Analyzing Power Outages

Name(s): Matteo Perona

Website Link: https://matteoperona.github.io/usa-power-outage-analysis/

Introduction

Questions:

- Which region tends to have the longest outages?
- Which causes tend to produce the longest outages?
- Do the total repair cost and the amount of outage time follow the same distribution?
- Is there a relationship between ANOMALY.LEVEL (ONI index) and the severity of the outage? This could realte to the total cost, the demand lost, the outage time, ...
- **Do states whose most common cause of outage is intentional attack have more outages per capita?**

Decided Question: **Do states whose most common cause of outage is intentional attack have more outages per capita?**

Code

```python
import pandas as pd
import numpy as np
import os

import plotly.express as px
pd.options.plotting.backend = 'plotly'
```

Cleaning and EDA

Note: First step of cleaning involved removing unnecessary rows and columns from the .xlsx file and exporting as a .csv

Cleaning and Reformatting Temporal Data

```python
# Create the file path for the dataset and load it into the notebook as a dataframe
outage_fp = os.path.join('data', 'outage.csv')
outage = pd.read_csv(outage_fp)
```

```python
print(outage[['OUTAGE.START.DATE', 'OUTAGE.START.TIME', 'OUTAGE.RESTORATION.DATE', 'OUTAGE.RESTORATION.TIME']].head().to_markdown(index=False))
```

```
| OUTAGE.START.DATE         | OUTAGE.START.TIME   | OUTAGE.RESTORATION.DATE    | OUTAGE.RESTORATION.TIME   |
|:--------------------------|:--------------------|:---------------------------|:--------------------------|
| Friday, July 01, 2011     | 5:00:00 PM          | Sunday, July 03, 2011      | 8:00:00 PM                |
| Sunday, May 11, 2014      | 6:38:00 PM          | Sunday, May 11, 2014       | 6:39:00 PM                |
| Tuesday, October 26, 2010 | 8:00:00 PM          | Thursday, October 28, 2010 | 10:00:00 PM               |
| Tuesday, June 19, 2012    | 4:30:00 AM          | Wednesday, June 20, 2012   | 11:00:00 PM               |
| Saturday, July 18, 2015   | 2:00:00 AM          | Sunday, July 19, 2015      | 7:00:00 AM                |
```

```python
# Create new series for outage start and outage restoration which contain timestamps
# To do this, add the respective dates to the times and convert each value to a timestamp
outage['OUTAGE.START'] = (outage['OUTAGE.START.DATE'] + ", " + outage['OUTAGE.START.TIME']) \
    .apply(lambda x: pd.Timestamp(x))
outage['OUTAGE.RESTORATION'] = (outage['OUTAGE.RESTORATION.DATE'] + ", " + outage['OUTAGE.RESTORATION.TIME']) \
    .apply(lambda x: pd.Timestamp(x))

# Drop the old time-keeping values from the dataframe
outage = outage.drop(columns=['OUTAGE.START.DATE', 'OUTAGE.START.TIME', 'OUTAGE.RESTORATION.DATE', 'OUTAGE.RESTORATION.TIME'])

print(outage[['OUTAGE.RESTORATION', 'OUTAGE.START']].head().to_markdown(index=False))
outage[['OUTAGE.RESTORATION', 'OUTAGE.START']].dtypes
```

```
|   OUTAGE.RESTORATION |   OUTAGE.START |
|---------------------:|---------------:|
|          1.30972e+18 |    1.30954e+18 |
|          1.39983e+18 |    1.39983e+18 |
|          1.2883e+18  |    1.28812e+18 |
|          1.34023e+18 |    1.34008e+18 |
|          1.43729e+18 |    1.43718e+18 |
```

```
OUTAGE.RESTORATION    datetime64[ns]
OUTAGE.START          datetime64[ns]
dtype: object
```

