Dempsey Wade

Module 8 - Experimental Design, Causal Research, and Targeting Analysis

Author: Favio Vázquez and Jessica Cervi

In this assignment we will be focusing on A/B testing.

You will be using data from a marketing campaign conducted in a restaurant that wants to add a new item to its menu. We will test the effectiveness of three possible marketing campaigns to promote the new item.

The new item was introduced in several randomly selected locations. A different promotion was used in each location, the goal is to determine which promotion had the greatest effect on sales. The weekly sales of the new item were recorded for the first four weeks.

Index:

Import the necessary libraries and read the data

In [1]:
```
# Import libraries
import pandas as pd
import matplotlib.pyplot as plt
from scipy import stats
```

Reading the data # The dataset is stored as usual in a CSV file so we will use pandas "read_csv" function to load it. # Note: don't rename this dataframe df = pd.read_csv("data/marketing.csv")

In [2]:
```
# Reading the data
# The dataset is stored as usual in a CSV file so we will use pandas "read_csv" function
# Note: don't rename this dataframe
df = pd.read_csv("/Users/dempseywade/Desktop/gitRepo/DartmouthCodingAssignments/data/Mod
```

In [3]:
```
# Visualize the first 5 rows of the dataframe
df.head()
```

Out[3]:

	MarketID	MarketSize	LocationID	AgeOfStore	Promotion	week	SalesInThousands

0	1	Medium	1	4	3	1	33.73
1	1	Medium	1	4	3	2	35.67
2	1	Medium	1	4	3	3	29.03
3	1	Medium	1	4	3	4	39.25
4	1	Medium	2	5	2	1	27.81

Basic Exploratory Data Analysis

In [4]:
```
#Retrieving information aboout the dataframe
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 548 entries, 0 to 547
Data columns (total 7 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   MarketID          548 non-null    int64 
 1   MarketSize        548 non-null    object
 2   LocationID        548 non-null    int64 
 3   AgeOfStore        548 non-null    int64 
 4   Promotion         548 non-null    int64 
 5   week              548 non-null    int64 
 6   SalesInThousands  548 non-null    float64
dtypes: float64(1), int64(5), object(1)
memory usage: 30.1+ KB
```

In [5]:
```
# Analyzes both numeric and object series
df.describe()
```

Out[5]:

	MarketID	LocationID	AgeOfStore	Promotion	week	SalesInThousands
count	548.000000	548.000000	548.000000	548.000000	548.000000	548.000000
mean	5.715328	479.656934	8.503650	2.029197	2.500000	53.466204
std	2.877001	287.973679	6.638345	0.810729	1.119055	16.755216
min	1.000000	1.000000	1.000000	1.000000	1.000000	17.340000
25%	3.000000	216.000000	4.000000	1.000000	1.750000	42.545000
50%	6.000000	504.000000	7.000000	2.000000	2.500000	50.200000
75%	8.000000	708.000000	12.000000	3.000000	3.250000	60.477500
max	10.000000	920.000000	28.000000	3.000000	4.000000	99.650000

In [6]:
```
#Retrieving the shape of the dataframe
# You should see 548 rows and 7 columns
df.shape
```

Out[6]:
```
(548, 7)
```

Return to top

Question 1

Get the sum of the total sales (`SalesInThousands`) across different promotions. Save your results in a dataframe named ans1. The resulting dataframe should contain `Promotion` as index, along with the column `SalesInThousands` .

Hint: Use use the `pandas` attribute `groupby` .

In [7]:
```
### GRADED

### YOUR SOLUTION HERE
ans1 = df.groupby(by=['Promotion']).sum()
ans1 = ans1.drop(['MarketID', 'LocationID', 'AgeOfStore','week'], axis = 1)

###
### YOUR CODE HERE
###
### Answer check
print("Total sales across promotions:")
print(ans1)
```

```
Total sales across promotions:
           SalesInThousands
Promotion
1                   9993.03
2                   8897.93
3                  10408.52
```

In [8]:
```
###
### AUTOGRADER TEST - DO NOT REMOVE
###
```

Return to top

Question 2

Compute the counts of different `MarketID` s for each `Promotion` group and `MarketSize` pair. Save your results in a dataframe called ans2. The final dataframe should contain `Promotion` and `MarketSize` as indexes, and the column `MarketID` .

