

分析A/B测试结果

目录

- [简介](#)
- [上一步](#)
- [上一步 进阶](#)
- [返回](#)

大家好，此次我带了一份自己完成的报告，该报告包括模拟零假设抽样建立正态分布、假定性检验验证我的零假设与备择假设、逻辑回归检测交互项的相关性。

简介

数学对于生活，同样也服务于生活，个人非常乐意得到这些你所学习的知识服务社会，A/B测试通常是非常多需要做的，它可以帮我们改善现存的状态，很大程度上帮助公司、机构、政府作出不同的决策，也正因为它，本人对这个非常感兴趣

首先，对于这个选题，该选题是电子商务网站运行的A/B测试的结果，我的目标是通过这个notebook来帮助公司弄清楚他们是否应该使用新的页面，保留旧的页面，或者看目标时间是否延长，之后得出相关结论。

```
In [69]: import pandas as pd
import numpy as np
import random
import matplotlib.pyplot as plt
import matplotlib

#We are setting the seed to ensure you get the same answers on quizes as we set up
random.seed(42)

1. 现在，导入 ab_data.csv 数据

a 导入数据查看有几行：
```

```
In [70]: df = pd.read_csv('ab_data.csv')
df.head()
```

```
Out[70]:
```

	user_id	timestamp	group	landing_page	converted
0	851104	2017-01-21 22:11:48.556739	control	old_page	0
1	804228	2017-01-12 08:01:45.159739	control	old_page	0
2	661590	2017-01-11 16:55:06.154213	treatment	new_page	0
3	853541	2017-01-08 18:28:03.143765	treatment	new_page	0
4	864975	2017-01-21 01:52:26.210827	control	old_page	1

b 使用下表的单元格来查找数据集中的行数。

```
In [71]: # 原始数据总行数
df.shape()
```

```
Out[71]: 294478
```

```
In [72]: # 按之用户数据
len(df.user_id.unique())
```

```
Out[72]: 290584
```

c 数据集中独立用户的数量。

```
In [73]: # 用户转化（付费）的比率
from functools import partial
len(df.query('converted == 1'))/len(df)
```

```
Out[73]: 0.1196913955065512
```

ps. 用户转化与付费的思想一样，用户浏览该网站付费了数相当于转化。同理，未付费就是未转化
关于control与treatment—control代表旧页面，treatment代表新页面
正常来说ab_page作为分组，把new_page作为实验组（请向上看实验数据）

d 用户转化的比例。

```
In [74]: df_new = df.query('landing_page == "new_page"')
df_new.groupby('group').shape()
```

```
Out[74]:
```

group	count
control	147239
treatment	147239

e new_page 与 treatment 不一致的次数，理由：因为我们想让新页面(new_page)对应的全部为测试用户(treatment),所以查看数据集中是否有新页面(new_page)对应的控制用户(control),这些字段的存在会干扰分析结果，所以删掉它，结果如需要删掉这1926行

```
In [75]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 294478 entries, 0 to 294477
Data columns (total 5 columns):
user_id      294478 non-null object
timestamp    294478 non-null object
group        294478 non-null object
landing_page  294478 non-null object
converted     294478 non-null int64
dtypes: int64(1), object(3)
memory usage: 11.2+ MB
```

f. 查看数据集中是否存在缺失值，结果无任何的缺失值

```
2.
```

a. 现在，我们将 dataframe 存储在 df2 中。

```
In [76]: lists = df_new.query('group != "treatment"').index
df2 = df.drop(lists)
df_3 = df.query('landing_page == "old_page"')
lists = df_3.query('group == "control"').index
df2 = df2.drop(lists)
```

现在我再一次清洗了下载数据集删掉了 treatment 不与 new_page 一致的行或 control 不与 old_page 一致的行

```
In [77]: df2.shape()
```

```
Out[77]: 290585
```

重新查看下行数

```
In [78]: # Double Check all of the correct rows were removed - this should be 0
df2[(df2['group'] == 'treatment') == (df2['landing_page'] == 'new_page')] == False).shape()
```

```
Out[78]: 0
```

检查是否有遗漏

```
In [79]: df2.user_id.value_counts().count()
```

```
Out[79]: 290584
```