Column Dtypes

```python
# open output in a text editor to view in full
# Finds the datatype for each column in the outage dataframe
print(outage.dtypes.to_markdown(index=True))
```

```
|                       | 0              |
|:----------------------|:---------------|
| OBS                   | int64          |
| YEAR                  | int64          |
| MONTH                 | float64        |
| U.S._STATE            | object         |
| POSTAL.CODE           | object         |
| NERC.REGION           | object         |
| CLIMATE.REGION        | object         |
| ANOMALY.LEVEL         | float64        |
| CLIMATE.CATEGORY      | object         |
| CAUSE.CATEGORY        | object         |
| CAUSE.CATEGORY.DETAIL | object         |
| HURRICANE.NAMES       | object         |
| OUTAGE.DURATION       | float64        |
| DEMAND.LOSS.MW        | float64        |
| CUSTOMERS.AFFECTED    | float64        |
| RES.PRICE             | float64        |
| COM.PRICE             | float64        |
| IND.PRICE             | float64        |
| TOTAL.PRICE           | float64        |
| RES.SALES             | float64        |
| COM.SALES             | float64        |
| IND.SALES             | float64        |
| TOTAL.SALES           | float64        |
| RES.PERCEN            | float64        |
| COM.PERCEN            | float64        |
| IND.PERCEN            | float64        |
| RES.CUSTOMERS         | int64          |
| COM.CUSTOMERS         | int64          |
| IND.CUSTOMERS         | int64          |
| TOTAL.CUSTOMERS       | int64          |
| RES.CUST.PCT          | float64        |
| COM.CUST.PCT          | float64        |
| IND.CUST.PCT          | float64        |
| PC.REALGSP.STATE      | int64          |
| PC.REALGSP.USA        | int64          |
| PC.REALGSP.REL        | float64        |
| PC.REALGSP.CHANGE     | float64        |
| UTIL.REALGSP          | int64          |
| TOTAL.REALGSP         | int64          |
| UTIL.CONTRI           | float64        |
| PI.UTIL.OFUSA         | float64        |
| POPULATION            | int64          |
| POPPCT_URBAN          | float64        |
| POPPCT_UC             | float64        |
| POPDEN_URBAN          | float64        |
| POPDEN_UC             | float64        |
| POPDEN_RURAL          | float64        |
| AREAPCT_URBAN         | float64        |
| AREAPCT_UC            | float64        |
| PCT_LAND              | float64        |
| PCT_WATER_TOT         | float64        |
| PCT_WATER_INLAND      | float64        |
| OUTAGE.START          | datetime64[ns] |
| OUTAGE.RESTORATION    | datetime64[ns] |
```

It looks as though all datatypes are already in order.

Analyzing Unique Values for Each Column

```python
# Open output in a text editor to get a better understanding
# Creates a list of unique values for each column in the dataframe
unique_values = outage.apply(lambda col: col.unique())

# for col, vals in unique_values.items():
#     print(col)
#     print(vals)
#     print('\n\n')
```

Further inspection of unique values shows that null values seem to be in order. This output is useful in order to think about how to analyze each column, and how columns might be aggregated.

```python
print(outage[['YEAR', 'MONTH', 'U.S._STATE', 'CAUSE.CATEGORY']].head().to_markdown(index=False))
```

```
|   YEAR |   MONTH | U.S._STATE   | CAUSE.CATEGORY     |
|-------:|--------:|:-------------|:-------------------|
|   2011 |       7 | Minnesota    | severe weather     |
|   2014 |       5 | Minnesota    | intentional attack |
|   2010 |      10 | Minnesota    | severe weather     |
|   2012 |       6 | Minnesota    | severe weather     |
|   2015 |       7 | Minnesota    | severe weather     |
```

Univariate, Bivariate, and Aggragate Analysis

Years and Seasons

```python
# Creates a histogram of the YEAR column in the outage dataframe
fig = outage.YEAR.hist(title='Num Outages Each Year')
# This line saves the plot as an html file in the assets folder
# There are many lines which do the same thing though this notebook, so I will only lable this one
fig.write_html('./assets/num-outages-each-year.html', include_plotlyjs='cdn')
fig
```

It looks like 2011 had the most outages by far. It would be interesting to look into why. Additionally, it seems as though the number of outages in the years leading up to and away from 2011 seem to follow a bell-like shape.

```python
# creates a histogram of the MONTH column in the dataframe
fig = outage.MONTH.hist(title='Num Outages By Month')
fig.write_html('./assets/num-outages-by-month.html', include_plotlyjs='cdn')
fig
```

Summer seems to have the most outages followed by winter with spring and fall seeming to have the fewest.

Looking into Extent of Outages

```python
fig = outage['OUTAGE.DURATION'].hist(title='Outage Duration Distribution [in Minutes]')
fig.write_html('./assets/outage-duration-distribution.html', include_plotlyjs='cdn')
fig
```

```python
fig = outage['OUTAGE.DURATION'][outage['OUTAGE.DURATION'] <= 999].hist(title='Outage Duration Distribution (0-999 minutes)')
fig.write_html('./assets/outage-duration-distribution-first-bin.html', include_plotlyjs='cdn')
fig
```

The outage durations seem to be heavily right skewed. Even when I zoom into the first bin, creating the second plot, the pattern holds. This shows that the majority of shortages last less than 1000 minutes.

```python
fig = outage['DEMAND.LOSS.MW'].hist(title='Distribution of Peak Demand Lost Diuring an Outage [in Megawatt]')
fig.write_html('./assets/distribution-peak-demand-lost-during-outage.html', include_plotlyjs='cdn')
fig
```

```python
fig = outage['DEMAND.LOSS.MW'][outage['DEMAND.LOSS.MW'] <= 499].hist(title='Distribution of Peak Demand Lost Diuring an Outage (0-500 Megawatt)')
fig.write_html('./assets/distribution-peak-demand-lost-during-outage-first-bin.html', include_plotlyjs='cdn')
fig
```

The peak demand lost also shows a heavy right skew with the great majority of outages losing 0-500 Megawatt of Peak Demand. This seems correlate with the outage duration column.

- Note: the DEMAND.LOSS.MW column is missing about half its values. Reference the *Plot Null Counts of Columns with Missing Data* section

```python
fig = outage['CUSTOMERS.AFFECTED'].hist(title='Distribution of Customers Affected by Each Outage')
fig.write_html('./assets/dist-customers-affected-each-outage.html', include_plotlyjs='cdn')
fig
```

We see another right skew, but this time it's far less prominent. It seems as though most outages are relatively short lasting (0-999 minutes), lose relatively little power (between 0-500 [one could even argue 0-60 MW] of peak demand), and affect relatively few customers (0-100k people). These statements are relative to the hard skews bringing the maximums far from the points of highest density.

```python
fig = outage['ANOMALY.LEVEL'].hist(title='Anomaly Level (ONI) Distribution')
fig.write_html('./assets/anomaly-level-dist.html', include_plotlyjs='cdn')
fig
```

Preface: "The ONI is the rolling 3-month average temperature anomaly—difference from average—in the surface waters of the east-central tropical Pacific, near the International Dateline. Index values of +0.5 or higher indicate El Niño. Values of -0.5 or lower indicate La Niña." (https://www.climate.gov/news-features/understanding-climate/climate-variability-oceanic-ni%C3%B1o-index#:~:text=The%20ONI%20is%20the%20rolling,or%20lower%20indicate%20La%20Ni%C3%B1a.)

It seems as thought outages tend to be most frequent around the -0.3 anomaly level. This is strange as -0.3 indicates neither an El Niño nor a La Niña weather event, and it's not at 0.0 (between the two). This plot brings up many questions. What relevance does -0.3 have to the frequency of power outages? Does is it really relevant or was this distribution random? Could this distribution indicate that outages are more frequent during La Niña events (<= -0.5).

```python
print('Correlation:', outage['OUTAGE.DURATION'].corr(outage['ANOMALY.LEVEL']))
fig = outage.plot(kind='scatter', y='OUTAGE.DURATION', x='ANOMALY.LEVEL', title='Outage Duration (Mins) by Anomaly Level (ONI)')
fig.write_html('./assets/outage-duration-by-anomaly-level.html', include_plotlyjs='cdn')
fig
```

```
Correlation: 0.008909995469844948
```

There does not appear to be a correlation between the outage duration and the anomaly level. Let's move onto the next section to look at the relationship between outage frequency and location.

Outages by Location

```python
# Groups rows by state and aggregates by the count of each group.
# Takes those counts and sorts them
# Creates a horizontal bar plot of outage observation counts for each state
fig = outage.groupby(by='U.S._STATE').count().OBS.sort_values().plot(kind='barh', width=800, height=1000, title='Outages Counts by State')
fig.write_html('./assets/outage-counts-state.html', include_plotlyjs='cdn')
fig
```

This plot shows the number of outages that were obseved in each state. It is not very interesting as it stands, the results speak for themselves. More outages happen in more populous states. It would be interesting to see the same plot adjusted for population and another adjusted for square footage.

```python
# Creates a series with the number of outages in each state
outage_dist_by_state = outage.groupby(by='U.S._STATE').count().OBS.sort_values()
# Creates a series of each state's mean population from 2000-2016
pop_by_state = outage.groupby(by='U.S._STATE').mean().POPULATION
# Creates a horizontal bar plot displaying the number of outages per 10k people in each state
fig = (outage_dist_by_state / pop_by_state * 10_000).sort_values().plot(kind='barh', width=800, height=1000, title='Outages per 10k People by State')
fig.write_html('./assets/outages_per_capita_state.html', include_plotlyjs='cdn')
fig
```

Ah! Much more interesting. Our first bivariate plot! It seems as though Delaware is has by far the most outages per person. More than double DC (the second highest). Why might Delaware's number of outages be so disproportionately high given their population?

Side Note: Texas and California (the top two of the previous plot) are middle of the pack. This is a good example for the importance of bivariate analysis. Also, note that the second highest is DC which is very clse to Delaware.

```python
# Counts of outages in each NERC region
outages_by_NERC = outage.groupby(by='NERC.REGION').count().OBS.sort_values()
# Hbar plot on the results from the previous line
fig = outages_by_NERC.plot(kind='barh', width=1100, height=500, title='Outages Counts by NERC Weather Region (North American Electric Reliability Corporation)')
fig.write_html('./assets/outage-counts-NERC.html', include_plotlyjs='cdn')
fig
```

One can see that, again the most populous weather regious seem to populate the top of the plot. Unsurprising

```python
# Finds the number of outages in each NERC region per 1M people
NERC_region_pop = outage.groupby(by=['NERC.REGION', 'U.S._STATE'])\
    .mean().reset_index().groupby('NERC.REGION').POPULATION.sum()
# Finds all NERC regions with fewer than ten outages
regions_fewer_ten_outages = outages_by_NERC[(outages_by_NERC < 10)].index
# We drop all weather recions with fewer than ten outages so as not to muddy
(
    (outages_by_NERC / NERC_region_pop * 1_000_000)
    .drop(regions_fewer_ten_outages).sort_values()
    .plot(
        kind='barh',
        width=1100,
        height=500,
        title='Outages per 1.0M People by NERC Weather Region (regions with < 10 outages removed)'
        )
)
```

One might have expected RFC, with Delaware and DC, to be the highest, but it is close to the middle. I suspect that I made a mistake when calculating the populations of each NERC region. I will come back to this section if I have time.

Lets look quickly at the regular weather regions to get a point of comparison.

```python
# Counts of outages in each weather region
outages_by_NERC = outage.groupby(by='CLIMATE.REGION').count().OBS.sort_values()
# HBar of above information
fig = outages_by_NERC.plot(kind='barh', width=800, height=500, title='Outages Counts by Weather Region')
fig.write_html('./assets/outage-counts-weather-region.html', include_plotlyjs='cdn')
fig
```

Outages by Category

```python
# Hbar of the number of outages by cause category by grouping by CAUSE.CATEGORY and aggregating with count()
fig = outage.groupby(by='CAUSE.CATEGORY').count().OBS.sort_values().plot(kind='barh', title='Number of Outages by Cause Category', width=800)
fig.write_html('./assets/outages-cause.html', include_plotlyjs='cdn')
fig
```

It seems as though severe weather is to blame for most outages, but it is interesting to see that a substantial portion of outages are intentional attacks. It would be interesting to look into how what types of intentional attack and severe weather are most prevalent. Fist let's also look into the number of intentional attacks that happen per capita (mean of POPULATION of each group of observations).

```python
# Finds the mean of the populations in each cause category group
mean_pop_cause = outage.groupby(by='CAUSE.CATEGORY').mean()['POPULATION']
# Outage counts for reach cause category
total_obs_cause = outage.groupby(by='CAUSE.CATEGORY').count()['OBS']
# hbar of each cause count divided by its respective mean population
fig = (total_obs_cause / mean_pop_cause).sort_values().plot(kind='barh', title='Outages per Capita by Cause')
fig.write_html('./assets/outages-per-captia-cause.html', include_plotlyjs='cdn')
fig
```

Here we can see the same relationship from above, but the difference between the top two bars and the rest seems to be more extreme. Also, the intentional attack bar reaches much closer to the severe weather bar. This may indicate a relationship between per capita effects and intentional attacks, but it's difficult to say as this is an imperfect metric. The POPULATION column measures the total population in the US state where the outage was observed, not the affected population, making this metric very shaky.

```python
# Hbar of the outage counts of all severe weather's sub categories
fig = outage[(outage['CAUSE.CATEGORY'] == 'severe weather')].groupby(by='CAUSE.CATEGORY.DETAIL')\
    .count().OBS.sort_values()\
    .plot(kind='barh', title='Outages by Severe Weather', height=800, width=800)
fig.write_html('./assets/cause-severe-weather.html', include_plotlyjs='cdn')
fig
```

It looks as though thunderstorms are responsible for the greatest number of power outages. Could this be because they are the most common storm with destructive potential?

```python
# plots counts for subcategories of intentional attack
fig = outage[(outage['CAUSE.CATEGORY'] == 'intentional attack')]\
    .groupby(by='CAUSE.CATEGORY.DETAIL').count().OBS.sort_values()\
    .plot(kind='barh', title='Outages by Intentional Attack')
fig.write_html('./assets/cause-intentional-attack.html', include_plotlyjs='cdn')
fig
```

It seems as thought most instentional attacks are vandalism. This gives less information than I would have hoped. Next, lets look at the cause breakdowns for each state using a pivot table.

```python
# creates a pivot table which counts the number of outages indexed by state with cause as columns
cause_by_state = pd.pivot_table(outage, columns=['CAUSE.CATEGORY'], index=['U.S._STATE'], values='OBS', aggfunc='count')
# adds the total for each state back in to allow for sorting
cause_by_state = cause_by_state.assign(total=cause_by_state.sum(axis=1))
# sorts by total and drops the column
cause_by_state = cause_by_state.sort_values(by='total').drop(columns=['total'])
print(cause_by_state.fillna(0).head().to_markdown())
```

```
| U.S._STATE   |   equipment failure |   fuel supply emergency |   intentional attack |   islanding |   public appeal |   severe weather |   system operability disruption |
|:-------------|--------------------:|------------------------:|---------------------:|------------:|----------------:|-----------------:|--------------------------------:|
| Alaska       |                   1 |                       0 |                    0 |           0 |               0 |                0 |                               0 |
| South Dakota |                   0 |                       0 |                    0 |           2 |               0 |                0 |                               0 |
| North Dakota |                   0 |                       1 |                    0 |           0 |               1 |                0 |                               0 |
| Montana      |                   0 |                       0 |                    1 |           2 |               0 |                0 |                               0 |
| Mississippi  |                   0 |                       0 |                    3 |           0 |               0 |                0 |                               1 |
```

```python
# plots the pivot table created above
fig = cause_by_state.plot(kind='barh', width=800, height=1300, title='Outages by Cause for each State')
fig.write_html('./assets/cause-state.html', include_plotlyjs='cdn')
fig
```

This is interesting, we see the same plot from the *Outages by Location* section, but with more context. Let's look at this plot adjusted for population as we did before, but first, lets also look at the distributions of Outage Causes for each State by normalizing the rows.

```python
# normalizes the pivot table created above
cause_by_state_norm = cause_by_state.apply(lambda x: x / cause_by_state.sum(axis=1))
# plots the normalized data
fig = cause_by_state_norm.plot(kind='barh', width=800, height=1300, title='Distribution of Outage Causes for each State')
fig.write_html('./assets/cause-state-norm.html', include_plotlyjs='cdn')
fig
```

This confirms our results from above; we are mostly seeing severe weather and intentional attack as the primary causes. Now, let's scale these values to understand the data per capita.

```python
# scales the pivot table by state population to give per capita view of the data
cause_by_state_scaled = cause_by_state.apply(lambda x: x / pop_by_state * 10_000)
# calculates the total of each row and adds it into the df as total for sorting
cause_by_state_scaled = cause_by_state_scaled.assign(total = cause_by_state_scaled.sum(axis=1))
# sorts the df using total and then drops total from the df
cause_by_state_scaled = cause_by_state_scaled.sort_values('total').drop(columns=['total'])

# plots the scaled pivot table
fig = cause_by_state_scaled.plot(kind='barh', width=800, height=1300, title='Outages per 10k People by Cause for each State')
fig.write_html('./assets/cause-state-per-capita.html', include_plotlyjs='cdn')
fig
```

