Analyzing AB Test Result For An eCommerce Website

March 9, 2018

1 Project Title: Analyzing A/B Test Results for an eCommerce Website.

1.1 Table of Contents

Introduction

A/B tests are very commonly performed by data analysts and data scientists. As a Data Analyst ,its important to spend a good time practicing and working with the difficulties , so as to ensure the necessary skills are mastered . By measuring the impact that the changes have on the metrics, A/B test ensures that the introduced change , that produces positive results are retained .In a simpler terms ,you will not be fooled by chance in accepting the alternative, unless the A/B test indicates that the variability is due to the alternative .

Another reason why A/B testing is so important for e-commerce websites is because conversion can directly be measured and could be related to a specific metric such as direct sale ,revenue etc...The testing types can broadly be catagorized based on the test area or sector of the web page investigated . The four major catacories are :- - Call to action button - Pricing , discounts or shipping - Product display - Check out pages .

Some examples of A/B testing adopted in diffrent industries include : - Testing two soil treatments to determine which produces better seed germination(*Natural Sciences*) - Testing two therapies to determine which suppresses cancer more effectively(*Medicine*) . - Testing two prices to determine which yields more net profit(*Business /Financial Sectors*). - Testing two web headlines to determine which produces more clicks (*Marketing*). etc...

For this project, an A/B test run by an e-commerce website was provided . The project goal is to perform statistical analysis on the data provided and help the company understand if they should implement the new page, or keep the old page, or perhaps run the experiment longer before making decision.

Source :** Practical Statistics for Data Scientists** '*Chapter 3: Statistical Experiments and Significance Testing*

Part I - Probability

** `Importing the required  libraries.`**

```
In [1]: import pandas as pd
        import numpy as np
```

```
import random
import matplotlib.pyplot as plt

%matplotlib inline

random.seed(42)
```

** 1a: Loading and checking the dataset.**

In [2]:
```
# Load the data set
df=pd.read_csv('ab_data.csv')
```

In [3]:
```
# checking some rows of the dataframe
df.head()
```

Out[3]:

	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

In [4]:
```
df.info()
```

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

** 1b: Number of rows in the dataset. **

In [5]:
```
df.shape
```

Out[5]: (294478, 5)

** 1c:The number of unique users in the dataset **

In [6]:
```
# number of unique id
a=df.user_id.nunique()
a
```

Out[6]: 290584

1d: The proportion of users converted.

```
In [7]: #Proportion  unique viewer(user id )  converted
        df1=df.query ('converted== 1').count ()
        P_User_conv=df1/df['converted'].count()
        P_User_conv.converted
```

```
Out[7]: 0.11965919355605512
```

** 1e: The number of times the new_page and treatment don't line up **

```
In [8]: # Number of times the treatment and new page  don't line up
        df_treat_dif=df[((df['group'] == 'treatment') == (df['landing_page'] == 'new_page')) ==
        #Number of times the control doesn't line up with old page
        df_cont_dif=df[((df['group'] == 'control') == (df['landing_page'] == 'old_page')) == Fa
        a=df_treat_dif.user_id.count()
        b=df_cont_dif.user_id.count()
        a,b
```

```
Out[8]: (3893, 3893)
```

** 1f: Missing values **

```
In [9]: # checking for null values
        df.info()
        print ('No null values')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 294478 entries, 0 to 294477
Data columns (total 5 columns):
user_id         294478 non-null int64
timestamp       294478 non-null object
group           294478 non-null object
landing_page    294478 non-null object
converted       294478 non-null int64
dtypes: int64(2), object(3)
memory usage: 11.2+ MB
No null values
```

** 2: Checking for inconsistencies like the rows where treatment is not aligned with new_page or control is not aligned with old_page **

```
In [10]: #  Extract rows having treatment as group
         df1=df[df['group'] == 'treatment']
         # Extract from df1 where the new page is the landing page
         df2=df1[df1['landing_page'] == 'new_page']
```

```
In [11]: # Double Check all of the correct rows were removed - this should be 0
         df2[((df2['group'] == 'treatment') == (df2['landing_page'] == 'new_page')) == False].s
```

Out[11]: 0

In [12]:
```
# Extract rows having control as a  group
df_ctrl=df[df['group'] == 'control']

# Extract from df1 where the old page as landing page
df_ctrl=df_ctrl[df_ctrl['landing_page'] == 'old_page']
sum(df_ctrl.user_id.duplicated())
```

Out[12]: 0

** 3a:Unique user_ids in df2 **

In [13]:
```
# Unique user ids
df2.user_id.nunique()
```

Out[13]: 145310

** Checking for duplicates **
** 3b:User_id repeated **

In [14]:
```
# duplicated user id
df2[df2['user_id'].duplicated ()==True]['user_id']
```

Out[14]:
```
2893    773192
Name: user_id, dtype: int64
```

** 3c:Row information for the repeated user_id **

In [15]:
```
#Row with duplicated data
df2[df2['user_id'].duplicated ()==True]
```

Out[15]:

	user_id	timestamp	group	landing_page	converted
2893	773192	2017-01-14 02:55:59.590927	treatment	new_page	0

** 3d: Removing the duplicate user_id **

In [16]:
```
# drop one of the duplicated row
df2=df2[df2.index != 2893]
```

In [17]:
```
#checking if the  duplicated row is removed .
sum(df2.user_id.duplicated())
```

Out[17]: 0

In [18]:
```
#Extract the correct controlgroup data
df3=df[df['group'] == 'control']
df4=df3[df3['landing_page'] == 'old_page']
```

In [72]:
```
df_cleaned= df2.append(df4)
df_cleaned.head(1)
```

Out[72]:

	user_id	timestamp	group	landing_page	converted
2	661590	2017-01-11 16:55:06.154213	treatment	new_page	0

** 4a: Probability of an individual converting regardless of the page view. **

In [20]:
```
# total  page view
Total_view =df_cleaned.user_id.count()

#converted
Total_converted = df_cleaned.query('converted == 1').count()

#Probability of being  converted
P_ttl_cnvrtd= Total_converted.user_id/Total_view
P_ttl_cnvrtd
```

Out[20]: 0.11959708724499628

** 4b: The probability viewer converted ,given that an individual was in the control group **

In [21]:
```


#Total controls
ctrl_ttl=df4['group'].count()

# Converted controls
ctrl_cnvrtd=df4.query('converted==1').count()

#probability control coud be converted
P_ctrl_cnvrtd= ctrl_cnvrtd / ctrl_ttl
P_ctrl_cnvrtd['converted']
```

Out[21]: 0.1203863045004612

** 4C:The probability viewer converted,given that an individual was in the treatment group **

In [22]:
```
#Total treatment
trmnt_ttl=df2['group'].count()

# Converted
trmnt_cnvrtd=df2.query('converted==1').count()

#probability treatment coud be converted
P_trmnt_ttl_cnvrtd= trmnt_cnvrtd/trmnt_ttl
P_trmnt_ttl_cnvrtd.converted
```

Out[22]: 0.11880806551510564

```
In [23]: diffs_obsrvd=P_trmnt_ttl_cnvrtd.converted- P_ctrl_cnvrtd['converted']
         diffs_obsrvd
```

```
Out[23]: -0.0015782389853555567
```

** 4d:The probability that an individual received the new page **

```
In [24]: #number of individual landed on  new page
         df_treat_new=df2[((df2['group'] == 'treatment') == (df2['landing_page'] == 'new_page')

         # numbre of individual landed in any page (sum of control and treatment landing page)
         total_landing_page=df_cleaned.landing_page.count()

         # Proportion
         P_land_newpage=df_treat_new/ total_landing_page

         P_land_newpage.landing_page
```

```
Out[24]: 0.5000619442226688
```

** 4e ## **Discussion** The resulting probabilities DO NOT suggest that the new treatment page leads to a higher conversion of viewers .The probability for a viewer who landed on the old page get converted is 12.04% and that of a user viewing the new-page converted is 11.88% . Eventhough both pages has equal (50.01%) chance of being viewed by the user ,the control group has better conversion rate than the new treatment page.**

Furthermore , the treatment page has resulted in a relativiely lower probability of conversion (11.88%) as compared to the general probability of any viewer being converetd (which is 11.96%) .Therefore I would say there is no sufficient evidence to conclude that usage of the new treatment page will lead to a more viwer conversion.

Part II - A/B Test

1.1.1 Assumption :

I :Decision made just based only the data provided.

** II** : **Type I error rate 5%**

** III ** : **under the null hypothesis, p_{new} and p_{old} both have "true" success rates equal to the converted success rate regardless of page view that is p_{new} and p_{old} are equal.**

** IV** : **Under the null hypothesis, p_{new} and p_{old} , are equal to the convert rate in the ab_data.csv regardless of the page view .**

**** 1: Hypothesis ****

- Null Hypothesis (Ho): new page conversion rate is worse than or equal to old page.
- Alternative Hypothesis (H1) : the new page conversion rate is better than the old page.

$H_0 : CTR_{new} <= CTR_{old}$

$H_1 : CTRnew > CTR_{old}$

$H_0 : CTR_{new} - CTR_{old} <= 0$
$H_1 : CTRnew - CTR_{old} > 0$
** 2a: The convert rate for p_{new} under the null **

In [25]:
```
#Convert rate for treatment coverted/ total page view
df_trtmnt=df_cleaned[df_cleaned['group']=='treatment']
total_trtmnt=df_trtmnt.count()
convrtd_trtmnt = df_trtmnt[df_trtmnt['converted'] == 1].count()
P_new_cnvrt=convrtd_trtmnt/total_trtmnt
Pnew=P_new_cnvrt.group
Pnew
```

Out[25]: 0.11880806551510564

** 2b:The convert rate for p_{old} under the null **.

In [26]:
```
#under the null Pnew and Pold are equal
Pold=Pnew
```

In [27]:
```
# observed difference in click through rate
p_diff_obsrvd_null = Pnew - Pold

p_diff_obsrvd_null
```

Out[27]: 0.0

** 2c:** n_{new}

In [28]:
```
n_new =df2.nunique ()
n_new=n_new.user_id
n_new
```

Out[28]: 145310

** 2d : ** n_{old}

In [29]:
```
n_old =df4.nunique ()
n_old=n_old.user_id
n_old
```

Out[29]: 145274

** 2e: Simulating n_{new} transactions with a convert rate of p_{new} under the null. Store these n_{new} 1's and 0's in new_page_converted **

In [30]:
```
new_converted_smltn = np.random.binomial(1,Pnew ,n_new)
Pnew_smltn=new_converted_smltn.mean()
Pnew_smltn
```

Out[30]: 0.1194756038813571

** 2f : Simulating n_{old} transactions with a convert rate of p_{old} under the null. Store these n_{old} 1's and 0's in old_page_converted . **

```
In [31]: old_converted_smltn = np.random.binomial(1,Pold ,n_old )
         Pold_smltn=old_converted_smltn.mean()
         Pold_smltn
```

Out[31]: 0.11894076021862136

** 2g : Calculate the p_{new} - p_{old} for simulated values **.

```
In [32]: P_diff_smltn_obsrvd = Pnew_smltn - Pold_smltn
         P_diff_smltn_obsrvd
```