3.

a. df2 中有290584个唯一的 user_id 属性值id

```
In [80]: df2 = df2[df2['user_id']. duplicated()]

C. 删除含有重复的 user_id 的行。
```

以上对原始数据的初步清洗基本完成

```
In [81]: df2.head()
```

```
Out[81]:
```

	user_id	timestamp	group	landing_page	converted
0	851104	2017-01-21 22:11:48.556739	control	old_page	0
1	804228	2017-01-12 08:01:45.159739	control	old_page	0
2	661590	2017-01-11 16:55:06.154213	treatment	new_page	0

放上三行数据便于查找变量

4.

a. 不管用户收到什么页面，单个用户的转化率约为 0.119697

```
In [82]: df2.query('converted == 1').shape[0] / df2.shape[0]
```

```
Out[82]: 0.11969708724499028
```

b. 假设一个用户处于 control 组中，他的转化率约为 0.06

```
In [83]: df_control = df2.query('group == "control"')
df_control.query('converted == 1').shape[0] / df2.shape[0]
```

```
Out[83]: 0.0603595901417834
```

c. 假设一个用户处于 treatment 组中，他的转化率约为 0.0594

```
In [84]: df_treatment = df2.query('group == "treatment"')
df_treatment.query('converted == 1').shape[0] / df2.shape[0]
```

```
Out[84]: 0.059413923381794
```

疑问

d. 因为网站的新旧页面是随机分发给用户的（为模拟真实场景的随机性），那么一个用户收到新页面的概率是多少？

```
In [85]: df2.user_id['landing_page == "new_page"'].shape[0] / df2.shape[0]
```

```
Out[85]: 0.5000019442220688
```

结果非常接近50%，因为有近三十万条的样本量，所以样本量保证了随机且独立，那我们就可以直观的认为一个用户不论收到新旧页面的先验概率都为50%

到现在为止，我现在还不能说明究竟是新页面可以让用户更可能的付费，或者是旧页面

下面我将开始运用假定性检验来验证新页面是否有效果

1. 现在，我需要先做出假设

下面我将给出我的原假设与备择假设

$H_0: p_{old} = p_{new}$

$H_1: p_{old} < p_{new}$

2. 假设在零假设中，不管看到页面还是旧页面， p_{new} 和 p_{old} 都具有等于转化成功率的“真成功率”，也就是说， p_{new} 与 p_{old} 是相等的。此外，假设它们等于原数据的转化率，新旧页面都是如此。

我需要执行两次决策之间转化率的抽样分布，计算零假设中10000次迭代的估计值。

a. 在零假设中， p_{new} 的 convert rate（转化率）为

```
In [86]: df2.query('converted == 1').shape[0] / df2.shape[0]
```

```
Out[86]: 0.11969708724499028
```

b. 在零假设中， p_{old} 的 convert rate（转化率）为

```
In [87]: df2.query('converted == 1').shape[0] / df2.shape[0]
```

```
Out[87]: 0.11969708724499028
```

c. n_{new} 为

```
In [88]: df2.query('landing_page == "new_page"').shape[0]
```

```
Out[88]: 145310
```

d. n_{old} 为

```
In [89]: df2.query('landing_page == "old_page"').shape[0]
```

```
Out[89]: 145274
```

e. 在零假设中，使用 p_{new} 转化率模拟 n_{new} 交易，并将这些 n_{new} 1's 与 0's 存储在 new_page_converted 中，也就是说对新页面与零数据相等的转化（模拟），来自动生成 0 和 1，0代表未转化，1代表转化

```
In [90]: p = df2.query('converted == 1').shape[0] / df2.shape[0]
df2_new = df2.query('landing_page == "new_page"')
new_page_converted = np.random.choice(2, df2_new.shape[0], p=[1-p, p])
old_page_converted
```

```
Out[90]: array([0, 0, ..., 0, 0, 0])
```

f. 在零假设中，使用 p_{old} 转化率模拟 n_{old} 交易，并将这些 n_{old} 1's 与 0's 存储在 old_page_converted 中。（同理，这又是旧页面模拟）

```
In [91]: df2_old = df2.query('landing_page == "old_page"')
old_page_converted = np.random.choice(2, df2_old.shape[0], p=[1-p, p])
old_page_converted
```

```
Out[91]: array([0, 0, ..., 0, 0, 0])
```

