Pruebas A/B: Una guía paso a paso en Python

Del diseño experimental a la comprobación de hipótesis

Renato Fillinich · Seguir
Publicado en Hacia la Ciencia de Datos · 9 min de lectura · 28 de mayo de 2020

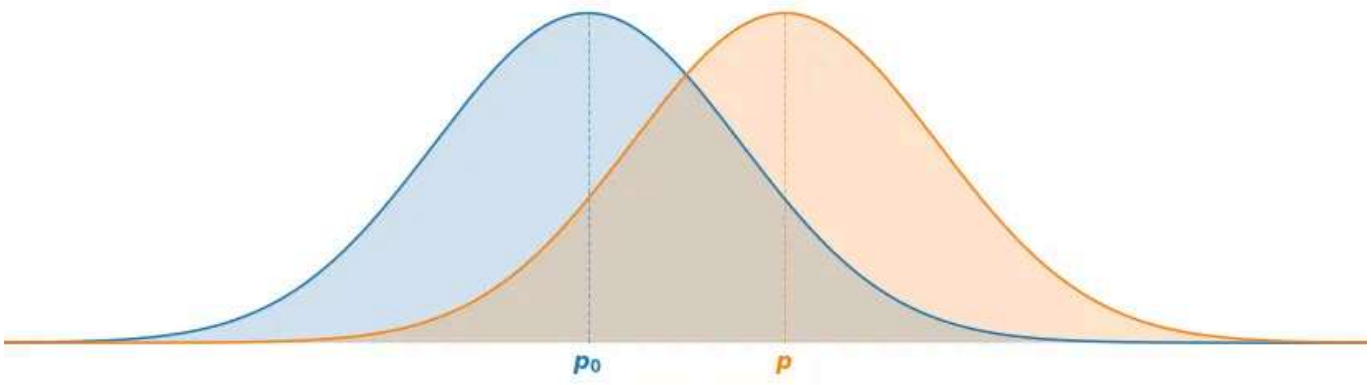


Imagen del autor

En este artículo repasaremos el proceso de análisis de un experimento A/B, desde la formulación de una hipótesis, su prueba y, finalmente, la interpretación de los resultados. Para nuestros datos, utilizaremos un conjunto de datos de Kaggle que contiene los resultados de una prueba A/B en lo que parecen ser 2 diseños diferentes de una página web (old_page frente a new_page). Si quieres seguir el código que utilicé, no dudes en descargar el cuaderno de Jupyter en mi página de GitHub.

Esto es lo que haremos:

1. Diseñando nuestro experimento
2. Recopilación y preparación de los datos
3. Visualización de los resultados
4. Comprobación de la hipótesis
5. Sacando conclusiones

Para hacerlo un poco más realista, aquí hay un **escenario** potencial para nuestro estudio:

> *Imaginemos que trabajas en el equipo de producto de una* ***empresa de comercio electrónico online*** *de tamaño medio. El diseñador de UX trabajó muy duro en una nueva versión de la página del producto, con la esperanza de que conduzca a una mayor tasa de conversión. El gerente de producto (PM) le dijo que la* ***tasa de conversión actual*** *es de aproximadamente el* ***13%*** *en promedio durante todo el año, y que el equipo estaría contento con un* ***aumento del 2%,*** *lo que significa que el nuevo diseño se considerará un éxito si eleva la tasa de conversión al 15%.*

Antes de implementar el cambio, el equipo se sentiría más cómodo probándolo en un pequeño número de usuarios para ver cómo funciona, por lo que sugiere ejecutar una **prueba A/B** en un subconjunto de los usuarios de la base de usuarios.

1. Diseño de nuestro experimento

Formulación de una hipótesis

Lo primero es lo primero, queremos asegurarnos de formular una hipótesis al comienzo de nuestro proyecto. Esto asegurará que nuestra interpretación de los resultados sea correcta y rigurosa.

Dado que no sabemos si el nuevo diseño funcionará mejor o peor (¿o igual?) que nuestro diseño actual, elegiremos una **prueba de dos colas**:

$$H_o: p = p_o$$

$$H_a: p \neq p_o$$

donde p y p_o representan la tasa de conversión del nuevo y antiguo diseño, respectivamente. También estableceremos un **nivel de confianza del 95%**:

$$\alpha = 0.05$$

El valor **α** es un umbral que establecemos, mediante el cual decimos "si la probabilidad de observar un resultado tan extremo o mayor (**valor *p***) es inferior **a α**, entonces rechazamos la hipótesis nula". Dado que nuestro **α=0,05** (que indica un 5% de probabilidad), nuestra confianza (1 — **α**) es del 95%.