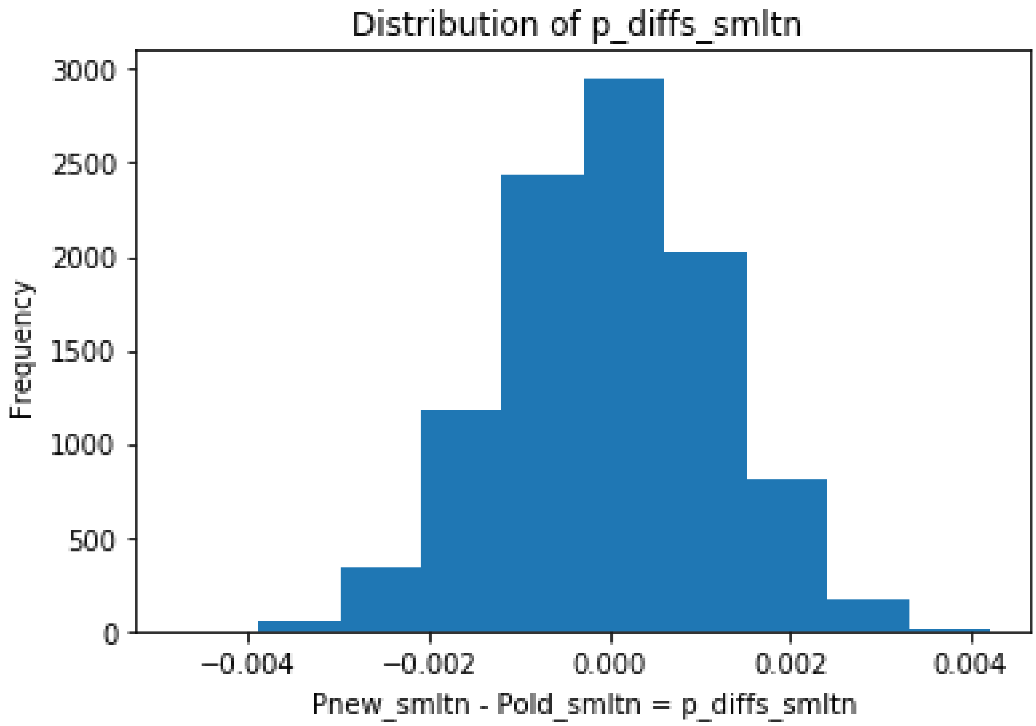
Out[32]: 0.0005348436627357345

** 2h :Simulating 10,000 p_{new} - p_{old} values and storing all 10,000 values in a numpy array called p_diffs **.

```
In [33]:
         #treatment sample
         Pnew_converted_smltd = np.random.binomial(n_new, Pnew,10000 )/n_new

         #control sample
         Pold_converted_smltd = np.random.binomial(n_old,Pold ,10000 ) / n_old

             # append the P value differece
         p_diffs_smltn = Pnew_converted_smltd  - Pold_converted_smltd
```

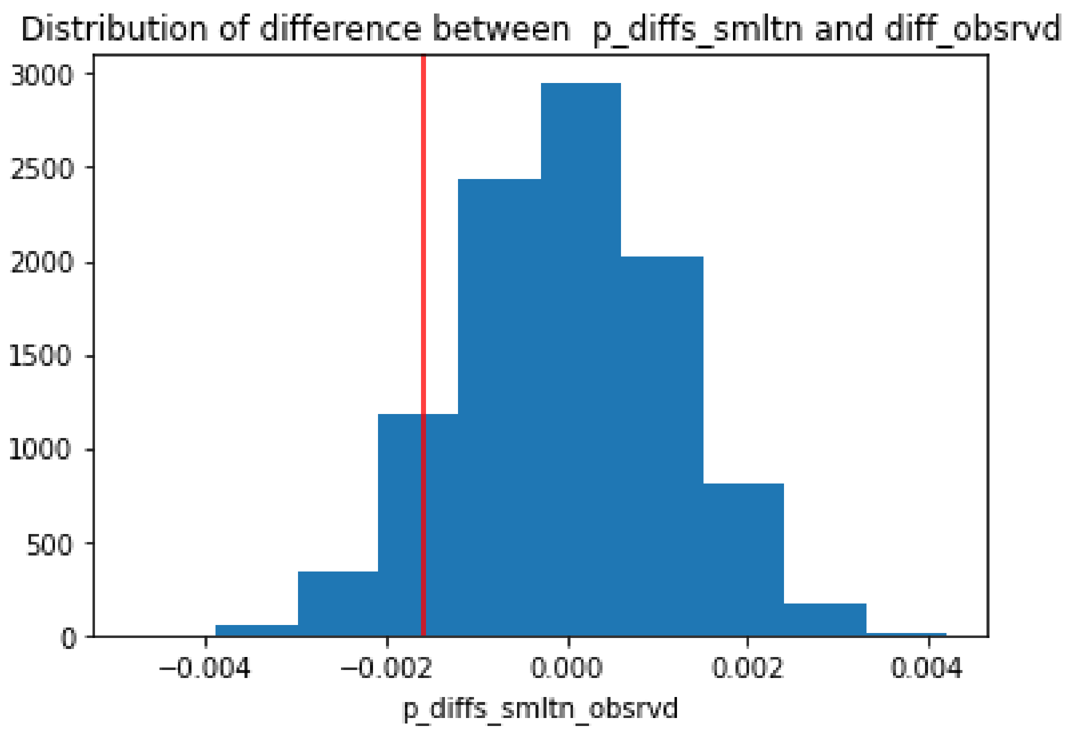
** 2i : Histogram of the p_diffs **

```
In [34]: #ploting histogram for p_diffs
         plt.hist(p_diffs_smltn)
         plt.title ('Distribution of p_diffs_smltn ');
         plt.xlabel('Pnew_smltn - Pold_smltn = p_diffs_smltn')
         plt.ylabel('Frequency ');
```



** 2j :Proportion of the p_diffs that are greater than the actual difference observed in ab_data.csv**

In [35]:
```
#Ploting p_diffs  and observed difference .
plt.hist(p_diffs_smltn)
plt.title ('Distribution of difference between  p_diffs_smltn and diff_obsrvd');
plt.xlabel('p_diffs_smltn_obsrvd')
plt.axvline(x=diffs_obsrvd ,color='r');
```



In [36]:
```
#Calculating P value .
P_value=(p_diffs_smltn > diffs_obsrvd).mean()
P_value
```

Out[36]: 0.9088

** 2k: ## **Discussion** The p value is 0.9065 which can be considered as a larger p value (since its greater than 0.05). Large p value indicates weak evidence against the null hypothesis.**

Therefore , based on the A/B simulation test performed , we CAN NOT reject the null hypothesis. This means that we are not able to conclude with enough evidence that there is significant difference between the treatment (the new page) and the control (the old page), interms of converting a page viewer.