g 计算 $p_{new} - p_{old}$ 的模拟值

```
In [92]: (new_page_converted == 1).sum() - (old_page_converted == 1).sum()
```


```
Out[92]: -0.001598923200194193
```

h. 下载我调用上面的流程，用同样的方式为随机抽样，计算10,000个 $p_{new} - p_{old}$ 值，并将这 10,000 个值存储在 p_diffs 列表中

```
In [93]: p_diffs = []
for i in range(10000):
    new_page_converted = np.random.choice(2, df2_new.shape[0], p=[1-p, p])
    old_page_converted = np.random.choice(2, df2_old.shape[0], p=[1-p, p])
    p_diffs.append((new_page_converted == 1).sum() - (old_page_converted == 1).sum())
```

i. 绘制一个 p_diffs 直方图。

```
In [95]: plt.hist(p_diffs)
```



j. 我想知道在我模拟的零假设下的10,000个 $p_{new} - p_{old}$ 的差值所形成的正态分布图中，原数据的 $p_{old} - p_{new}$ 在此分布中处在什么位置，所模拟的图像中有多大的比例的大于原数据实际得到的值

```
In [96]: plt.hist(df_treatment.query('converted == 1').shape[0]/df_treatment.shape[0]-df_control.query('converted == 1').shape[0]/df_control.shape[0], color='red');
```



红线为原数据中 $p_{old} - p_{new}$ 的值

```
In [97]: p0 = (new_page_converted == 1).sum() - (old_page_converted == 1).sum()
p_diff = np.array(p_diffs)
(p_diff > p0).sum()
```

```
Out[97]: 0.9010000000000001
```

得出的p值为0.9011，所以不能拒绝原假设，即 $p_{old} = p_{new}$ ，所以转化率在新旧页面无差别

i. 我们也可以用使用一个内置程序（built-in）来实施类似的结果，尽管使用内置程序可能便于编码写码，但上面的内容是对正确实施统计显著性至关重要的原理的一个提醒， n_{old} 与 n_{new} 分别引证与旧页面和新页面关联的行数。下面是计算个页面的转化次数，以及每个页面的访问人数。

```
In [98]: df2.head()
```

```
Out[98]:
```

	user_id	timestamp	group	landing_page	converted
0	851104	2017-01-21 22:11:48.556739	control	old_page	0
1	804228	2017-01-12 08:01:45.159739	control	old_page	0
2	661590	2017-01-11 16:55:06.154213	treatment	new_page	0

```
In [99]: df_old = df2.query('landing_page == "old_page"')
df_new = df2.query('landing_page == "new_page"')
```

```
In [100]: import statsmodels.api as sm
convert_old = df_old.query('converted == 1').shape[0]
convert_new = df_new.query('converted == 1').shape[0]
n_old = df_old.shape[0]
n_new = df_new.shape[0]
```

m. 我现在使用 stats.proportions_ztest 来计算我的检验统计量与 p 值。

```
In [101]: z_scores, p_value = sm.stats.proportions_ztest([convert_old, convert_new], [n_old, n_new], alternative='smaller')
```

```
In [102]: z_scores, p_value
```

```
Out[102]: (1.3109241984234794, 0.9059881275903449)
```

```
In [103]: from scipy.stats import norm
norm.cdf(z_scores)
norm.ppf(1 - 0.05)
```

```
Out[103]: 1.6448536269514722
```

得到p值的值为p值，p值是在原假设认为真的情况下（观察到该数量或支持对假设更加极端）的概率，p值越大不能拒绝原假设，p值与错误概率α通常用来判断是否拒绝原假设或采纳对假设，在新旧页面中无区别

III - 回归分析法之一

1.