In [9]:
```
### GRADED

### YOUR SOLUTION HERE
ans2 = df.groupby(by=['Promotion', 'MarketSize']).count()
ans2 = ans2.drop(['LocationID','AgeOfStore','week', 'SalesInThousands'], axis = 1)

###
### YOUR CODE HERE
###
### Answer check
print("Number of market counts for each (Promotion, MarketSize):")
print(ans2)
```

```
Number of market counts for each (Promotion, MarketSize):
                     MarketID
Promotion MarketSize
1         Large            56
          Medium           96
          Small            20
2         Large            64
          Medium          108
          Small            16
```

```
3          Large              48
           Medium            116
           Small              24
```

In [10]:
```
###
### AUTOGRADER TEST - DO NOT REMOVE
###
```

Return to top

Question 3

Find the counts of `MarketID` across all different `Promotion` groups and `AgeOfStore`. Save your results in a dataframe called ans3. The resulting dataframe should have `AgeOfStore` as index, a MultiIndex column given by `Promotion` and the counts of `MarketID`.

Hint: To create the MultiIndex column, use `unstack()` on `Promotion`.

In [11]:
```
### GRADED
# Expect some NaN values in the dataset

### YOUR SOLUTION HERE
ans3 = df.groupby(by=['Promotion', 'AgeOfStore']).count()
ans3 = ans3.drop(['LocationID', 'MarketSize','week', 'SalesInThousands'], axis = 1)

#In the end you have to unstack the promotion
ans3 = ans3.unstack(level = 0)

###
### YOUR CODE HERE
###
### Answer check
print("Number of markets given promotion and the age of the store:")
print(ans3)
```

```
Number of markets given promotion and the age of the store:
            MarketID
Promotion          1     2     3
AgeOfStore
1               24.0  36.0  20.0
2                8.0   8.0   4.0
3               16.0  12.0   4.0
4               16.0  12.0  16.0
5                8.0  12.0  24.0
6               20.0   4.0  12.0
7                4.0  24.0  12.0
8               12.0   8.0  20.0
9                8.0  12.0   8.0
10               NaN  16.0   8.0
11               4.0   NaN  12.0
12              12.0   4.0   8.0
13              12.0   8.0   NaN
14               NaN   8.0   4.0
15               4.0   4.0   NaN
17               NaN   NaN   4.0
18               8.0   NaN   NaN
19               4.0   8.0   8.0
20               NaN   NaN   4.0
22               4.0   NaN   8.0
23               NaN   4.0   4.0
24               4.0   NaN   8.0
```

```
25              NaN   4.0   NaN
27              4.0   NaN   NaN
28              NaN   4.0   NaN
```

In [12]:
```
###
### AUTOGRADER TEST - DO NOT REMOVE
###
```

Understanding the promotions with statistics

Return to top

Question 4

Create a two-sided T-test with the null hypothesis that the sales for **Promotion 1** and **Promotion 2** have identical expected average values. Save the obtained t-statistic and the p-value into two variables called `t_1` and `p_1`, respectively.

For this question, use the `scipy` module `stats.ttest_ind`. Set the value of the argument `equal_var` to `False` to perform a Welch's t-test that **does not assume equal population variance.**

You can find more information about the usage of the module `stats.ttest_ind` here: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_ind.html

Hint: Use the `SalesInThousands` variable for the values.

In [13]:
```
### GRADED

### YOUR SOLUTION HERE

pro1 = df.loc[df['Promotion'] == 1]
pro2 = df.loc[df['Promotion'] == 2]

t_1, p_1 = stats.ttest_ind(pro1['SalesInThousands'], pro2['SalesInThousands'], equal_var

###
### YOUR CODE HERE
###
### Answer check
print("Test: sales of Promotions 1 and 2.\n  t-statistic: {}\n  p-value: {}".format(t_1,
```

```
Test: sales of Promotions 1 and 2.
  t-statistic: 6.42752867090748
  p-value: 4.2903687179871785e-10
```

In [14]:
```
###
### AUTOGRADER TEST - DO NOT REMOVE
###
```

Return to top

Question 5

Create a two-sided T-test with the null hypothesis that the sales for **Promotion 1** and **Promotion 3** have identical expected average values. Save the obtained t-statistic and the p-value into two variables called `t_2` and `p_2`, respectively. Remember to set `equal_var` to `False`.

In [15]:
```
### GRADED


pro1 = df.loc[df['Promotion'] == 1]
pro3 = df.loc[df['Promotion'] == 3]

### YOUR SOLUTION HERE
t_2, p_2 = stats.ttest_ind(pro1['SalesInThousands'], pro3['SalesInThousands'], equal_var


###
### YOUR CODE HERE
###
### Answer check
print("Test: sales of Promotions 1 and 3.\n  t-statistic: {}\n  p-value: {}".format(t_2,
```

```
Test: sales of Promotions 1 and 3.
  t-statistic: 1.5560224307758634
  p-value: 0.12059147742229478
```

In [16]:
```
###
### AUTOGRADER TEST - DO NOT REMOVE
###
```

Return to top

Question 6

Create a two-sided T-test with the null hypothesis that the sales for **Promotion 2** and **Promotion 3** have identical expected average values. Save the obtained t-statistic and the p-value into two variables called `t_3` and `p_3`, respectively. Remember to set `equal_var` to `False`.

In [17]:
```
### GRADED

pro2 = df.loc[df['Promotion'] == 2]
pro3 = df.loc[df['Promotion'] == 3]

### YOUR SOLUTION HERE
t_3, p_3 = stats.ttest_ind(pro2['SalesInThousands'], pro3['SalesInThousands'], equal_var

###
### YOUR CODE HERE
###
### Answer check
print("Test: sales of Promotions 2 and 3.\n  t-statistic: {}\n  p-value: {}".format(t_3,
```

```
Test: sales of Promotions 2 and 3.
  t-statistic: -4.88139271089348
  p-value: 1.5692733176039892e-06
```

In [18]:
```
###
### AUTOGRADER TEST - DO NOT REMOVE
###
```

Return to top

Question 7

Which promotion had the biggest sales in average and how much was it? Save your result in a tuple named ans4. The resulting tuple should have the following structure:

```
("Promotion #", Average_sale)
```

Round the value of Average_sale to two decimals digits.

Example:

```
ans4 = ("Promotion 0", 12.15)
```

Hint: The first element of the tuple should be a string and the second one a float.

In [19]:
```
### GRADED

### YOUR SOLUTION HERE
ans4 = ("Promotion 1", 58.10)
print(df.groupby(['Promotion']).mean()[["SalesInThousands"]].unstack("Promotion"))

###
### YOUR CODE HERE
###
```

```
                  Promotion
SalesInThousands  1            58.099012
                  2            47.329415
                  3            55.364468
dtype: float64
```

In [20]:
```
###
### AUTOGRADER TEST - DO NOT REMOVE
###
```

Return to top

Question 8

Based on your previous results, can we say that that the marketing performance of promotion group 1 is not statistically different from the marketing performance of promotion group 3, even though the average number of sales of promotion group 1 is higher than the average number of sales of promotion group 3?

Assign the booloean value `True` or `False` to a variable called ans5.

In [21]:
```
### GRADED

### YOUR SOLUTION HERE
ans5 = True

###
### YOUR CODE HERE
###
```

In [22]:
```
###
### AUTOGRADER TEST - DO NOT REMOVE
###
```

Return to top

Question 9

Based on your previous results, can we say that promotion 2 significantly increased sales?

Assign the booloean value `True` or `False` to a variable called ans6.

In [23]:
```
### GRADED

### YOUR SOLUTION HERE
ans6 = False

###
### YOUR CODE HERE
###
```

In [24]:
```
###
### AUTOGRADER TEST - DO NOT REMOVE
###
```