** 2l : Utilizing Built-in functions to equate and compare with the simulation result, and see if the built-in code yield similar results **

In [37]:
```
import statsmodels.api as sm
convert_old = ctrl_cnvrtd.converted
convert_new = trmnt_cnvrtd.converted
total_new=trmnt_ttl
total_old=ctrl_ttl
```

```
C:\ProgramFiles\ANACONDA\lib\site-packages\statsmodels\compat\pandas.py:56: FutureWarning: The
  from pandas.core import datetools
```

** 2m: Using `stats.proportions_ztest` to compute `test statistic and p-value` **.

```
In [38]: # Calculating  the z score
         import statsmodels.api as sm
         a=z_score,p_value=sm.stats.proportions_ztest(
             [convert_new,convert_old], [total_new,total_old], alternative='larger')
         a
```

```
Out[38]: (-1.3109241984234394, 0.9050583127590245)
```

```
In [39]: from scipy.stats import norm
         norm.cdf(z_score)
```

```
Out[39]: 0.09494168724097551
```

```
In [40]: # critical value at 95% confidence interval
         norm.ppf(1-(0.05/2))
```

```
Out[40]: 1.959963984540054
```

```
In [41]: a=df_cleaned[df_cleaned['landing_page']== 'new_page']
         b=df1[df1['converted']==1]
         c=df_cleaned[df_cleaned['landing_page']== 'old_page']
         d=df3[df3['converted']==1]
         #new page View ,converted new page ,old page view , converted old page
         a.shape,b.shape,c.shape , d.shape
```

```
Out[41]: ((145310, 5), (17514, 5), (145274, 5), (17723, 5))
```

** 2n : ## **Discussion** Since the z-score of -1.3109241984234394 exceeds the critical value of -1.959963984540054, we CAN NOT reject the null hypothesis since the difference between the two proportions is almost near to zero. The P value from the Z-test score is also consistant with the P value obtained from simulation.**

** The treatment or the new landing page conversion rate \$(17,264 \$ / \$ 145,310)\$ IS NOT statistically different; or even we can say , its not better than the control or the old landing page conversion rate $(17,489$ / \$ 145,274)\$. There is no sufficient evidence which suggests that in long-term performance of the control and treatment page, to be different from one another.**

The built-in functions has the same result that indicating that there is no sufficient evidence to conclude that the usage of the new treatment page will lead to more conversion.

Part III - A REGRESSION APPROACH
** 1a: `Since the each row under investigation is catagorical , the most appropriat regresion method  that should  be adopted in this case is  Multi Linear or Logestic Regression` **
The choice of the regression methodologies is based on :

** i : `Outcome` **

In linear regression, the outcome (dependent variable) is continuous. It can have any one of an infinite number of possible values.

In logistic regression, the outcome (dependent variable) has only a limited number of possible values.

** ii : `The dependent variable` **

Logistic regression is used when the response variable is categorical in nature. For instance, yes/no, true/false, red/green/blue, 1st/2nd/3rd/4th, etc.

Linear regression is used when your response variable is continuous. For instance, weight, height, number of hours, etc.

** iii : `Equation` **

Linear regression gives an equation which is of the form Y = mX + C, means equation with degree 1.

However, logistic regression gives an equation which is of the form Y = eˆX/1 + eˆ-X

** iv : `Coefficient interpretation` **

In linear regression, the coefficient interpretation of independent variables are quite straightforward (i.e. holding all other variables constant, with a unit increase in this variable, the dependent variable is expected to increase/decrease by xxx).

However, in logistic regression, depends on the family (binomial, Poisson, etc.) and link (log, logit, inverse-log, etc.) you use, the interpretation is different.

** iv : `Error minimization technique` **

Linear regression uses ordinary least squares method to minimise the errors and arrive at a best possible fit, while logistic regression uses maximum likelihood method to arrive at the solution.

Linear regression is usually solved by minimizing the least squares error of the model to the data, therefore large errors are penalized quadratically.

Logistic regression is just the opposite. Using the logistic loss function causes large errors to be penalized to an asymptotically constant.

** 1b: `Creating dummy variables for landing pages` **

In [50]:
```
#create dummy variable

df_dummy=pd.get_dummies(df_cleaned,columns=['landing_page'],prefix='dummy')
df_dummy.head(1)
```

Out[50]:
```
   user_id                   timestamp      group  converted  dummy_new_page  \
2   661590  2017-01-11 16:55:06.154213  treatment          0               1

   dummy_old_page
2               0
```

In [51]:
```
# drop columns
# only 1 dummy variable required since we have 2  catagories of the variable .
columns = ['user_id','timestamp','group','dummy_old_page' ]
# Drop all other columns except the convert and dummy variable
df_dummy.drop(columns, inplace=True, axis=1)
```

** 1c: Using statsmodels to fit the regression model to check if there is a significant difference in conversion based on the page a customer receives. **

```
In [52]: # work arround for summary error kept recieving .
         from scipy import stats
         stats.chisqprob = lambda chisq, df: stats.chi2.sf(chisq, df)

         # fitting the logestic regression
         df_dummy['interecept']=1
         log_mod=sm.Logit(df_dummy['converted'], df_dummy [['interecept','dummy_new_page']])
         results =log_mod.fit()
         results.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.366118
         Iterations 6
```

```
Out[52]: <class 'statsmodels.iolib.summary.Summary'>
         """
                                    Logit Regression Results
         ==============================================================================
         Dep. Variable:              converted   No. Observations:               290584
         Model:                          Logit   Df Residuals:                   290582
         Method:                           MLE   Df Model:                            1
         Date:                Fri, 09 Mar 2018   Pseudo R-squ.:               8.077e-06
         Time:                        10:27:13   Log-Likelihood:            -1.0639e+05
         converged:                       True   LL-Null:                   -1.0639e+05
                                                 LLR p-value:                    0.1899
         ==================================================================================
                              coef    std err          z      P>|z|      [0.025      0.975]
         ----------------------------------------------------------------------------------
         interecept        -1.9888      0.008   -246.669      0.000      -2.005      -1.973
         dummy_new_page    -0.0150      0.011     -1.311      0.190      -0.037       0.007
         ==================================================================================
         """
```

1d ### Discussion

The Lofistic regression model could be summurized as follows:

Intercept Coefficient = -0.9888

new_page coefficient(slope) = -0.0150

The regression line equation is "y=-0.9888 - 0.015*x"

We can say for every user landed on the new page (in other ways who has recieved the treatment), the conversion rate decrease by 0.015.