No se preocupe si no está familiarizado con lo anterior, todo lo que esto realmente significa es que cualquiera que sea la tasa de conversión que

observemos para nuestro nuevo diseño en nuestra prueba, queremos estar 95% seguros de que es estadísticamente diferente de la tasa de conversión de nuestro diseño anterior, antes de decidir rechazar la hipótesis nula H-.

Elección de las variables

Para nuestra prueba necesitaremos **dos grupos**:

- Un grupo: se les mostrará el diseño anterior `control`
- Un grupo (o experimental) - Se les mostrará el nuevo diseño `treatment`

Esta será nuestra *Variable Independiente*. La razón por la que tenemos dos grupos a pesar de que conocemos la tasa de conversión de referencia es que queremos controlar otras variables que podrían tener un efecto en nuestros resultados, como la estacionalidad: al tener un grupo podemos comparar directamente sus resultados con el grupo, porque la única diferencia sistemática entre los grupos es el diseño de la página del producto, y, por lo tanto, podemos atribuir cualquier diferencia en los resultados a los diseños. `control treatment`

Para nuestra *variable dependiente* (es decir, lo que estamos tratando de medir), estamos interesados en capturar el . Una forma en que podemos

codificar esto es mediante cada sesión de usuario con una variable binaria: `conversion rate`

- `0` - El usuario no compró el producto durante esta sesión de usuario
- `1` - El usuario compró el producto durante esta sesión de usuario

De esta forma, podemos calcular fácilmente la media de cada grupo para obtener la tasa de conversión de cada diseño.

Elección de un tamaño de muestra

Es importante tener en cuenta que, dado que no probaremos toda la base de usuarios (nuestra población), las tasas de conversión que obtendremos inevitablemente serán solo *estimaciones* de las tasas reales.

El número de personas (o sesiones de usuario) que decidamos capturar en cada grupo tendrá un efecto en la precisión de nuestras tasas de conversión estimadas: **cuanto mayor sea el tamaño de la muestra**, más precisas sean nuestras estimaciones (es decir, cuanto menores sean nuestros intervalos de confianza), **mayor será la posibilidad de detectar una diferencia** en los dos grupos, si está presente.

Por otro lado, cuanto más grande es nuestra muestra, más caro (y poco práctico) se vuelve nuestro estudio.

Entonces, ¿cuántas personas debemos tener en cada grupo?

El tamaño de la muestra que necesitamos se estima a través de algo llamado *análisis de potencia*, y depende de algunos factores:

- **Potencia de la prueba** (1 — β): representa la probabilidad de encontrar una diferencia estadística entre los grupos de nuestra prueba cuando realmente hay una diferencia. Por lo general, esto se establece en 0.8 por convención (aquí hay más información sobre el poder estadístico, si tiene curiosidad)
- **Valor alfa** (α): el valor crítico que establecimos anteriormente en 0,05
- **Tamaño del efecto:** qué tan grande esperamos que haya una diferencia entre las tasas de conversión

Dado que nuestro equipo estaría satisfecho con una diferencia del 2%, podemos usar el 13% y el 15% para calcular el tamaño del efecto que esperamos.

Por suerte, **Python se encarga de todos estos cálculos por nosotros:**

```
# Packages imports
import numpy as np
import pandas as pd
import scipy.stats as stats
import statsmodels.stats.api as sms
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
from math import ceil

%matplotlib inline

# Some plot styling preferences
plt.style.use('seaborn-whitegrid')
font = {'family' : 'Helvetica',
        'weight' : 'bold',
        'size'   : 14}

mpl.rc('font', **font)

effect_size = sms.proportion_effectsize(0.13, 0.15)    # Calculating
effect size based on our expected rates

required_n = sms.NormalIndPower().solve_power(
    effect_size,
    power=0.8,
    alpha=0.05,
    ratio=1
    )                                                  # Calculating
sample size needed

required_n = ceil(required_n)                          # Rounding up
to next whole number
```

```
print(required_n)
```

```
4720
```

