0.212692	0.095342	0.091788
Tue	0.207303	0.184162	0.123449	0.118822
Wed	0.217761	0.201516	0.100874	0.110807

In [87]:
```
wd = weekday.index.tolist()
wd
```

Out[87]:
```
['Fri', 'Mon', 'Sat', 'Sun', 'Thu', 'Tue', 'Wed']
```

Visualize the weekly data

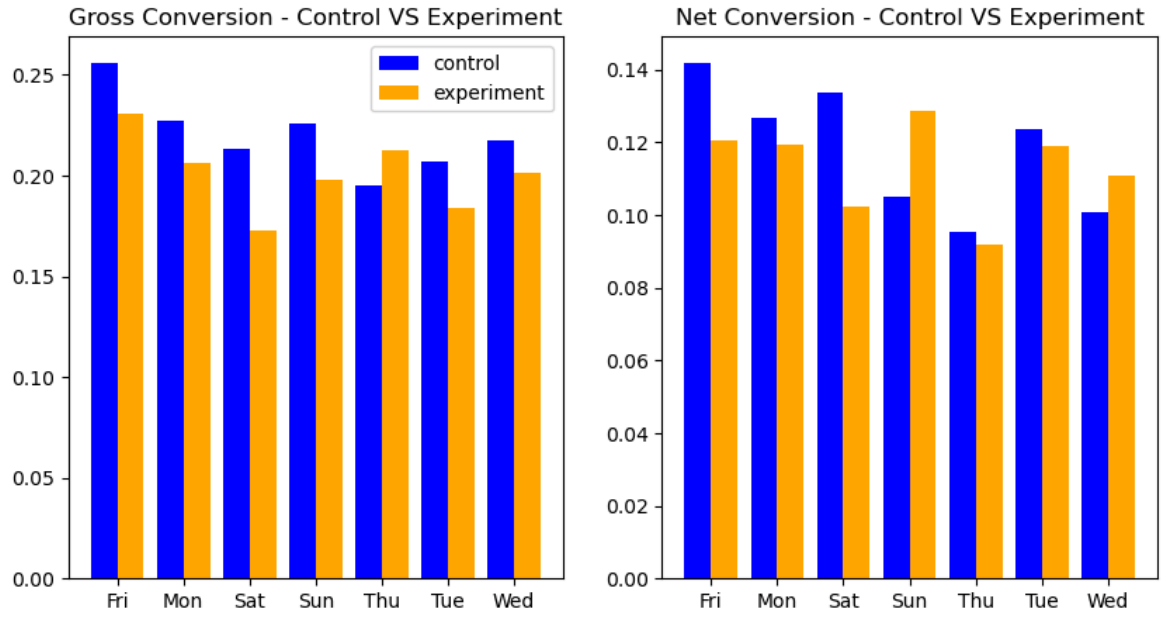
1.1 Gross Conversion

In [108]:
```
x = range(weekday.shape[0])
fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (10, 5))

ax1.bar(x, weekday['GC_con'], color = 'blue', width = 0.4, label = 'control')
ax1.bar([i + 0.4 for i in x], weekday['GC_exp'], width = 0.4, color = 'orange', label = 'experiment')
ax1.set_xticks([i+0.2 for i in x])
ax1.set_xticklabels(wd)
ax1.legend()
ax1.set_title('Gross Conversion - Control VS Experiment')

ax2.bar(x, weekday['NC_con'], color = 'blue', width = 0.4, label = 'control')
ax2.bar([i + 0.4 for i in x], weekday['NC_exp'], width = 0.4, color = 'orange', label = 'experiment')
ax2.set_xticks([i+0.2 for i in x])
ax2.set_xticklabels(wd)
ax2.set_title('Net Conversion - Control VS Experiment')

plt.savefig(filepath+'weekly analysis')
plt.show()
```



In [ ]: