Illuminating Cognizance: A Comprehensive Look Into Major Power Outages in the U.S.

Name(s): Phu Dang

Website Link: https://pndang.com/illuminating-cognizance/

Code

In [ ]:
```
# Importing packages and libraries

import pandas as pd
import numpy as np
import os
from scipy.stats import ks_2samp

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

In [ ]:
```
# Adjust dataframe display options to view full output
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)
# pd.set_option('display.max_rows', None)

# Reset to default, comment-out 2 (or 3) lines above and
    # uncomment 2 lines below, run all before pushing to GitHub
    # Reset to default helps limit the display scope of output dataframes,
    # easier to navigate notebook
# pd.set_option('display.max_columns', 20)
# pd.set_option('display.width', 80)
```

Introduction

Analysis questions of interest:

1. Are there any possible connections between outage duration and regional consumption information?
    - Look into different types of information: price vs. consumption vs. customers served
    - **(Personal favorite)** I plan to investigate this thoroughly

2. Is there a relation between anomaly level and cause category?

- If there are trends, are they statistically significant?
- Do extreme anomaly levels appear to be related to different cause categories than regular levels?
- **(New Personal favorite)** I plan to investigate this thoroughly

3. Do the start times of outages seem to affect outage duration?

   - Rationale: A sudden outage at night can give workers more time and space to fix overnight

4. What are some common characteristics of longer outages?

Cleaning and EDA

In [ ]:
```
# Importing dataset

path = os.path.join('data', 'outage.csv').replace('\\', '/')

df = pd.read_csv(path, header=5, skiprows=[6])
df = df.loc[:, ~df.columns.isin(['OBS', 'variables'])]
df.head()
```

Out[ ]:

	YEAR	MONTH	U.S._STATE	POSTAL.CODE	NERC.REGION	CLIMATE.REGION	ANOMALY.
0	2011	7.0	Minnesota	MN	MRO	East North Central	
1	2014	5.0	Minnesota	MN	MRO	East North Central	
2	2010	10.0	Minnesota	MN	MRO	East North Central	
3	2012	6.0	Minnesota	MN	MRO	East North Central	
4	2015	7.0	Minnesota	MN	MRO	East North Central	

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

Out[ ]:
```
(1534, 55)
```

In [ ]:
```
df.columns
```

Out[ ]:
```
Index(['YEAR', 'MONTH', 'U.S._STATE', 'POSTAL.CODE', 'NERC.REGION',
       'CLIMATE.REGION', 'ANOMALY.LEVEL', 'CLIMATE.CATEGORY',
       'OUTAGE.START.DATE', 'OUTAGE.START.TIME', 'OUTAGE.RESTORATION.DATE',
       'OUTAGE.RESTORATION.TIME', 'CAUSE.CATEGORY', 'CAUSE.CATEGORY.DETAIL',
       'HURRICANE.NAMES', 'OUTAGE.DURATION', 'DEMAND.LOSS.MW',
       'CUSTOMERS.AFFECTED', 'RES.PRICE', 'COM.PRICE', 'IND.PRICE',
       'TOTAL.PRICE', 'RES.SALES', 'COM.SALES', 'IND.SALES', 'TOTAL.SALES',
       'RES.PERCEN', 'COM.PERCEN', 'IND.PERCEN', 'RES.CUSTOMERS',
       'COM.CUSTOMERS', 'IND.CUSTOMERS', 'TOTAL.CUSTOMERS', 'RES.CUST.PCT',
       'COM.CUST.PCT', 'IND.CUST.PCT', 'PC.REALGSP.STATE', 'PC.REALGSP.USA',
       'PC.REALGSP.REL', 'PC.REALGSP.CHANGE', 'UTIL.REALGSP', 'TOTAL.REALGSP',
       'UTIL.CONTRI', 'PI.UTIL.OFUSA', 'POPULATION', 'POPPCT_URBAN',
       'POPPCT_UC', 'POPDEN_URBAN', 'POPDEN_UC', 'POPDEN_RURAL',
       'AREAPCT_URBAN', 'AREAPCT_UC', 'PCT_LAND', 'PCT_WATER_TOT',
       'PCT_WATER_INLAND'],
      dtype='object')
```

In [ ]:
```
df.head()
```

Out[ ]:

	YEAR	MONTH	U.S._STATE	POSTAL.CODE	NERC.REGION	CLIMATE.REGION	ANOMALY.
0	2011	7.0	Minnesota	MN	MRO	East North Central	
1	2014	5.0	Minnesota	MN	MRO	East North Central	
2	2010	10.0	Minnesota	MN	MRO	East North Central	
3	2012	6.0	Minnesota	MN	MRO	East North Central	
4	2015	7.0	Minnesota	MN	MRO	East North Central	

In [ ]:
```
df.dtypes
```

Out[ ]:
```
YEAR                 int64
MONTH              float64
U.S._STATE          object
POSTAL.CODE         object
NERC.REGION         object
                    ...
AREAPCT_URBAN      float64
AREAPCT_UC         float64
PCT_LAND           float64
PCT_WATER_TOT      float64
PCT_WATER_INLAND   float64
Length: 55, dtype: object
```

In [ ]:
```
# Combining OUTAGE.START.DATE and OUTAGE.START.TIME

dates_start = df['OUTAGE.START.DATE'] + ' ' + df['OUTAGE.START.TIME']
df['OUTAGE.START'] = pd.to_datetime(dates_start)
```

```
dates_end = df['OUTAGE.RESTORATION.DATE'] + ' ' + df['OUTAGE.RESTORATION.TIME']
df['OUTAGE.END'] = pd.to_datetime(dates_end)

df.drop(columns=['OUTAGE.START.DATE', \
    'OUTAGE.RESTORATION.DATE', 'OUTAGE.RESTORATION.TIME'], inplace=True)
```

In [ ]:
```
# Get counts of missing values in each column

df.isna().sum()
```

Out[ ]:
```
YEAR                 0
MONTH                9
U.S._STATE           0
POSTAL.CODE          0
NERC.REGION          0
                    ..
PCT_LAND             0
PCT_WATER_TOT        0
PCT_WATER_INLAND     0
OUTAGE.START         9
OUTAGE.END          58
Length: 54, dtype: int64
```

In [ ]:
```
df.dtypes
```

Out[ ]:
```
YEAR                         int64
MONTH                      float64
U.S._STATE                  object
POSTAL.CODE                 object
NERC.REGION                 object
                             ...
PCT_LAND                   float64
PCT_WATER_TOT              float64
PCT_WATER_INLAND           float64
OUTAGE.START        datetime64[ns]
OUTAGE.END          datetime64[ns]
Length: 54, dtype: object
```

In [ ]:
```
# Adding a duration column in hours

df['DURATION.HR'] = df['OUTAGE.DURATION'] / 60
```

In [ ]:
```
# Check min and max duration
print(df['DURATION.HR'].min())
print(df['DURATION.HR'].max())
```

```
0.0
1810.8833333333334
```

In [ ]:
```
df.head()
```

Out[ ]:

	YEAR	MONTH	U.S._STATE	POSTAL.CODE	NERC.REGION	CLIMATE.REGION	ANOMALY.
0	2011	7.0	Minnesota	MN	MRO	East North Central	
1	2014	5.0	Minnesota	MN	MRO	East North Central	
2	2010	10.0	Minnesota	MN	MRO	East North Central	
3	2012	6.0	Minnesota	MN	MRO	East North Central	
4	2015	7.0	Minnesota	MN	MRO	East North Central	

In [ ]:
```
# Convert dataframe to markdown

print(df[['U.S._STATE', 'POSTAL.CODE', 'ANOMALY.LEVEL', 'OUTAGE.START.TIME',
    'CAUSE.CATEGORY', 'CAUSE.CATEGORY.DETAIL', 'DEMAND.LOSS.MW', 'RES.PRICE',
    'PC.REALGSP.STATE', 'PCT_WATER_INLAND', 'DURATION.HR']].head().to_markdown(inde
```

```
| U.S._STATE    | POSTAL.CODE    |   ANOMALY.LEVEL | OUTAGE.START.TIME    | CAUSE.CATEG
ORY      | CAUSE.CATEGORY.DETAIL    |    DEMAND.LOSS.MW |    RES.PRICE |    PC.REALGSP.ST
ATE |    PCT_WATER_INLAND |    DURATION.HR |
|:-------------|:--------------|----------------:|:--------------------|:-----------
--------|:-----------------------|-----------------:|------------:|----------------
---:|-------------------:|--------------:|
| Minnesota    | MN            |            -0.3 | 5:00:00 PM           | severe weat
her      | nan                     |               nan |        11.6  |               51
268 |             5.47874 |    51            |
| Minnesota    | MN            |            -0.1 | 6:38:00 PM           | intentional
attack | vandalism                |               nan |        12.12 |              534
99 |             5.47874 |     0.0166667 |
| Minnesota    | MN            |            -1.5 | 8:00:00 PM           | severe weat
her      | heavy wind               |               nan |        10.87 |               50
447 |             5.47874 |    50            |
| Minnesota    | MN            |            -0.1 | 4:30:00 AM           | severe weat
her      | thunderstorm             |               nan |        11.79 |               51
598 |             5.47874 |    42.5          |
| Minnesota    | MN            |             1.2 | 2:00:00 AM           | severe weat
her      | nan                     |               250 |        13.07 |               54
431 |             5.47874 |    29            |
```

In [ ]:
```
# Change postal code column name to STATE.ABBR (state abbreviation)

df.rename(columns={'POSTAL.CODE': 'STATE.ABBR'}, inplace=True)
```

Univariate Analysis

In [ ]:
```
# Plotting the distribution of outage durations
```

```
fig1 = px.histogram(df, x='DURATION.HR', nbins=200, histnorm='probability', \
    title='Distribution of outage duration, in hours')
fig1.update_layout(xaxis_range=[0, 360])
fig1.update_xaxes(title_text='duration in hours')
fig1
```

In [ ]:
```
# Write to html file

path = os.path.join('assets', 'uni_1.html')
fig1.write_html(path, include_plotlyjs='cdn')
```

In [ ]:
```
# Sanity check

573 / len(df['DURATION.HR'])
```

Out[ ]: 0.37353324641460234

In [ ]:
```
# Plotting the distribution of cause category

fig2 = px.histogram(df, x='CAUSE.CATEGORY', histnorm='probability', \
    title='Distribution of outage cause category')
fig2.update_xaxes(title_text='cause category')
fig2
```

In [ ]:
```
# Write to html file

path = os.path.join('assets', 'uni_2.html')
fig2.write_html(path, include_plotlyjs='cdn')
```

Bivariate Analysis

In [ ]:
```
# Plotting a scatterplot between outage duration and residential electricity price

fig3 = px.scatter(df, x='RES.PRICE', y='DURATION.HR', title='Outage duration and re
fig3.update_xaxes(title_text='cents/kilowatt-hour')
fig3.update_yaxes(title_text='hours')
fig3
```

In [ ]:
```
# Write to html file

path = os.path.join('assets', 'bi_1.html')
fig3.write_html(path, include_plotlyjs='cdn')
```

In [ ]:
```
# Plotting a scatterplot between outage duration and customers served

fig4 = px.scatter(df, x='TOTAL.CUSTOMERS', y='DURATION.HR'\
    , title='Outage duration and total number of customers served, annually')
fig4.update_xaxes(title_text='number of customers served')
fig4.update_yaxes(title_text='hours')
fig4
```

In [ ]:
```
# Plotting a barplot between outage duration and regional economic output

fig5 = px.scatter(df, x='PC.REALGSP.STATE', y='DURATION.HR'\
    , title='Outage duration and per capita GSP')
fig5.update_xaxes(title_text='per capita real gross state product (measured in 2009
fig5.update_yaxes(title_text='hours')
fig5
```

In [ ]:
```
# Write to html file