Necesitaríamos **al menos 4720 observaciones para cada grupo.**

Haber establecido el parámetro en 0,8 en la práctica significa que si existe una diferencia real en la tasa de conversión entre nuestros diseños, suponiendo que la diferencia sea la que estimamos (13% frente a 15%), tenemos aproximadamente un 80% de posibilidades de detectarla como estadísticamente significativa en nuestra prueba con el tamaño de muestra que calculamos. `power`

2. Recopilación y preparación de los datos

¡Genial! Así que ahora que tenemos el tamaño de muestra requerido, necesitamos recopilar los datos. Por lo general, en este punto, trabajaría con su equipo para configurar el experimento, probablemente con la ayuda del

equipo de ingeniería, y se aseguraría de recopilar suficientes datos en función del tamaño de muestra necesario.

Sin embargo, dado que usaremos un conjunto de datos que encontramos en línea, para simular esta situación:

1. Descargar el conjunto de datos de Kaggle
2. Leer los datos en un DataFrame de pandas
3. Compruebe y limpie los datos según sea necesario
4. Muestree aleatoriamente las filas del DataFrame para cada grupo
    * `n=4720`

*__Nota:__ Normalmente, no necesitaríamos realizar el paso 4, esto es solo por el bien del ejercicio

Como ya descargué el conjunto de datos, iré directamente al número 2.

```
df = pd.read_csv('ab_data.csv')
df.head()
```

	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

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 294478 entries, 0 to 294477
Data columns (total 5 columns):
 #   Column        Non-Null Count   Dtype 
---  ------        --------------   ----- 
 0   user_id       294478 non-null  int64 
 1   timestamp     294478 non-null  object
 2   group         294478 non-null  object
 3   landing_page  294478 non-null  object
 4   converted     294478 non-null  int64 
dtypes: int64(2), object(3)
memory usage: 11.2+ MB
```

```
# To make sure all the control group are seeing the old page and
viceversa

pd.crosstab(df['group'], df['landing_page'])
```

landing_page group	new_page	old_page
control	1928	145274
treatment	145311	1965

Hay **294478 filas** en el DataFrame, cada una de las cuales representa una sesión de usuario, así como **5 columnas**:

- `user_id` - El ID de usuario de cada sesión
- `timestamp` - Marca de tiempo de la sesión
- `group` - A qué grupo se asignó el usuario para esa sesión {, `control treatment` }
- `landing_page` - Qué diseño vio cada usuario en esa sesión {, `old_page new_page` }
- `converted` - Si la sesión terminó en una conversión o no (binario, =no convertido, =convertido) `0 1`

En realidad, solo usaremos las columnas y para el análisis. `group converted`

Antes de seguir adelante y muestrear los datos para obtener nuestro subconjunto, asegurémonos de que no haya usuarios que se hayan muestreado varias veces.

```
session_counts = df['user_id'].value_counts(ascending=False)
multi_users = session_counts[session_counts > 1].count()

print(f'There are {multi_users} users that appear multiple times in
the dataset')
```

```
There are 3894 users that appear multiple times in the dataset
```

De hecho, hay 3894 usuarios que aparecen más de una vez. Dado que el número es bastante bajo, seguiremos adelante y los eliminaremos del DataFrame para evitar muestrear a los mismos usuarios dos veces.

```
users_to_drop = session_counts[session_counts > 1].index

df = df[~df['user_id'].isin(users_to_drop)]
print(f'The updated dataset now has {df.shape[0]} entries')
```

```
The updated dataset now has 286690 entries
```

Muestreo

Ahora que nuestro DataFrame está limpio y agradable, podemos continuar y muestrear las entradas de cada uno de los grupos. Podemos usar el método de los pandas para hacer esto, que realizará un muestreo aleatorio simple por nosotros. `n=4720 DataFrame.sample()`

Nota: Lo he configurado para que los resultados sean reproducibles si te apetece seguirlo en tu propio Notebook: sólo tienes que utilizarlo en tu función y deberías obtener la misma muestra que yo. `random_state=22 random_state=22`