Ah! This plot immediately gives us more context for Delaware's outlandish number of outages per capita. There are a disproportionate number of intentional attack related outages in Delaware. In fact, many of the higher ranking states tend to have high instances of intentional attack. Might states with high rates of intentional attack have more outages per person overall? Let's explore this possible trend.

```python
# adds the total back in in case one wants to sort the data descending
df = cause_by_state_scaled.assign(
    total = cause_by_state_scaled.sum(axis=1)
    ).sort_values(by='total', ascending=True)
# Calculates the correlation and prints it
print('Correlation between total outages per capidta and severe:', df['total'].corr(df['intentional attack']))
# Scatter plots the intentional attack by total
fig = px.scatter(
    data_frame=df, x='total', y='intentional attack', color=df.index, color_discrete_sequence=px.colors.qualitative.G10,
    title = 'Intentional Attack Outages (per 10k people) compared to total Outages (per 10k people)')
fig.write_html('./assets/intentional-attack-scatter-total-outages.html', include_plotlyjs='cdn')
fig
```

```
Correlation between total outages per capidta and severe: 0.9542374010674626
```

It seems like we're onto something. Hoverver, there could still be confounders. Below I created the same graph but removed Delaware since it seems to be an outlier.

```python
# adds the total back in in case one wants to sort the data descending and removes Delaware from the data
df = cause_by_state_scaled.assign(
    total = cause_by_state_scaled.sum(axis=1)
    ).sort_values(by='total', ascending=True).drop(['Delaware'])
# Calculates the correlation and prints it
print('Correlation between total outages per capidta and severe:', df['total'].corr(df['intentional attack']))
# Scatter plots the intentional attack by total
fig = px.scatter(
    data_frame=df, x='total', y='intentional attack', color=df.index, color_discrete_sequence=px.colors.qualitative.G10,
    title = 'Intentional Attack Outages (per 10k people) compared to total Outages (per 10k people) [Delaware Removed]')
fig.write_html('./assets/intentional-attack-scatter-total-no-delaware.html', include_plotlyjs='cdn')
fig
```

```
Correlation between total outages per capidta and severe: 0.8246248841759625
```

We can see that the correlation still holds, albeit, to a lesser degree.

Assessment of Missingness

Plot Null Counts of Columns with Missing Data

```python
# counts all null values in each column outages
null_counts = outage.isna().sum().sort_values()
# plots hbar of missing value conts for each column with missing values
fig = null_counts[null_counts > 0].plot(
    kind='barh',
    width=1100,
    height=600,
    title='Number of Missing Values by Column in the Outage Dataframe',
)
fig.write_html('./assets/missing-values-by-column.html', include_plotlyjs='cdn')
fig
```

Missingness Dependency

Testing Dependent

Null Hypothesis: The distribution of CAUSE.CATEGORY.DETAIL is the same when CAUSE.CATEGORY name is missing and when it is not missing
Alt Hypothesis: The distribution of CAUSE.CATEGORY.DETAIL is different when CAUSE.CATEGORY name is missing as opposed to when it is not missing

```python
# Variables for columns we are testing to enable hot switching
indep_var = 'CAUSE.CATEGORY'
q_var = 'CAUSE.CATEGORY.DETAIL'
```

```python
# generate pivot table to display missingness of q_var with relation to indep_var
df_indep = (
    outage
    .assign(missing = outage[q_var].isna())
    .pivot_table(index=indep_var, columns='missing', aggfunc='size')
).fillna(0)

# normalize the df
df_indep = df_indep / df_indep.sum()
# plot the distribution
fig = df_indep.plot(
    kind='barh',
    title='Observed Distribution of Cause Category Detail Conditional on Cause Category',
    barmode='group',
    height=500
    )
fig.write_html('./assets/observed-dist-dependent.html', include_plotlyjs='cdn')
fig
```

```python
print(df_indep.to_markdown())
```

```
| CAUSE.CATEGORY                |     False |      True |
|:------------------------------|----------:|----------:|
| equipment failure             | 0.0451552 | 0.0254777 |
| fuel supply emergency         | 0.0301035 | 0.0403397 |
| intentional attack            | 0.348071  | 0.101911  |
| islanding                     | 0         | 0.0976645 |
| public appeal                 | 0         | 0.146497  |
| severe weather                | 0.541863  | 0.397028  |
| system operability disruption | 0.0348071 | 0.191083  |
```

```python
# calculates the tvd for between missing and not missing cause category details distributions for each cause category
observed_tvd = df_indep.diff(axis=1).iloc[:, -1].abs().sum() / 2
observed_tvd
```

```
0.41067323382726845
```