** 1e **

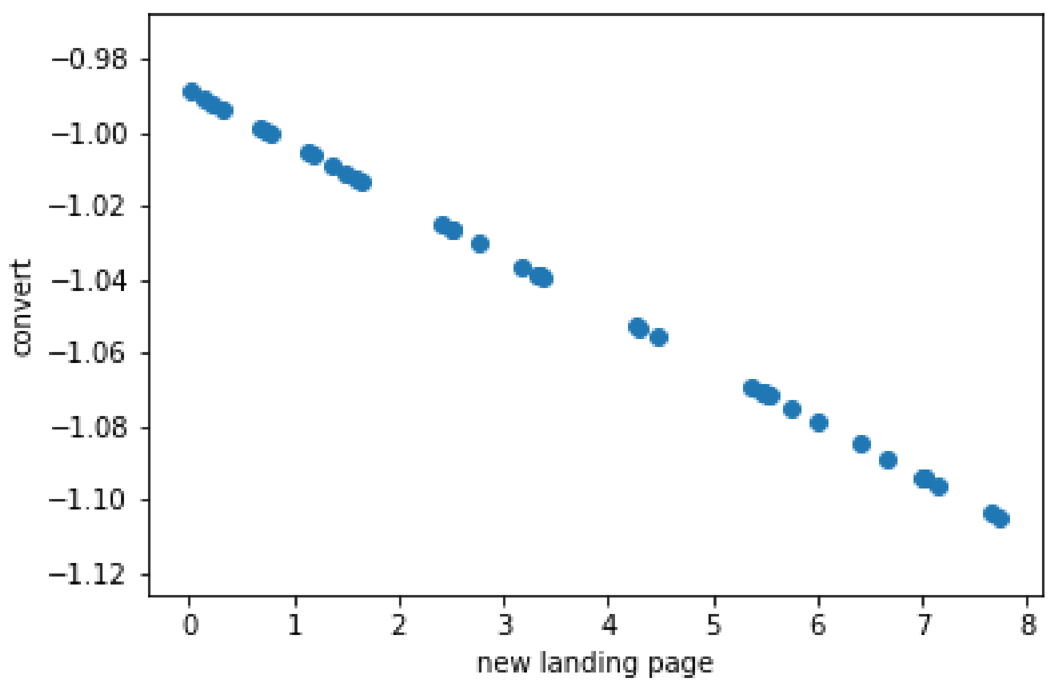
p vaues of the particular variable is useful to predict its respone to the hypothesis . In this eCommerce web page test data ,the p value for the intercept is 0.000 and that of the new page is 0.190.

The p value for the interecept is zero, which makes the null hypothesis much more statistically significant than that of its counterpart the Alternative hypothesis .

** The P value from the logistic regression is difrent from A/B test result because logistic regression is a two way test for significance while the A/B is one way test .**

`** The above  Case can be  graphically represented  as folllows :**`

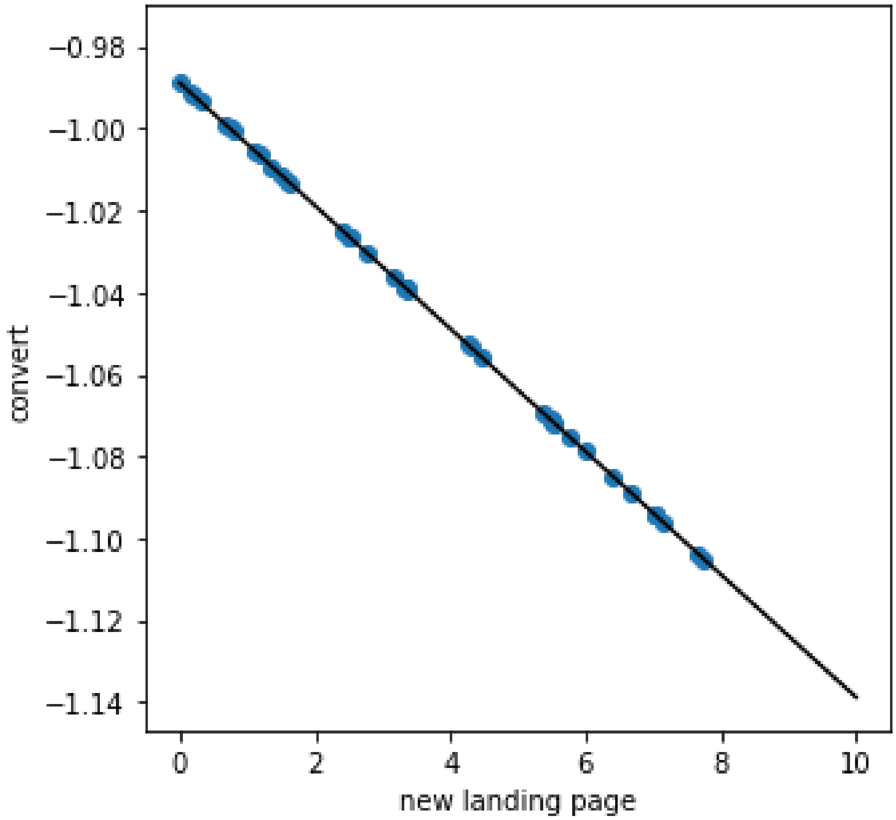
In [53]:
```
# standard devations are smoothed out
rng=np.random.RandomState(1)
x=8*rng.rand(40)
y=-.9888-0.015*x
plt.scatter(x,y)
plt.rcParams["figure.figsize"] = (5,5);
plt.xlabel('new landing page')
plt.ylabel('convert');
```