```
control_sample = df[df['group'] == 'control'].sample(n=required_n,
random_state=22)
treatment_sample = df[df['group'] ==
'treatment'].sample(n=required_n, random_state=22)

ab_test = pd.concat([control_sample, treatment_sample], axis=0)
ab_test.reset_index(drop=True, inplace=True)

ab_test
```

	user_id	timestamp	group	landing_page	converted
0	763854	2017-01-21 03:43:17.188315	control	old_page	0
1	690555	2017-01-18 06:38:13.079449	control	old_page	0
2	861520	2017-01-06 21:13:40.044766	control	old_page	0
3	630778	2017-01-05 16:42:36.995204	control	old_page	0
4	656634	2017-01-04 15:31:21.676130	control	old_page	0
...	...	...	...	...	...
9435	908512	2017-01-14 22:02:29.922674	treatment	new_page	0
9436	873211	2017-01-05 00:57:16.167151	treatment	new_page	0
9437	631276	2017-01-20 18:56:58.167809	treatment	new_page	0
9438	662301	2017-01-03 08:10:57.768806	treatment	new_page	0
9439	944623	2017-01-19 10:56:01.648653	treatment	new_page	1

9440 rows × 5 columns

```
ab_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9440 entries, 0 to 9439
Data columns (total 5 columns):
# Column Non-Null Count Dtype
--- ------ -------------- -----
0 user_id 9440 non-null int64
```

```
1 timestamp 9440 non-null object
2 group 9440 non-null object
3 landing_page 9440 non-null object
4 converted 9440 non-null int64
dtypes: int64(2), object(3)
memory usage: 368.9+ KB
```

```
ab_test['group'].value_counts()
```

```
control 4720
treatment 4720
Name: group, dtype: int64
```

Genial, parece que todo salió según lo planeado, y ahora estamos listos para analizar nuestros resultados.

3. Visualización de los resultados

Lo primero que podemos hacer es calcular algunas **estadísticas básicas** para hacernos una idea de cómo son nuestras muestras.

```
conversion_rates = ab_test.groupby('group')['converted']

std_p = lambda x: np.std(x, ddof=0)              # Std. deviation of
the proportion
se_p = lambda x: stats.sem(x, ddof=0)            # Std. error of the
proportion (std / sqrt(n))

conversion_rates = conversion_rates.agg([np.mean, std_p, se_p])
conversion_rates.columns = ['conversion_rate', 'std_deviation',
'std_error']


conversion_rates.style.format('{:.3f}')
```

group	conversion_rate	std_deviation	std_error
control	0.123	0.329	0.005
treatment	0.126	0.331	0.005

A juzgar por las estadísticas anteriores, parece que **nuestros dos diseños tuvieron un rendimiento muy similar**, con nuestro nuevo diseño

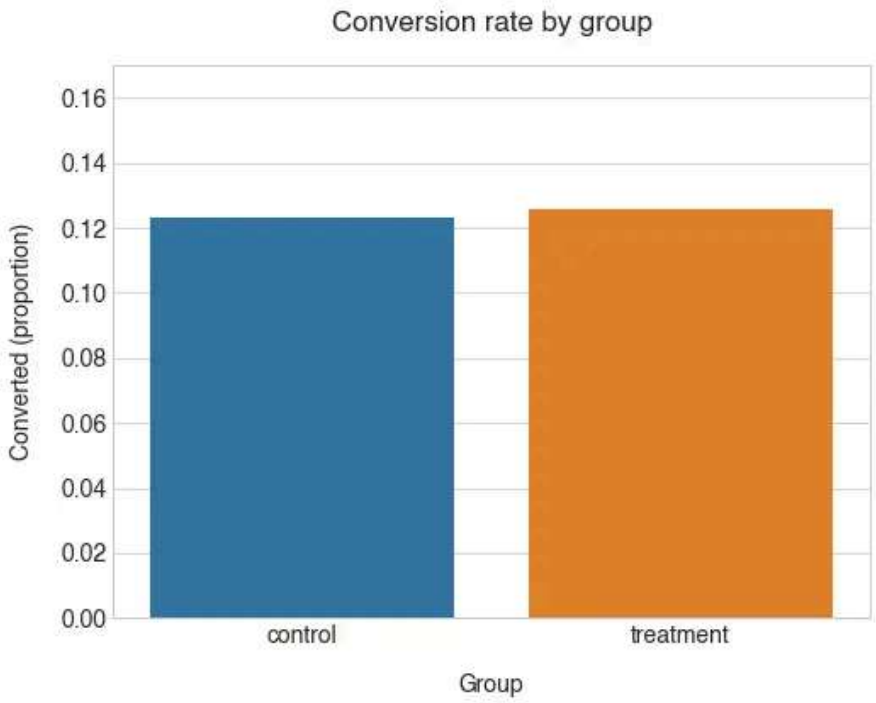
funcionando ligeramente mejor, aproximadamente **un 12,3% frente a una tasa de conversión del 12,6%.**

El trazado de los datos hará que estos resultados sean más fáciles de comprender:

```
plt.figure(figsize=(8,6))

sns.barplot(x=ab_test['group'], y=ab_test['converted'], ci=False)

plt.ylim(0, 0.17)
plt.title('Conversion rate by group', pad=20)
plt.xlabel('Group', labelpad=15)
plt.ylabel('Converted (proportion)', labelpad=15);
```



Las tasas de conversión de nuestros grupos son realmente muy similares. También hay que tener en cuenta que la tasa de conversión del grupo es más baja de lo que hubiéramos esperado dado lo que sabíamos sobre nuestra tasa de conversión media (12,3% frente a 13%). Esto demuestra que hay cierta variación en los resultados cuando se toman muestras de una población. `control`

Así que... El valor del grupo es mayor. **¿Es esta diferencia *estadísticamente significativa*?** `treatment`

4. Comprobación de la hipótesis

El último paso de nuestro análisis es probar nuestra hipótesis. Dado que tenemos una muestra muy grande, podemos usar la aproximación normal para calcular nuestro valor *p* (es decir, la prueba z).

Una vez más, Python hace que todos los cálculos sean muy fáciles. Podemos usar el módulo para obtener el valor *p* y los intervalos de confianza: `statsmodels.stats.proportion`

```
from statsmodels.stats.proportion import proportions_ztest,
proportion_confint

control_results = ab_test[ab_test['group'] == 'control']['converted']
treatment_results = ab_test[ab_test['group'] == 'treatment']
['converted']

n_con = control_results.count()
n_treat = treatment_results.count()
successes = [control_results.sum(), treatment_results.sum()]
nobs = [n_con, n_treat]

z_stat, pval = proportions_ztest(successes, nobs=nobs)
(lower_con, lower_treat), (upper_con, upper_treat) =
proportion_confint(successes, nobs=nobs, alpha=0.05)
```

```
print(f'z statistic: {z_stat:.2f}')
print(f'p-value: {pval:.3f}')
print(f'ci 95% for control group: [{lower_con:.3f},
{upper_con:.3f}]')
print(f'ci 95% for treatment group: [{lower_treat:.3f},
{upper_treat:.3f}]')
```

```
z statistic: -0.34

p-value: 0.732

ci 95% for control group: [0.114, 0.133]

ci 95% for treatment group: [0.116, 0.135]
```

5. Sacar conclusiones

Dado que nuestro **valor *p* = 0.732 está muy por encima de nuestro umbral de α = 0.05**, no podemos rechazar la hipótesis nula H-, lo que significa que **nuestro nuevo diseño no funcionó significativamente diferente (y mucho menos mejor) que el anterior :(**

Además, si nos fijamos en el intervalo de confianza del grupo ([0,116, 0,135], o 11,6-13,5%) observamos que: `treatment`

1. Incluye nuestro valor de referencia de una tasa de conversión del 13%
2. No incluye nuestro valor objetivo del 15 % (el aumento del 2 % que buscábamos)

Lo que esto significa es que es más probable que la verdadera tasa de conversión del nuevo diseño sea similar a nuestra línea de base, en lugar del objetivo del 15% que esperábamos. Esta es una prueba más de que no es probable que nuestro nuevo diseño sea una mejora de nuestro diseño anterior, y que desafortunadamente hemos vuelto a la mesa de dibujo.

> *¿Te ha gustado mi historia? ¡Por favor, hágamelo saber!*
>
> *Y, por favor, siéntase libre de descargar el cuaderno de Jupyter en mi página de GitHub.*

Investigación de usuarios | Pruebas de abdominales | UX | Pitón | Prueba de hipótesis