Simulation

```python
# number of times to repeat permutations and calculate TVD
n_repetitions = 500
shuffled = outage.copy()

tvds = []
for _ in range(n_repetitions):
    # Create permutation
    shuffled[indep_var] = np.random.permutation(shuffled[indep_var])

    # Computing and storing the TVD.
    pivoted = (
        shuffled
        .assign(missing = shuffled[q_var].isna())
        .pivot_table(index=indep_var, columns='missing', aggfunc='size')
        .apply(lambda x: x / x.sum())
    )

    tvd = pivoted.diff(axis=1).iloc[:, -1].abs().sum() / 2
    tvds.append(tvd)
```

```python
# plotting the imerical distribution of the TVD
fig = px.histogram(pd.DataFrame(tvds), x=0, nbins=50, histnorm='probability',
                   title='Empirical Distribution of the TVD')
fig.add_vline(x=observed_tvd, line_color='red')
fig.add_annotation(text=f'<span style="color:red">Observed TVD = {round(observed_tvd, 2)}</span>',
                   x=2.3 * observed_tvd, showarrow=False, y=0.16)
fig.update_layout(yaxis_range=[0, 0.2])
fig.write_html('./assets/empirical-distribution-TVD-dependent.html', include_plotlyjs='cdn')

fig
```

```python
# calculate p-val
np.mean(np.array(tvds) >= observed_tvd)
```

```
0.0
```

P-val is less than our significance level of 0.05, so we reject the null hypothesis. Thet CAUSE.CATEGORY.DETAILS column is dependent on CAUSE.CATEGORY.

Testing Independent

Null Hypothesis: The distribution of CAUSE.CATEGORY.DETAIL is the same when CAUSE.CATEGORY name is missing and when it is not missing
Alt Hypothesis: The distribution of CAUSE.CATEGORY.DETAIL is different when CAUSE.CATEGORY name is missing as opposed to when it is not missing

```python
indep_var = 'YEAR'
q_var = 'CAUSE.CATEGORY.DETAIL'
```

```python
df_indep = (
    outage
    .assign(missing = outage[q_var].isna())
    .pivot_table(index=indep_var, columns='missing', aggfunc='size')
).fillna(0)


df_indep = df_indep / df_indep.sum()
df_indep.plot(
    kind='barh',
    title='Observed Distribution of Hurricane Names Conditional on Cause Category Detail',
    barmode='group',
    height=1100
    )
```

```python
df_indep
```

missing	False	True
YEAR		
2000	0.015992	0.019108
2001	0.001881	0.027601
2002	0.012230	0.008493
2003	0.036689	0.014862
2004	0.057385	0.021231
2005	0.043274	0.019108
2006	0.046096	0.038217
2007	0.042333	0.023355
2008	0.070555	0.076433
2009	0.048918	0.055202
2010	0.052681	0.106157
2011	0.145814	0.242038
2012	0.139229	0.055202
2013	0.111947	0.072187
2014	0.085607	0.044586
2015	0.052681	0.133758
2016	0.036689	0.042463

```python
observed_tvd = df_indep.diff(axis=1).iloc[:, -1].abs().sum() / 2
observed_tvd
```

```
0.277548419826913
```

Simulation

```python
n_repetitions = 500
shuffled = outage.copy()

tvds = []
for _ in range(n_repetitions):
    shuffled[indep_var] = np.random.permutation(shuffled[indep_var])

    # Computing and storing the TVD.
    pivoted = (
        shuffled
        .assign(missing = shuffled[q_var].isna())
        .pivot_table(index=indep_var, columns='missing', aggfunc='size')
        .apply(lambda x: x / x.sum())
    )

    tvd = pivoted.diff(axis=1).iloc[:, -1].abs().sum() / 2
    tvds.append(tvd)
```

```python
fig = px.histogram(pd.DataFrame(tvds), x=0, nbins=50, histnorm='probability',
                   title='Empirical Distribution of the TVD')
fig.add_vline(x=observed_tvd, line_color='red')
fig.add_annotation(text=f'<span style="color:red">Observed TVD = {round(observed_tvd, 2)}</span>',
                   x=2.3 * observed_tvd, showarrow=False, y=0.16)
fig.update_layout(yaxis_range=[0, 0.2])
```

```python
np.mean(np.array(tvds) >= observed_tvd)
```

```
0.0
```

The p-value is, again less than our significance level of 0.05. This means we reject the null hypothesis; making CAUSE.CATEGORY.DETAIL dependent on YEAR. I actually tried with most columns and could not find one that did not display dependency. Therefore I will move on to the next section.