path = os.path.join('assets', 'bi_2.html')
fig5.write_html(path, include_plotlyjs='cdn')
```

Interesting Aggregates

In [ ]:
```
# Plot average outage duration for every combination of state and cause category

group1 = pd.pivot_table(df, index=['U.S._STATE'], columns=['CAUSE.CATEGORY'], \
    values='DURATION.HR', aggfunc=np.mean)
group1['dummy'] = group1.sum(axis=1)
group1 = group1.sort_values(by='dummy', ascending=True).drop(columns=['dummy'])
fig6 = group1.plot(kind='barh', \
    title='Average outage duration by state and cause category',
    labels={'U.S._STATE': 'state ', 'value': 'average outage duration in hours ', \
        'CAUSE.CATEGORY': 'cause category '})
fig6.update_layout(height=1000, width=1500, legend=dict(title='cause category'))
fig6
```

In [ ]:
```
# Write to html file

path = os.path.join('assets', 'multi_1.html')
fig6.write_html(path, include_plotlyjs='cdn')
```

In [ ]:
```
# Sanity check

group1.sum(axis=1).sort_values(ascending=False).index
```

Out[ ]:
```
Index(['Michigan', 'Louisiana', 'Wisconsin', 'New York', 'Arizona', 'Indiana',
       'Texas', 'Kentucky', 'California', 'Florida', 'Kansas', 'West Virginia',
       'Iowa', 'Washington', 'New Jersey', 'Pennsylvania', 'Ohio', 'Illinois',
       'Oklahoma', 'Missouri', 'District of Columbia', 'Massachusetts',
       'Tennessee', 'Maryland', 'Arkansas', 'Maine', 'Minnesota', 'Utah',
       'Nebraska', 'South Carolina', 'Colorado', 'Oregon', 'North Carolina',
       'Connecticut', 'Delaware', 'Virginia', 'Idaho', 'New Hampshire',
       'Georgia', 'Alabama', 'Hawaii', 'North Dakota', 'Nevada', 'Mississippi',
       'New Mexico', 'Wyoming', 'Montana', 'South Dakota', 'Vermont'],
      dtype='object', name='U.S._STATE')
```

In [ ]:
```
group1
```

Out[ ]:

CAUSE.CATEGORY	equipment failure	fuel supply emergency	intentional attack	islanding	public appeal	severe weather	o d
U.S._STATE							
Vermont	NaN	NaN	0.590741	NaN	NaN	NaN	
South Dakota	NaN	NaN	NaN	2.000000	NaN	NaN	
Montana	NaN	NaN	1.550000	0.575000	NaN	NaN	
Wyoming	1.016667	NaN	0.005556	0.533333	NaN	1.766667	
New Mexico	NaN	1.266667	2.908333	NaN	NaN	NaN	
...	...	...	...	...	...	...	
Arizona	2.308333	NaN	10.660000	NaN	NaN	428.775000	
New York	4.116667	278.120833	5.151389	NaN	44.250000	100.576263	
Wisconsin	NaN	566.187500	7.650000	NaN	6.466667	25.457143	
Louisiana	2.938889	469.500000	NaN	NaN	22.653571	119.782143	
Michigan	440.588889	NaN	60.587500	0.016667	17.966667	80.527510	

49 rows × 7 columns

In [ ]:

```
# Get aggregated mean outage duration in minutes, hours, and customers affected by

# pd.set_option('display.max_rows', None)
group2 = df.groupby(by=['STATE.ABBR', 'CAUSE.CATEGORY']).mean()[['OUTAGE.DURATION',
group2.rename(columns={'OUTAGE.DURATION': 'Avg duration (mins)', \
    'DURATION.HR': 'Avg duration (hrs)',
    'CUSTOMERS.AFFECTED': 'Avg # of customers affected'})
pd.set_option('display.max_rows', 20)
group2.head()
```

Out[ ]:

		OUTAGE.DURATION	DURATION.HR	CUSTOMERS.AFFECTED
STATE.ABBR	**CAUSE.CATEGORY**			
AK	**equipment failure**	NaN	NaN	14273.0
AL	**intentional attack**	77.000000	1.283333	NaN
	severe weather	1421.750000	23.695833	94328.8
AR	**equipment failure**	105.000000	1.750000	NaN
	intentional attack	547.833333	9.130556	9200.0

In [ ]:
```
# sanity check

df[df['STATE.ABBR'] == 'AK']
```

Out[ ]:

	YEAR	MONTH	U.S._STATE	STATE.ABBR	NERC.REGION	CLIMATE.REGION	ANOMAL
1533	2000	NaN	Alaska	AK	ASCC	NaN	

In [ ]:
```
print(group1[-10:-5].to_markdown(index=True))
```