a. 同样的，我将尝试使用逻辑回归来对数据进行预测。

b. 使用 statsmodels 并拟合逻辑回归模型，以查看用户收到的不同页面是否存在显著的变化趋势。但是，首先，我需要为这个数据创建一个列，并为每个用户收到的页面建立一个虚拟变量列。添加一个截距列，一个 ab_page 列，当用户接收 treatment 时为1，control 时为0。

```
In [107]: df2['intercept'] = 1
df2['control', 'ab_page'] = pd.get_dummies(df['group'])
```

```
In [108]: df2 = df2.drop(['control'], axis = 1)
df2.head()
```

```
Out[108]:
```

	user_id	timestamp	group	landing_page	converted	intercept	ab_page
0	851104	2017-01-21 22:11:48.556739	control	old_page	0	1	0
1	804228	2017-01-12 08:01:45.159739	control	old_page	0	1	0
2	661590	2017-01-11 16:55:06.154213	treatment	new_page	0	1	1
3	853541	2017-01-08 18:28:03.143765	treatment	new_page	0	1	1
4	864975	2017-01-21 01:52:26.210827	control	old_page	1	1	0

c. 使用 statsmodels 导入我的逻辑模型，实例化该模型，使用 b. 中创建的两个虚拟变量模型，用来预测一个用户是否会发生转化。

```
In [109]: # 创建并拟合逻辑模型
logit_model = sm.Logit(df2['converted'], df2[['intercept', 'ab_page']])
```

d. 请在下方提供你的模型摘要，并详细说明使用它来解决下面的问题。

```
In [110]: # 运行命令，查看模型摘要
results = logit_model.fit()
results.summary()
```

Optimization terminated successfully.
Current function value: 0.369118
Iterations: 6

```
Out[110]:
```

Dep. Variable:	converted	No. Observations:
Model:	Logit	290584
Method:	MLE	DF Residuals:
Date:	Mon, 26 Nov 2018	DF Model:
Time:	05:17:45	Pseudo R-squ:
converged:	True	Log-Likelihood:
		LL-Null:
		LLR p-value:

	coef	std err	z	P> z	[0.025	0.975]
intercept	-1.9888	0.008	-246.869	0.000	-2.005	-1.973
ab_page	-0.0150	0.011	-1.311	0.190	-0.037	0.007

与 ab_page 关联的p值为0.19，在 0 中我的零假设与备择假设分别为

$H_0: p_{old} = p_{new}$

$H_1: p_{old} < p_{new}$

而 III 中的零假设与备择假设分别为

$H_0: p_{old} = p_{new}$

$H_1: p_{old} > p_{new}$

而 III 中的零假设不同，从而方向性不同，所以 p 值出现两种情况

f. 好的，之前我只是单纯的将转化次数与页面数据放到分类模型中，但并未放入其他的变量（条件），假设用户发生行为与其他的变量因素有关呢？城市、国家、种族等等。

g. 现在，除了测试不同页面的转化率是否发生变化之外，我还想将用户居住的国家或地区添加一个 effect 项。

```
In [112]: country_df = pd.read_csv('country.csv')
df_new = country_df.set_index('user_id').join(df2.set_index('user_id'), how='inner')
```

```
In [113]: df_new.head()
```

```
Out[113]:
```

	country	timestamp	group	landing_page	converted	intercept	ab_page
user_id							
834778	UK	2017-01-14 23:08:43.304998	control	old_page	0	1	0
928468	US	2017-01-23 14:44:16.387854	treatment	new_page	0	1	1
822059	UK	2017-01-16 14:04:14.191771	treatment	new_page	1	1	1
711587	UK	2017-01-22 03:14:24.763511	control	old_page	0	1	0
710616	UK	2017-01-16 13:14:44.000513	treatment	new_page	0	1	1

这次的数据事先处理过，所以我直接导入了 country 变量找到了假数据中

```
In [117]: # 查看下载网站注册下的三个国家的数据列表
df_new.country.value_counts()
```

```
Out[117]:
```

country	count
US	72499
CA	14499
UK	14958

```
In [114]: df_new[['CA', 'UK', 'US']] = pd.get_dummies(df_new['country'])
```

结构向之前的那样，建立逻辑回归模型

```
In [115]: logit_model = sm.Logit(df_new['converted'], df_new[['intercept', 'CA', 'UK']])
results = logit_model.fit()
results.summary()
```

