# template-

November 17, 2023

# 1 Your Title Here

**Name(s)**: (your name(s) here)

**Website Link**: https://lr580.github.io/power_outages_stats/

## 1.1 Code

[ ]:
```
import pandas as pd
import numpy as np
import os
import plotly.express as px
pd.options.plotting.backend = 'plotly'
```

### 1.1.1 Cleaning and EDA

**Data Cleaning**

**Read the Data** Since the data is not large(as stated above, only over a thousand rows), we opened it beforehand via Microsoft Excel and found that the columns names are located at 6th rows, while the data starts at 8th row. And it's noticed that the first column is empty, which means that the real column starts at the second column. Therefore, we need to open the dataset starting from the 8th row and the 2nd column, while reading the 6th column as column names.

To realize this, we first use `header=5` to skip the first 5 rows and uses the 6th row as the names for columns. Then, we drop the first row(actually the 7th row in the raw file) and the first column.

[ ]:
```
# skip first 5 rows
data = pd.read_excel("outage.xlsx", header=5)
# drop the 7th row
data = data.drop(0)
# drop the 1st column
data = data.drop(data.columns[0], axis=1)
# show the data
print(data)
```

```
         OBS    YEAR  MONTH    U.S._STATE POSTAL.CODE NERC.REGION  \
1        1.0  2011.0    7.0     Minnesota          MN         MRO
2        2.0  2014.0    5.0     Minnesota          MN         MRO
3        3.0  2010.0   10.0     Minnesota          MN         MRO
```

```
4        4.0  2012.0    6.0     Minnesota          MN         MRO
5        5.0  2015.0    7.0     Minnesota          MN         MRO
…       …      …     …            …           …          …
1530  1530.0  2011.0   12.0  North Dakota          ND         MRO
1531  1531.0  2006.0    NaN  North Dakota          ND         MRO
1532  1532.0  2009.0    8.0  South Dakota          SD         RFC
1533  1533.0  2009.0    8.0  South Dakota          SD         MRO
1534  1534.0  2000.0    NaN        Alaska          AK        ASCC

          CLIMATE.REGION ANOMALY.LEVEL CLIMATE.CATEGORY    OUTAGE.START.DATE  \
1     East North Central          -0.3           normal  2011-07-01 00:00:00
2     East North Central          -0.1           normal  2014-05-11 00:00:00
3     East North Central          -1.5             cold  2010-10-26 00:00:00
4     East North Central          -0.1           normal  2012-06-19 00:00:00
5     East North Central           1.2             warm  2015-07-18 00:00:00
…                    …             …                …                    …
1530  West North Central          -0.9             cold  2011-12-06 00:00:00
1531  West North Central           NaN              NaN                  NaN
1532  West North Central           0.5             warm  2009-08-29 00:00:00
1533  West North Central           0.5             warm  2009-08-29 00:00:00
1534                 NaN           NaN              NaN                  NaN

      … POPPCT_URBAN POPPCT_UC POPDEN_URBAN POPDEN_UC POPDEN_RURAL  \
1     …        73.27     15.28         2279    1700.5         18.2
2     …        73.27     15.28         2279    1700.5         18.2
3     …        73.27     15.28         2279    1700.5         18.2
4     …        73.27     15.28         2279    1700.5         18.2
5     …        73.27     15.28         2279    1700.5         18.2
…   …            …         …            …         …            …
1530  …         59.9      19.9       2192.2    1868.2          3.9
1531  …         59.9      19.9       2192.2    1868.2          3.9
1532  …        56.65     26.73       2038.3    1905.4          4.7
1533  …        56.65     26.73       2038.3    1905.4          4.7
1534  …        66.02     21.56       1802.6      1276          0.4

     AREAPCT_URBAN AREAPCT_UC   PCT_LAND  PCT_WATER_TOT PCT_WATER_INLAND
1             2.14        0.6  91.592666       8.407334         5.478743
2             2.14        0.6  91.592666       8.407334         5.478743
3             2.14        0.6  91.592666       8.407334         5.478743
4             2.14        0.6  91.592666       8.407334         5.478743
5             2.14        0.6  91.592666       8.407334         5.478743
…                …          …          …              …                …
1530          0.27        0.1  97.599649       2.401765         2.401765
1531          0.27        0.1  97.599649       2.401765         2.401765
1532           0.3       0.15  98.307744       1.692256         1.692256
1533           0.3       0.15  98.307744       1.692256         1.692256
1534          0.05       0.02  85.761154      14.238846         2.901182
```

[1534 rows x 56 columns]

**Combine Date Column and Time Column** It's not convenience if a specific datetime is described by two columns, so first we combine the `OUTAGE.START.DATE` column and the `OUTAGE.START.TIME` column into a single `OUTAGE.START` column with the `pd.Timestamp` type.

Before the convertion, we first check the two columns to see its types and raw data.

[ ]:
```python
print(data['OUTAGE.START.DATE'].dtype)
print(data['OUTAGE.START.TIME'].dtype)
print(data.loc[1,'OUTAGE.START.DATE'])
print(data.loc[1,'OUTAGE.START.TIME'])
print(data.loc[240,'OUTAGE.START.DATE'])
print(data.loc[240,'OUTAGE.START.TIME'])
```

```
object
object
2011-07-01 00:00:00
17:00:00
nan
nan
```

It's noticed that some rows(like 240th row shown above) have NaN, so we should consider such special case when converting. The steps is shown below.

[ ]:
```python
# clone the raw data
df = data.copy()

# data type convertion
df['OUTAGE.START.DATE'] = pd.to_datetime(df['OUTAGE.START.DATE'])
df['OUTAGE.START.TIME'] = pd.to_timedelta(df['OUTAGE.START.TIME'].astype(str))

# combine two columns into one column
df['OUTAGE.START'] = df['OUTAGE.START.DATE'] + df['OUTAGE.START.TIME']
```

We check the result of the convertion.

[ ]:
```python
print(df['OUTAGE.START'].dtype)
print(df.loc[1,'OUTAGE.START'],df.loc[1,'OUTAGE.START.TIME'],df.loc[1,'OUTAGE.
  ↪START.DATE'])
print(df.loc[240,'OUTAGE.START'],df.loc[240,'OUTAGE.START.TIME'],df.
  ↪loc[240,'OUTAGE.START.DATE'])
```

```
datetime64[ns]
2011-07-01 17:00:00 0 days 17:00:00 2011-07-01 00:00:00
NaT NaT NaT
```

The cleaned dataframe's datatype for each useful column is listed below.

[ ]:
```python
selected_columns = ['OUTAGE.START', 'OUTAGE.DURATION', 'U.S._STATE', 'CLIMATE.
↪CATEGORY', 'OUTAGE.RESTORATION.DATE', 'CAUSE.CATEGORY.DETAIL']
print(df[selected_columns].dtypes)
df = df[selected_columns] # drop other useless columns
```

```
OUTAGE.START               datetime64[ns]
OUTAGE.DURATION                    object
U.S._STATE                         object
CLIMATE.CATEGORY                   object
OUTAGE.RESTORATION.DATE            object
CAUSE.CATEGORY.DETAIL              object
dtype: object
```

The cleaned dataframe is shown below(with only representative rows selected for display).

[ ]:
```python
selected_rows = df.loc[[5,4,3,54,833]]
print(selected_rows)
with open('table1.txt', 'w') as f:
    f.write(selected_rows.to_markdown(index=False))
```

```
           OUTAGE.START OUTAGE.DURATION U.S._STATE CLIMATE.CATEGORY  \
5   2015-07-18 02:00:00            1740  Minnesota             warm
4   2012-06-19 04:30:00            2550  Minnesota           normal
3   2010-10-26 20:00:00            3000  Minnesota             cold
54  2014-01-24 00:00:00          108653  Wisconsin             cold
833 2013-08-12 11:55:00               4     Oregon           normal

    OUTAGE.RESTORATION.DATE CAUSE.CATEGORY.DETAIL  OUTAGE.RESTORATION.MISSING  \
5       2015-07-19 00:00:00                   NaN                       False
4       2012-06-20 00:00:00          thunderstorm                       False
3       2010-10-28 00:00:00            heavy wind                       False
54      2014-04-09 00:00:00                  Coal                       False
833     2013-08-12 00:00:00   suspicious activity                       False

     OUTAGE.DURATION.MISSING  OUTAGE.START.MISSING  \
5                      False                 False
4                      False                 False
3                      False                 False
54                     False                 False
833                    False                 False

     CAUSE.CATEGORY.DETAIL.MISSING RANDOM  NO_RANDOM
5                             True      a      False
4                            False      b      False
3                            False      b       True
54                           False      b       True
833                          False      b      False
```

**Exploratory Data Analysis**

**Univariate Analysis** In the univariate analysis, we would analyze the distribution of the outage duration and U.S. states.

**Distribution of Outage Duration** First, we write codes to analyse our first chosen column: outage duration.

```
[ ]: fig = px.histogram(df, x='OUTAGE.DURATION', title = 'Distribution of Outage␣
     ↪Duration')
     fig.update_layout(xaxis_title='Outage Duration (Minutes)')
     fig.show()
```

This shows that exception from very few data, most of the duration is short. About half outages are less than 1,000 minutes, and most are less than 5,00 minutes, while only few outages lasting longer than that. Also, it means that the variance may be large since the minimal and maximal values differ a lot.

For convenience, we write a helper function to export the `plotly` figure into HTML file.

```
[ ]: def export_plotly_fig(fig, filename):
         fig.write_html(filename, include_plotlyjs='cdn')
```

We export the `fig` using the above function to local disk, which will be useful later in next part.

```
[ ]: export_plotly_fig(fig, 'univariate1.html')
```

**Distribution of Outage U.S. States** Then, we use the same way to analyze the distribution of U.S. states of the outages.

```
[ ]: # sorted by counts
     count_df = df['U.S._STATE'].value_counts()

     fig = px.bar(count_df, y='U.S._STATE', title = 'Distribution of Outage\'s U.S.␣
     ↪State')
     fig.update_layout(xaxis_title='Outage U.S. State', yaxis_title='counts')
     fig.show()
     export_plotly_fig(fig, 'univariate2.html')
```

This shows that some states frequently occur power outages(more than 50 times in total), and most others are less frequent, even several states only have several power outages. It's seems that the distribution is similar to gaussian distribution.

**Bivariate Analysis** Then, we do bivariate analysis between the outage duration and the state where the outage occurs.

```
[ ]: fig = px.scatter(df, x = 'U.S._STATE', y = 'OUTAGE.DURATION', title =␣
     ↪'Distribution of Outage Duration between Different States')
     fig.update_layout(xaxis_title='U.S. State', yaxis_title='Outage Duration')
     fig.show()
     export_plotly_fig(fig, 'bivariate1.html')
```

The scatter plot shows that they may exist very strong correlation between the outage duration and the state. We could say that there may have positive relationship between the two variables.

To further explore it, we calculate the average outage duration minutes of different states and plot a line graph.

```
[ ]: # calculate the average duration for each states
     avg_df = df.groupby('U.S._STATE')['OUTAGE.DURATION'].mean()
     # sorted in ascending order
     avg_df = avg_df.sort_values()
     fig = px.line(avg_df, title = 'Average Outage Duration of Different States')
     fig.update_layout(xaxis_title='U.S. State')
     fig.show()
     export_plotly_fig(fig, 'bivariate2.html')
```

It seems that the average outage duration is different between different states. To check it further, we calculate the median, and the quartile below.

```
[ ]: df2 = df.copy()
     df2.dropna(subset=['OUTAGE.DURATION'])
     df2['OUTAGE.DURATION'] = df2['OUTAGE.DURATION'].astype(float)
     stat_df = df2.groupby('U.S._STATE')['OUTAGE.DURATION'].agg(['mean', 'median'])
     stat_df = stat_df.sort_values(by='mean')
     q1 = df2.groupby('U.S._STATE')['OUTAGE.DURATION'].quantile(0.25)
     q3 = df2.groupby('U.S._STATE')['OUTAGE.DURATION'].quantile(0.75)
     stat_df = pd.merge(stat_df, q1, on='U.S._STATE')
     stat_df = pd.merge(stat_df, q3, on='U.S._STATE')
     stat_df.columns = ['mean', 'median', 'Q1', 'Q3']
     fig = px.line(stat_df, title = 'Statistics Outage Duration of Different States')
     fig.update_layout(xaxis_title='U.S. State')
     fig.show()
     export_plotly_fig(fig, 'bivariate3.html')
```

It seems that the durtion time differs a lot between different states. So we can roughly conclude that it may have strong correlation.

**Interesting Aggregates** In the above mentioned bivariate analysis, we get a grouped table `stat_df`. Here, we show it again in table form. Below shows some rows of a grouped table aggreated by states and calculated the mean, median, and quartile of the outage duration.

```
[ ]: print(stat_df.loc[['Mississippi', 'Hawaii', 'Massachusetts', 'Tennessee', 'New␣
      ↪York']])
     with open('table2.txt','w') as f:
         f.write(stat_df.loc[['Mississippi', 'Hawaii', 'Massachusetts', 'Tennessee',␣
      ↪'New York']].to_markdown())
```

```
                    mean  median     Q1       Q3
U.S._STATE
Mississippi    84.000000    17.5    4.0    97.50
Hawaii        845.400000   543.0  237.0  1367.00
```

```
Massachusetts   944.166667   211.0   19.0  1443.75
Tennessee      1041.967742   310.0   39.0  1230.00
New York       6034.957143  2880.0  268.5  8156.25
```

Furthermore, we want to explore the relationship between different climate catogory(i.e. `CLIMATE.CATEGORY` column) and the outage duration of different states. So we plot an pivot table below (Also, only show some rows).

[ ]:
```python
pivot_table = df2.pivot_table(index='U.S._STATE', columns='CLIMATE.CATEGORY',␣
↪values='OUTAGE.DURATION', aggfunc='mean')
rows = pivot_table.loc[['Wisconsin', 'Hawaii', 'Massachusetts', 'Tennessee',␣
↪'New York']]
print(rows)
with open('table3.txt','w') as f:
    f.write(rows.to_markdown())
```

```
CLIMATE.CATEGORY          cold       normal         warm
U.S._STATE
Wisconsin         12545.555556  3962.555556  1605.000000
Hawaii             1367.000000   205.500000  1224.500000
Massachusetts       160.333333  1159.384615   721.000000
Tennessee          1476.583333  1015.750000   341.857143
New York           8914.576923  3673.562500  6092.833333
```

To observe it more intuitively, we plot the table below.

[ ]:
```python
fig = px.bar(pivot_table, title = 'Average Outage Duration of Different Climate␣
↪Category of Different States')
fig.update_layout(xaxis_title='U.S. State', yaxis_title='Average Outage␣
↪Duration')
fig.show()
export_plotly_fig(fig, filename='bar_climate_category.html')
```

It seems that the outage duration differ a lot in different climate. The warmer, the longer it lasts. But it's not necessary that it's true, so we'd check that in the hypothesis test below.

### 1.1.2 Assessment of Missingness

**NMAR Analysis** One of the column in our dataset with missing values that is possibly NMAR is the `OUTAGE.DURATION` column. The reason for its missingness is most likely due to some reasons that why the `OUTAGE.START.DATE` or more frequently, the `OUTAGE.RESTORATION.DATE` is missing. Any of these columns miss will lead to the `OUTAGE.DURATION` miss. However, there's no clear reasoning why they miss recorded in the dataset. If we want additional data to record the reason why the start and restoration datetime of a outage is missing, then we could make it MAR.

**Missingness Dependency** We constructed permutation tests to determine the relationship. We've decided to test the dependency between the missingness of `OUTAGE.DURATION` with two columns: `OUTAGE.RESTORATION.DATE` and `CAUSE.CATEGORY.DETAIL`.

**OUTAGE.DURATION and OUTAGE.RESTORATION.DATE (MAR)** To make our code suitable for more different columns, we first build a universal function to check whether the missingness of one column depends on another column.

It's no doubt that the missingness of the `OUTAGE.DURATION` depend on the missingness `OUTAGE.RESTORATION.DATE`. We can easily figure out this from both observing the data and reasoning in the reality. And now, we'd like check it by performing permutation tests.

More specifically, it means that the missingness of `OUTAGE.RESTORATION.DATE` affects the missingness of `OUTAGE.DURATION`. So, we first create an additional column `OUTAGE.RESTORATION.MISSING` to record the whether the `OUTAGE.RESTORATION.DATE` is missing.

[ ]:
```
df['OUTAGE.RESTORATION.MISSING'] = df['OUTAGE.RESTORATION.DATE'].isna()
print(df['OUTAGE.RESTORATION.MISSING'].tail())
```

```
1530    False
1531     True
1532    False
1533    False
1534     True
Name: OUTAGE.RESTORATION.MISSING, dtype: bool
```

Since it's a category column, we use TVD(Total Variation Distance) to perform permutation test. Recall that:

$$TVD(X,Y) = \frac{1}{2}\sum_{i=1}^{n}|X_i - Y_i|$$

The null hypothesis for the permutation test is that the specific column does not depend on another column. So if p-value is less than the significance level, we reject the null hypothesis, which means that we may think the column depend on another. Otherwise, we fail to reject the null hypothesis, which means that it's more possible to believe the column does not depend on another.

Below, we make codes to compute the TVD, run the permutation tests of 500 rounds, set the significance level to 0.05, calculate the p-value, draw the plot and get the conclusion.

[ ]:
```
def TVD(df, col1, col2):
    pivoted = (
        df
        .pivot_table(index=col1, columns=col2, aggfunc='size')
        .apply(lambda x: x / x.sum() ).fillna(0)
    )
    return pivoted.diff(axis=1).iloc[:, -1].abs().sum() / 2

def report_perm(original_tvd, p_value, tvds, col1, col2):
    # plot
    fig = px.histogram(pd.DataFrame(tvds), x=0, nbins=20,␣
 ↪histnorm='probability')
    fig.add_vline(x=original_tvd, line_color='red')
```

```
    fig.add_annotation(text=f'<span style="color:red">Observed TVD =␣
↪{round(original_tvd, 2)}, p_value = {round(p_value ,2)}</span>',
                    x= 0.4, showarrow=False, y=0.1)
    fig.update_layout(title = f"Permutation Test whether <br> {col1} Depends on␣
↪<br> {col2}", xaxis_title="TVD")
    fig.show()
    export_plotly_fig(fig, f'permutation_test_{col1}_{col2}.html')

    # make conclusion
    if p_value < 0.05:
        print(f'Reject the null hypothesis, \nit\'s more likely that {col1}␣
↪depends on {col2}.')
    else:
        print(f'Fail to reject the null hypothesis, \nit\'s more likely that␣
↪{col1} does not depend on {col2}.')

def permutation_test(df, col1, col2, n_permutations=500):
    original_tvd = TVD(df, col1, col2)

    permuted_tvds = []
    shuffled = df.copy()
    for _ in range(n_permutations):
        shuffled[col1] = np.random.permutation(df[col1])
        permuted_tvds.append(TVD(shuffled, col1, col2))

    p_value = np.mean([tvd >= original_tvd for tvd in permuted_tvds])
    report_perm(original_tvd, p_value, permuted_tvds, col1, col2)
    return original_tvd, p_value, permuted_tvds
```

We use the codes to check whether the `OUTAGE.DURATION` missingness depends on the `OUTAGE.RESTORATION` missingness.

```
[ ]: df['OUTAGE.DURATION.MISSING'] = df['OUTAGE.DURATION'].isna()
tvd, p, tvds = permutation_test(df, 'OUTAGE.DURATION.MISSING', 'OUTAGE.
↪RESTORATION.MISSING')
print(tvd, p)
```

```
Reject the null hypothesis,
it's more likely that OUTAGE.DURATION.MISSING depends on
OUTAGE.RESTORATION.MISSING.
1.0 0.0
```

The result shows that p-value is 0.00, since the significance level is 5%, we reject the null hypothesis. Therefore, we conclude that it is highly possible that the missingness of the `OUTAGE.DURATION` depends on the missingness of the `OUTAGE.RESTORATION`.

We may get the same conclusion that the missingness of the `OUTAGE.DURATION` is also depends on the missingness of the `OUTAGE.START`, but the TVDs of the permutation tests should be larger than the previous ones, since the `OUTAGE.START` data miss less, which can be easily seen from the dataset. The result below shows that it's rational.

```
[ ]: df['OUTAGE.START.MISSING'] = df['OUTAGE.START'].isna()
tvd, p, tvds = permutation_test(df, 'OUTAGE.DURATION.MISSING', 'OUTAGE.START.
 ↪MISSING')
print(tvd, p)
```

```
Reject the null hypothesis,
it's more likely that OUTAGE.DURATION.MISSING depends on OUTAGE.START.MISSING.
0.9678688524590164 0.0
```

We can easily find that, on the other hand, the missingness of the `CAUSE.CATEGORY.DETAIL`, have nothing to do with the missingness of `OUTAGE.DURATION`. Since it only dipicts some additional information of the `CAUSE.CATEGORY`. So we should find that the missingness of the `CAUSE.CATEGORY.DETAIL` is independent of the missingness of the `CAUSE.CATEGORY.DETAIL`. Below, we will check our conjecture.

```
[ ]: df['CAUSE.CATEGORY.DETAIL.MISSING'] = df['CAUSE.CATEGORY.DETAIL'].isna()
tvd, p, tvds = permutation_test(df, 'OUTAGE.DURATION.MISSING', 'CAUSE.CATEGORY.
 ↪DETAIL')
print(tvd, p)
```

```
Fail to reject the null hypothesis,
it's more likely that OUTAGE.DURATION.MISSING does not depend on
CAUSE.CATEGORY.DETAIL.
0.009900990099009906 0.89
```

The plot result shows that p-value is 0.89, which is largely greater than the significance level of 5%, so we fail to reject the null hypothesis. Hence, we conclude that the missingness of `OUTAGE.DURATION` does not depend on the missingness of `CAUSE.CATEGORY.DETAIL`.

#### 1.1.3 Hypothesis Testing

#### 1.1.4 Permutation Test

Going back to our investigation topic, we are investigating does climate category affect the duration of a power outage. Recalling the plot in the interesting aggregates part, we intuitively propose the idea that there are some relationships between the climate and the duration of power outages. But observation alone cannot be a good indicator. So we determined a good way to test is making a hypothesis test.

We think that permutation test on the distribution of outage durations in the warm climate areas and the distribution in the cold climate areas to see whether there is an actual increase in duration time to wamrer areas or not.

Null hypothesis: the outage duration time in the warm climate and the cold climate comes from the same distribution.

Alternative hypothesis: the outage duration time in the warm climate and the cold climate comes from different distributions.

Significance level: 5%.

Number of permutation test rounds: 500.

Test statistics: the outage duration is numeric data rather than categorical. And recall that numeric data uses diff of means, which the categorical data uses TVD. Thereby, we use the diff of means as our test statistics.

The value is numeric rather than category, so we use the absolute difference between the mean of the two groups to check it. Recall the formula:

$$diff\_of\_means(A, B) = |\frac{1}{n}\sum_{i=1}^{n} A_i - \frac{1}{m}\sum_{i=1}^{m} B_i|$$

Using the similar way, we write codes of the permutation test below.

[ ]:

```python
def diff_of_means(df, col1, col2, v1, v2):
    mean1 = df[df[col2] == v1][col1].mean()
    mean2 = df[df[col2] == v2][col1].mean()
    return abs(mean1 - mean2)
def report_hyp(observed, p_value, simluated, col1, col2, v1, v2):
    # plot
    fig = px.histogram(pd.DataFrame(simluated), x=0, nbins=20, histnorm='probability')
    fig.add_vline(x=observed, line_color='red')
    fig.add_annotation(text=f'<span style="color:red">Observed = {round(observed, 2)}, p_value = {round(p_value ,2)}</span>',
                       x= 0.4, showarrow=False, y=0.1)
    fig.update_layout(title = f"Empirical Distribution to check whether <br> {col1} Have Different Distribution when <br> {col2} is {v1} and {v2}", xaxis_title="Diff of Means")
    fig.show()
    export_plotly_fig(fig, filename=f'hyp_{col1}_{col2}_{v1}_{v2}.html')

    # make conclusion
    if p_value < 0.05:
        print(f'Reject the null hypothesis, \nit\'s more likely that {col1} is different when {col2} is {v1} and {v2}.')
    else:
        print(f'Fail to reject the null hypothesis, \nit\'s more likely that {col1} is same when {col2} is {v1} and {v2}.')
def hypothesis_test(df, col1, col2, v1, v2, rounds=500):
    observed = diff_of_means(df, col1, col2, v1, v2)
    df2 = df.copy()
    simulated = np.zeros(rounds)
    for _ in range(rounds):
        df2[col2] = df[col2].sample(frac=1).reset_index(drop=True)
        simulated[_] = diff_of_means(df2, col1, col2, v1, v2)
    p_value = np.mean(simulated >= observed)
    report_hyp(observed, p_value, simulated, col1, col2, v1, v2)
```

```
    return observed, p_value
```

[ ]:
```
hypothesis_test(df, 'OUTAGE.DURATION', 'CLIMATE.CATEGORY', 'warm', 'cold')
```

```
Fail to reject the null hypothesis,
it's more likely that OUTAGE.DURATION is same when CLIMATE.CATEGORY is warm and
cold.
```

[ ]:
```
(160.36121774568983, 0.712)
```

### 1.1.5 Conclusion

Out of our expectation, the result shows that p-value is 0.71, which is larger than 0.05, so we fail to reject the null hypothesis, which means that the outage duration time in the warm climate and the cold climate comes from the same distribution. That is, the outage duration have no relationship with the climate.

To check the correctness, we make a totally random column to check our correctness, it should be a same distribution below.

[ ]:
```
df['RANDOM'] = np.random.choice(['a','b'],df.shape[0],p=[0.4,0.6])
hypothesis_test(df, 'OUTAGE.DURATION', 'RANDOM', 'a', 'b')
```

```
Fail to reject the null hypothesis,
it's more likely that OUTAGE.DURATION is same when RANDOM is a and b.
```

[ ]:
```
(261.0528683297448, 0.448)
```

And the below should come from different distribution. We divide duration into two groups by whether it is greater than it's mean value. So obviously it's different distribution.

[ ]:
```
mean = df['OUTAGE.DURATION'].mean()
df['NO_RANDOM'] = df['OUTAGE.DURATION'] > mean
hypothesis_test(df, 'OUTAGE.DURATION', 'NO_RANDOM', True, False)
```

```
Reject the null hypothesis,
it's more likely that OUTAGE.DURATION is different when NO_RANDOM is True and
False.
```

[ ]:
```
(7224.882703670063, 0.0)
```

Similarily, we can use the same function to test whether different states affect the distribution.

[ ]:
```
hypothesis_test(df, 'OUTAGE.DURATION', 'U.S._STATE', 'New York', 'Tennessee')
```

```
Reject the null hypothesis,
it's more likely that OUTAGE.DURATION is different when U.S._STATE is New York
and Tennessee.
```

[ ]:
```
(4992.989400921659, 0.0)
```

It shows that it's highly possible that the outage duration differs a lot in the two states.