Hypothesis Testing

Null: number of outages by state per capita comes from the same distribution as the proportion of outages that are caused by intentional attack by state
Alt: number of outages by state per capita and the proportion of outages that are caused by intentional attack by state come from different distributions
Significance level: 0.05

```python
# number of outages by state per capita
X = outage.groupby(by='U.S._STATE').count().OBS / outage.groupby(by='U.S._STATE').mean().POPULATION

# proportion of outages that are caused by intentional attack by state
cause_by_state = pd.pivot_table(outage, columns=['CAUSE.CATEGORY'], index=['U.S._STATE'], values='OBS', aggfunc='count').fillna(0)
Y = cause_by_state['severe weather'] / cause_by_state.sum(axis=1)

# put X and Y into a df
df = pd.DataFrame().assign(out_per_cap=X, prop_attack=Y)
# normalize the df
df = df / df.sum(axis=0)

# plot the distributions
fig = df.plot(kind='barh', height=1000, barmode='group', title='Observed Distributions of Outages by State Per Capita and Prop Outages Caused by Intentional Attack')
fig.write_html('./assets/hyp-test-observed.html', include_plotlyjs='cdn')
fig
```

```python
print(df.head().to_markdown())
```

```
| U.S._STATE   |   out_per_cap |   prop_attack |
|:-------------|--------------:|--------------:|
| Alabama      |    0.00414147 |    0.0367439  |
| Alaska       |    0.0051098  |    0          |
| Arizona      |    0.0142132  |    0.00629895 |
| Arkansas     |    0.0274285  |    0.0176371  |
| California   |    0.0181408  |    0.0146976  |
```

```python
# calculate observed tvd
observed_tvd = df.diff(axis=1).iloc[:, -1].abs().sum() / 2
observed_tvd
```

```
0.4082858650674824
```

```python
n_repetitions = 500
shuffled = outage.copy()

tvds = []
for _ in range(n_repetitions):
    shuffled['U.S._STATE'] = np.random.permutation(shuffled['U.S._STATE'])

    # number of outages by state per capita
    X = shuffled.groupby(by='U.S._STATE').count().OBS / shuffled.groupby(by='U.S._STATE').mean().POPULATION

    # proportion of outages that are caused by intentional attack by state
    cause_by_state = pd.pivot_table(shuffled, columns=['CAUSE.CATEGORY'], index=['U.S._STATE'], values='OBS', aggfunc='count').fillna(0)
    Y = cause_by_state['severe weather'] / cause_by_state.sum(axis=1)

    # put X and Y into a df
    df = pd.DataFrame().assign(out_per_cap=X, prop_attack=Y)
    # normalize the df
    df = df / df.sum(axis=0)

    tvd = df.diff(axis=1).iloc[:, -1].abs().sum() / 2
    tvds.append(tvd)
```

```python
# generate eimpirical distribution of the TVD
fig = px.histogram(pd.DataFrame(tvds), x=0, nbins=50, histnorm='probability',
                   title='Empirical Distribution of the TVD')
fig.add_vline(x=observed_tvd, line_color='red')
fig.add_annotation(text=f'<span style="color:red">Observed TVD = {round(observed_tvd, 2)}</span>',
                   x=2.3 * observed_tvd, showarrow=False, y=0.16)
fig.update_layout(yaxis_range=[0, 0.2])
fig.write_html('./assets/empirical-dist-tvd-hyp-test.html', include_plotlyjs='cdn')
fig
```

```python
# calculate p-value
np.mean(np.array(tvds) >= observed_tvd)
```

```
0.0
```

Our p-value 0.0 is less than our significance level of 0.05, so we reject the null hypothesis. We can conclude that the number of outages per capita and the proportion of outages caused by intentional attack in each state are likely drawn from different distributions. What does this tell us about out original question?

> Do states with a high number of outages per capita have higher instances of intentional attack?

While we can say nothing for certain, the permutation tests above indicate that there is likely a relationship between the number of outages per capita and seeing higher instances of intentional attack.

OBS	int64
YEAR	int64
MONTH	float64
U.S._STATE	object
POSTAL.CODE	object
NERC.REGION	object
CLIMATE.REGION	object
ANOMALY.LEVEL	float64

... | POPDEN_RURAL | float64 | | AREAPCT_URBAN | float64 | | AREAPCT_UC | float64 | | PCT_LAND | float64 | | PCT_WATER_TOT | float64 | | PCT_WATER_INLAND | float64 | | OUTAGE.START | datetime64[ns] | | OUTAGE.RESTORATION | datetime64[ns] |