Optimization terminated successfully.
Current function value: 0.369116
Iterations: 6

```
Out[115]:
```

Dep. Variable:	converted	No. Observations:
Model:	Logit	290584
Method:	MLE	DF Residuals:
Date:	Mon, 26 Nov 2018	DF Model:
Time:	05:38:06	Pseudo R-squ:
converged:	True	Log-Likelihood:
		LL-Null:
		LLR p-value:

	coef	std err	z	P> z	[0.025	0.975]
intercept	-1.9967	0.007	-292.314	0.000	-2.010	-1.983
CA	-0.0408	0.027	-1.518	0.129	-0.093	0.012
UK	0.0099	0.013	0.746	0.456	-0.016	0.036

结果显示于表格，CA 与 UK 两页 P 值都大于 alpha 水平，所以不能拒绝原假设，意味着不具备统计显著性，对转化无影响

h. 虽然我目前在已经查看国家与页面在转化率上的个体性影响，但现在我要看看国家与页面地区之间的相互作用，测试其是否会对转化率产生重大影响，在创建新的列时，并创建一个新的模型。所以我要将 new_page 与 CA、UK，分别相乘得到两个新变量放到分类器中

```
df2['new_CA'] = df2['new_page'] * df2['CA']
df2['new_UK'] = df2['new_page'] * df2['UK']
```

```
In [121]: # 分别创建 new_page_new_CA, new_UK
df_new[['new_page', 'new_CA', 'new_UK']] = pd.get_dummies(df_new[['landing_page', 'CA', 'UK']])
df_new[['new_CA'], 'new_UK'] = df_new[['new_page'] * df_new['CA'], 'new_UK']
```

```
In [122]: # 查看新行数据
df_new.head()
```

```
Out[122]:
```

	country	timestamp	group	landing_page	converted	intercept	ab_page	CA	UK	US	new_page	old_page	new_CA	new_UK
user_id														
834778	UK	2017-01-14 23:08:43.304998	control	old_page	0	1	0	0	1	0	0	1	0	0
928468	US	2017-01-23 14:44:16.387854	treatment	new_page	0	1	1	0	0	1	1	0	0	0
822059	UK	2017-01-16 14:04:14.191771	treatment	new_page	1	1	1	0	1	0	1	0	0	1
711587	UK	2017-01-22 03:14:24.763511	control	old_page	0	1	0	0	1	0	0	1	0	0
710616	UK	2017-01-16 13:14:44.000513	treatment	new_page	0	1	1	0	1	0	1	0	0	1

```
In [123]: logit_model = sm.Logit(df_new['converted'], df_new[['intercept', 'new_CA', 'new_UK', 'ab_page', 'CA', 'UK']])
results = logit_model.fit()
results.summary()
```

Optimization terminated successfully.
Current function value: 0.369109
Iterations: 6

```
Out[123]:
```

Dep. Variable:	converted	No. Observations:
Model:	Logit	290578
Method:	MLE	DF Residuals:
Date:	Mon, 26 Nov 2018	DF Model:
Time:	05:49:08	Pseudo R-squ:
converged:	True	Log-Likelihood:
		LL-Null:
		LLR p-value:

	coef	std err	z	P> z	[0.025	0.975]
intercept	-1.9955	0.010	-206.534	0.000	-2.005	-1.985
new_CA	-0.0469	0.024	-0.872	0.383	-0.102	0.009
new_UK	0.0014	0.027	1.181	0.238	-0.021	0.024
ab_page	-0.0206	0.014	-1.505	0.132	-0.047	0.006
CA	-0.0178	0.038	-0.465	0.642	-0.091	0.056
UK	-0.0557	0.019	-0.306	0.760	-0.043	0.031

根据所得结果 p 值大于 alpha 水平，不能拒绝原假设，页面与国家地区之间相互作用不明显

```
In [11]: import pandas
df = pd.read_csv('data.csv')
```

```
Out[11]:
```

	user_id	timestamp	group	landing_page	converted
0	851104	2017-01-21 22:11:48.556739	control	old_page	0
1	804228	2017-01-12 08:01:45.159739	control	old_page	0
2	661590	2017-01-11 16:55:06.154213	treatment	new_page	0