1.1.2 If regression is used for prediction

In [54]:
```
from sklearn.linear_model import LinearRegression
model=LinearRegression (fit_intercept=True )
```

```
model.fit (x[:,np.newaxis],y)
xfit=np.linspace(0,10,100,500)

yfit=model.predict (xfit[:,np.newaxis])
plt.scatter(x,y)
plt.plot(xfit,yfit, color='black' )
plt.rcParams["figure.figsize"] = (5,5);
plt.xlabel('new landing page')
plt.ylabel('convert');
```




```
In [55]: print("model slope:",model.coef_[0])
         print("Model intercept:", model.intercept_)
```

```
model slope: -0.014999999999999989
Model intercept: -0.9888
```

```
In [60]: from sklearn.model_selection import train_test_split
         from sklearn.linear_model import LogisticRegression
         from sklearn.metrics import precision_score ,recall_score,accuracy_score, confusion_ma
```

```
In [61]: df_dummy.head(1)
```

Out[61]:   converted dummy_new_page interecept
```
2          0               1           1
```

In [62]:
```
# split the  data into train and test
X = df_dummy.iloc[:,1:]
y = df_dummy['converted']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.20,random_state=
```

In [63]:
```
log_mod=LogisticRegression()
log_mod.fit(X_train , y_train)
y_preds = log_mod.predict (X_test)
print (precision_score(y_test , y_preds))
print (recall_score(y_test , y_preds))
print (accuracy_score ( y_test , y_preds))
confusion_matrix(y_test ,y_preds)
```

```
0.0
0.0
0.8791919748094361


C:\ProgramFiles\ANACONDA\lib\site-packages\sklearn\metrics\classification.py:1135: UndefinedMe
  'precision', 'predicted', average, warn_for)
```

Out[63]:
```
array([[51096,     0],
       [ 7021,     0]], dtype=int64)
```

The confusion matrix telling us that we have 51096+0 correct predictions and 7021+0 incorrect predictions.

** 2f: Other factors **

**   Its important to investigate other factors in the regression model as well to throughly understand and examine the significance.**

** 2g :Investigating if the location(country) that the page has been viewed , has an impact on conversion. **

In [98]:
```
#Loading the country  data csv
countries_df = pd.read_csv('countries.csv')
df_new = countries_df.set_index('user_id').join(df2.set_index('user_id'), how='inner'

# remove index
df_new = df_new.rename_axis(None)
# check the available catagories
df_new.head()
```

Out[98]:

	country	timestamp	group	landing_page	converted
928468	US	2017-01-23 14:44:16.387854	treatment	new_page	0
822059	UK	2017-01-16 14:04:14.719771	treatment	new_page	1

```
710616      UK  2017-01-16 13:14:44.000513  treatment     new_page          0
909908      UK  2017-01-06 20:44:26.334764  treatment     new_page          0
811617      US  2017-01-02 18:42:11.851370  treatment     new_page          1
```

In [99]:
```
### Create the necessary dummy variables
df_new1=pd.get_dummies(df_new,columns=['landing_page'],prefix='dummy')
df_new_dummy=pd.get_dummies(df_new1,columns=['country'], drop_first=True)

# remove columns

#df_new_dummy = df_new_dummy.rename_axis(None)
df_new_dummy.head(1)
```

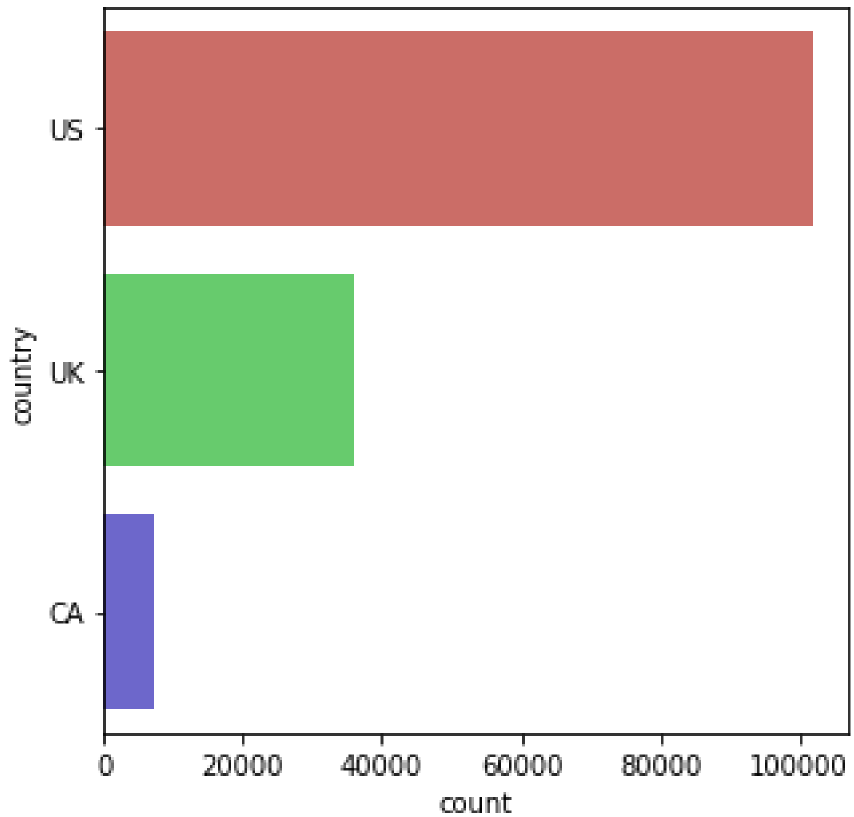
Out[99]:
```
                         timestamp      group  converted  dummy_new_page  \
928468  2017-01-23 14:44:16.387854  treatment          0               1

        country_UK  country_US
928468           0           1
```

In [100]:
```
cols = [0,1]
df_new_dummy.drop(df_new_dummy.columns[cols],axis=1,inplace=True)
df_new_dummy.head ()
```

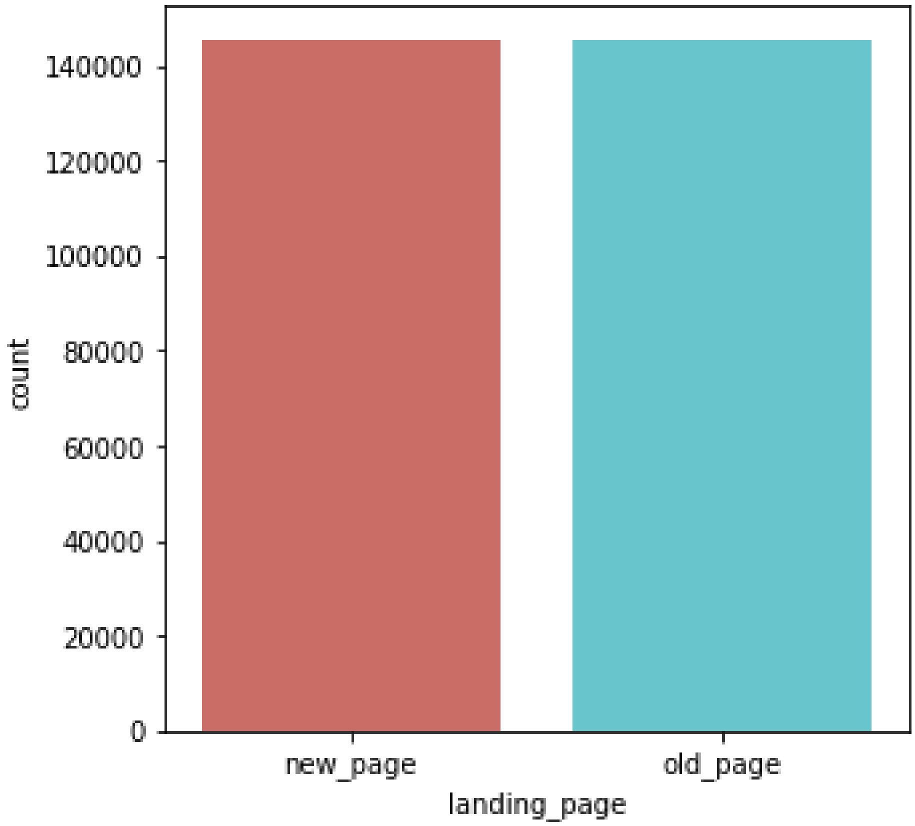
Out[100]:

	converted	dummy_new_page	country_UK	country_US
928468	0	1	0	1
822059	1	1	1	0
710616	0	1	1	0
909908	0	1	1	0
811617	1	1	0	1

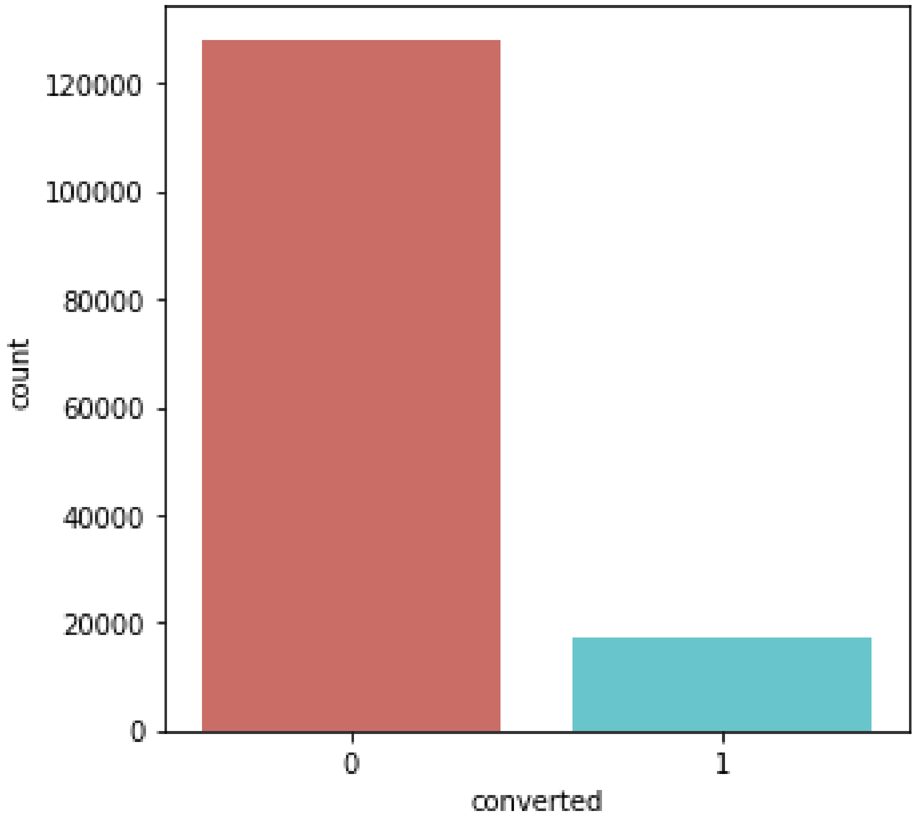
In [77]:
```
#Barplot for the independent  variable (country )
import  seaborn as sns
sns.countplot(y='country',data=df_new, palette='hls')
plt.show()
```




```
In [78]: #Barplot for the independent  variable (Landing Page )
         sns.countplot(x='landing_page',data=df_cleaned, palette='hls')
         plt.show()
```



In [79]:
```
#(binary: 1, means converted, 0 means not converted)

#Barplot for the dependent variable
sns.countplot(x='converted',data=df_new_dummy, palette='hls')
plt.show()
```




```
In [104]: #Fitting the new model
          df_new_dummy['interecept']=1
          log_mod=sm.Logit(df_new_dummy['converted'], df_new_dummy [['interecept','dummy_new_pa
          results =log_mod.fit()
          results.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.364535
         Iterations 6
```

```
Out[104]: <class 'statsmodels.iolib.summary.Summary'>
          """
                                     Logit Regression Results
          ==============================================================================
          Dep. Variable:              converted   No. Observations:               145310
          Model:                          Logit   Df Residuals:                   145308
          Method:                           MLE   Df Model:                            1
          Date:                Fri, 09 Mar 2018   Pseudo R-squ.:               2.409e-05
          Time:                        10:51:56   Log-Likelihood:                -52971.
          converged:                       True   LL-Null:                       -52972.
```

```
                                        LLR p-value:                    0.1101
=================================================================================
                    coef    std err          z      P>|z|      [0.025      0.975]
---------------------------------------------------------------------------------
interecept       -1.0056   7.04e+05  -1.43e-06      1.000   -1.38e+06    1.38e+06
dummy_new_page   -1.0056   7.04e+05  -1.43e-06      1.000   -1.38e+06    1.38e+06
country_UK        0.0299      0.019      1.601      0.109      -0.007       0.066
=================================================================================
"""
```

In [ ]:

** 2h: **

1.2 Discussion

** The following interpretations can be infered from the above regression table :-**

** If the country that the old page is viewed is from Canada(CA), there is 1005.6 % less chance the viewer get converted ,Ceteris paribus . **

** If viwed from UK ,* 2.99%% * more viewers can be expected to convert than that of when viewed from CA ,Ceteris paribus.**

**

** Furthermore , if a viewer recieved the treatment (or viwed the new page),there is *1005.6 %* marginal converts than customers who viewed the control group , Ceteris paribus . **

p value for new page is 1 which essentially rejects the null hypothesis , if all other things held constant .

When we look at the p value for the country variables , UK has lower p value (0.109 as compared to CA (1) , indicating statistically less significant influence on the reponse than page viwed in CA.

Key Assumptions

- **There is a linear relationship between the converted and the independent variables landing page and country.**
- **The residuals are normally distributed.**
- **The independent variables are not highly correlated with each other. This assumption is tested using Variance Inflation Factor (VIF) values.**
- **Homoscedasticity–that is the variance of error terms are similar across the values of the independent variables.**
- **No Multicollinearity**

Logestic Regression Model

```
In [64]: df_new_dummy.head(1)
```

Out[64]:

	converted	dummy_new_page	country_UK	country_US	interecept
928468	0	1	0	1	1

In [65]:
```
from sklearn import preprocessing
import matplotlib.pyplot as plt
plt.rc("font", size=14)
from sklearn.linear_model import LogisticRegression
from sklearn.cross_validation import train_test_split
import seaborn as sns
sns.set(style="white")
sns.set(style="whitegrid", color_codes=True)
```

```
C:\ProgramFiles\ANACONDA\lib\site-packages\sklearn\cross_validation.py:41: DeprecationWarning:
  "This module will be removed in 0.20.", DeprecationWarning)
```

In [66]:
```
df_new1=pd.get_dummies(df_new,columns=['landing_page'],prefix='dummy')
df_new_dummy2=pd.get_dummies(df_new1,columns=['country'], drop_first=True)

# remove columns
columns = ['timestamp','group']
df_new_dummy=pd.get_dummies(df_new1,columns=['country'], drop_first=True)
df_new_dummy2.drop(columns, inplace=True, axis=1)
```

In [67]:
```
# split the  data into train and test
X = df_new_dummy2.iloc[:,1:]
y = df_new_dummy2.iloc[:,0]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.20,random_state=(
```

In [68]:
```
#Fit logestic Regression to training set
classifier = LogisticRegression(random_state=0)
classifier.fit(X_train, y_train)
```

Out[68]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=0, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

In [69]:
```
#Predicting the test set results and creating confusion matrix
y_pred = classifier.predict(X_test)
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test, y_pred)
print(confusion_matrix)
```

```
[[25592     0]
 [ 3470     0]]
```

** The result is telling us that we have 25592+0 correct predictions and 3470+0 incorrect predictions. **

```
In [70]: #Accuracy of logistic regression classifier on test set
         print('Accuracy : {:.2f}'.format(classifier.score(X_test, y_test)))
```

```
Accuracy : 0.88
```

```
In [71]: #Compute precision, recall, F-measure and support
         from sklearn.metrics import classification_report
         print(classification_report(y_test, y_pred))
```

```
             precision    recall  f1-score   support

          0       0.88      1.00      0.94     25592
          1       0.00      0.00      0.00      3470

avg / total       0.78      0.88      0.82     29062


C:\ProgramFiles\ANACONDA\lib\site-packages\sklearn\metrics\classification.py:1135: UndefinedMe
  'precision', 'predicted', average, warn_for)
```

Conclusion

** In this project , three type of approches , namely the Probability , A/B Test and Regression analysis, are adopted to investigate the data obtained from eCommerce website A/B test . Simulation ,Z-test and Logistic Regression tests were implemented to examin and appriciate the statitistical significance .**

** The results obtained from Simulation and Z-test are identical , that there is no sufficient evidence to reject the null hypothesis which states the conversion rate for new page is less than or equal to the conversion rate of the old page .In contrast ,the result from Logistic Regression , however , rejects the null hypothesis and confirms that the new web page has statistically significant influence on conversion .The difference could be attributed to the test type .That is regression considered as two tailed test while the simulation and Z-test tests are one tailed test (where p_new > p_old) .**

** Practically speaking ,if all other factors assumed constant and the only changes adopted were on the landing page ; based on the observations , the statistical tests and prediction conducted ,I would suggest to expire the experimentation , assimilate the sunken cost and keep utilizing the exisiing landing page , until an alternative landing page is developed .**