```
| U.S._STATE   |   equipment failure |   fuel supply emergency |   intentional attac
k |   islanding |   public appeal |   severe weather |   system operability disrupti
on |
|:-------------|--------------------:|------------------------:|--------------------
-:|------------:|----------------:|-----------------:|------------------------------
--:|
| Florida      |           9.24167   |                     nan |                0.83333
3 |   nan       |         72        |          107.003 |                            3.428
33 |
| California   |           8.74683   |                 102.577 |               15.7743
|     3.58095 |         33.8019 |          48.8062 |                            6.06111
|
| Kentucky     |          10.8667    |                 209.5   |                1.8
|   nan       |        nan       |          74.6685 |                          nan
|
| Texas        |           6.76      |                 232     |                4.97949
|   nan       |         19.0069 |          64.2482 |                           13.5133
|
| Indiana      |           0.0166667 |                 204     |                7.03125
|     2.08889 |        nan       |          75.3882 |                           77.86
|
```

In [ ]:
```
# Get pivot table of average anomaly level by cause category and category detail

group2 = pd.pivot_table(df, index=['CAUSE.CATEGORY'], \
    columns=['CAUSE.CATEGORY.DETAIL'], values='ANOMALY.LEVEL', aggfunc=np.mean)
fig7 = group2.plot(kind='barh', \
    title='Average anomaly level by cause category, subset by cause '+
    'detail<br><sup>Anomaly level represents the oceanic El Niño/La Niña (ONI)'+
    ' index, estimated as a 3-month running mean of ERSST.v4 SST anomalies in'+
    ' the Niño 3.4 region</sup>',
    labels={'CAUSE.CATEGORY': 'cause category', 'value': 'average anomaly level (ON
       'CAUSE.CATEGORY.DETAIL': 'cause detail'})
fig7.update_layout(legend=dict(title='cause detail'))
fig7
```

In [ ]:
```
# Write to html file

path = os.path.join('assets', 'oni_by_cause.html')
fig7.write_html(path, include_plotlyjs='cdn')
```

In [ ]:
```
group2
```

Out[ ]:

CAUSE.CATEGORY.DETAIL	Coal	Hydro	Natural Gas	100 MW loadshed	Coal	HVSubstation interruption	Hyd
CAUSE.CATEGORY							
equipment failure	NaN	NaN	NaN	NaN	NaN	NaN	Na
fuel supply emergency	-0.29	-0.4	-0.514286	NaN	-0.471429	NaN	-0.
intentional attack	NaN	NaN	NaN	NaN	NaN	NaN	Na
severe weather	NaN	NaN	NaN	NaN	NaN	NaN	Na
system operability disruption	NaN	NaN	NaN	-0.2	NaN	-0.1	Na

In [ ]:

```
# Write to html file

path = os.path.join('assets', 'multi_2.html')
fig7.write_html(path, include_plotlyjs='cdn')
```

In [ ]:

```
df.head()
```

Out[ ]:

	YEAR	MONTH	U.S._STATE	STATE.ABBR	NERC.REGION	CLIMATE.REGION	ANOMALY.LE
0	2011	7.0	Minnesota	MN	MRO	East North Central	
1	2014	5.0	Minnesota	MN	MRO	East North Central	
2	2010	10.0	Minnesota	MN	MRO	East North Central	
3	2012	6.0	Minnesota	MN	MRO	East North Central	
4	2015	7.0	Minnesota	MN	MRO	East North Central	

Assessment of Missingness

In [ ]:

```
# Inspect the outages where month is missing (likely MAR, explained by year)

df[df['MONTH'].isna()].head(11)
```

Out[ ]:

	YEAR	MONTH	U.S._STATE	STATE.ABBR	NERC.REGION	CLIMATE.REGION	ANOMAL
239	2000	NaN	Texas	TX	FRCC	South	
339	2000	NaN	Alabama	AL	SERC	Southeast	
365	2000	NaN	Illinois	IL	SERC	Central	
766	2000	NaN	North Carolina	NC	SERC	Southeast	
887	2000	NaN	Delaware	DE	RFC	Northeast	
1318	2000	NaN	Virginia	VA	SERC	Southeast	
1506	2002	NaN	Kansas	KS	SPP	South	
1530	2006	NaN	North Dakota	ND	MRO	West North Central	
1533	2000	NaN	Alaska	AK	ASCC	NaN	

In [ ]:

```
# Inspect the outages where peak-hours demand loss is missing (demand loss is likel

df[df['DEMAND.LOSS.MW'].isna()].head(7)
```

Out[ ]:

	YEAR	MONTH	U.S._STATE	STATE.ABBR	NERC.REGION	CLIMATE.REGION	ANOMALY.LE
0	2011	7.0	Minnesota	MN	MRO	East North Central	
1	2014	5.0	Minnesota	MN	MRO	East North Central	
2	2010	10.0	Minnesota	MN	MRO	East North Central	
3	2012	6.0	Minnesota	MN	MRO	East North Central	
5	2010	11.0	Minnesota	MN	MRO	East North Central	
6	2010	7.0	Minnesota	MN	MRO	East North Central	
9	2013	6.0	Minnesota	MN	MRO	East North Central	

Missingness Assessment Analysis notes:

DEMAND.LOSS.MW could possibly **depend** on outage start time and/or outage duration

- Rationale: A short outage outside of high-demand times (4 PM - 9 PM) will not have data for DEMAND.LOSS.MW

DEMAND.LOSS.MW could also **depend** on the number of customers affected (anticipating a positive correlation)

DEMAND.LOSS.MW is likely to **not depend** on PCT_WATER_INLAND

- PCT_WATER_INLAND ~ percentage of inland water area in the U.S. state as compared to the overall inland water area in the continental U.S. (in %)

In [ ]:
```
# Query out relevant columns for assessing missingness to keep dataframe simple

aom = df[['DEMAND.LOSS.MW', 'OUTAGE.START.TIME', 'DURATION.HR', \
    'CUSTOMERS.AFFECTED', 'PCT_WATER_INLAND']]
print(aom.shape[0])
aom.head()
```

```
1534
```

Out[ ]:

	DEMAND.LOSS.MW	OUTAGE.START.TIME	DURATION.HR	CUSTOMERS.AFFECTED	PCT_W
0	NaN	5:00:00 PM	51.000000	70000.0	
1	NaN	6:38:00 PM	0.016667	NaN	
2	NaN	8:00:00 PM	50.000000	70000.0	
3	NaN	4:30:00 AM	42.500000	68200.0	
4	250.0	2:00:00 AM	29.000000	250000.0	

In [ ]:
```
# Inspect the missingness of columns in aom

aom.isna().sum()
```

Out[ ]:
```
DEMAND.LOSS.MW        705
OUTAGE.START.TIME       9
DURATION.HR            58
CUSTOMERS.AFFECTED    443
PCT_WATER_INLAND        0
dtype: int64
```

Permutation test wrt outage duration

- Assess missingness of DEMAND.LOSS.MW

In [ ]:
```
# Assign a boolean column DEMAND.MISSING that indicates whether, or not, peak
# demand lost is missing
```

```
aom['DEMAND.MISSING'] = aom['DEMAND.LOSS.MW'].isna()
aom.head(3)
```

```
C:\Users\phuro\AppData\Local\Temp\ipykernel_3788\1607562000.py:4: SettingWithCopyWar
ning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
```

Out[ ]:

	DEMAND.LOSS.MW	OUTAGE.START.TIME	DURATION.HR	CUSTOMERS.AFFECTED	PCT_W
0	NaN	5:00:00 PM	51.000000	70000.0	
1	NaN	6:38:00 PM	0.016667	NaN	
2	NaN	8:00:00 PM	50.000000	70000.0	

In [ ]:

```
# Box plot of duration hours by missingness of peak-hours demand loss

aom_fig1 = px.box(aom, x='DURATION.HR', color='DEMAND.MISSING')
aom_fig1.update_layout(xaxis_range=[0, 500])
aom_fig1
```

In [ ]:

```
# Histogram of duration hours by missingness of peak-hours demand loss

aom_fig2 = px.histogram(aom, x='DURATION.HR', color='DEMAND.MISSING', \
    histnorm='probability', nbins=3000, marginal='rug')
aom_fig2.update_layout(xaxis_range=[0, 24])
aom_fig2
```

In [ ]:

```
aom.dtypes
```

Out[ ]:

```
DEMAND.LOSS.MW        float64
OUTAGE.START.TIME      object
DURATION.HR           float64
CUSTOMERS.AFFECTED    float64
PCT_WATER_INLAND      float64
DEMAND.MISSING           bool
dtype: object
```

In [ ]:

```
# Convert outage start times to datetime data type

times = pd.to_datetime(aom['OUTAGE.START.TIME'])
aom['START.TIME'] = times
aom = aom.sort_values(by='START.TIME', ascending=True)
aom.head(5)
```

```
C:\Users\phuro\AppData\Local\Temp\ipykernel_3788\3097801354.py:4: SettingWithCopyWar
ning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
```

Out[ ]:

	DEMAND.LOSS.MW	OUTAGE.START.TIME	DURATION.HR	CUSTOMERS.AFFECTED	PC
557	0.0	12:00:00 AM	0.016667	0.0	
545	2000.0	12:00:00 AM	214.833333	650000.0	
544	0.0	12:00:00 AM	0.033333	0.0	
1374	10.0	12:00:00 AM	13.500000	47165.0	
542	0.0	12:00:00 AM	24.000000	0.0	

In [ ]:
```
# Histogram of outage start time by missingness of peak-hours demand lost

mean = aom['START.TIME'].mean()
median = aom['START.TIME'].median()

aom_fig3 = px.histogram(aom, x='START.TIME', color='DEMAND.MISSING', \
    histnorm='probability',
    title='Outage start time by missingness of peak demand lost',
    labels={'START.TIME': 'outage start time', 'DEMAND.MISSING': 'demand lost missi
    barmode='overlay', marginal='box', opacity=0.7)

aom_fig3.add_vline(x=median, line_color='#00CC96')
aom_fig3.add_vline(x=mean, line_color='#AB63FA')

aom_fig3.update_xaxes(
    ticktext=["00:00", "03:00", "06:00", "09:00", "12:00", "14:30", "18:00", \
            "21:00<br><b>|<b></br>", "00:00", "16:00<br><b>|<b></br>",
            "<br>peak demand window</br>", "|<br>mean</br>",
            "|<br>|<br>median</br></br>"],
    tickvals=[pd.to_datetime('00:00:00'), pd.to_datetime('03:00:00'), \
            pd.to_datetime('06:00:00'), pd.to_datetime('09:00:00'), \
            pd.to_datetime('12:00:00'), pd.to_datetime('14:30:00'),
            pd.to_datetime('18:00:00'), pd.to_datetime('21:00:00'),
            pd.to_datetime('23:59:00'), pd.to_datetime('16:00:00'),
            pd.to_datetime('18:30:00'), mean, median],
    tickangle=0
)
aom_fig3
```

In [ ]:
```
# Write to html file
```

```
path = os.path.join('assets', 'aom_perm1_observed_dist.html')
aom_fig3.write_html(path, include_plotlyjs='cdn')
```

In [ ]:
```
aom.head(3)
```

Out[ ]:

	DEMAND.LOSS.MW	OUTAGE.START.TIME	DURATION.HR	CUSTOMERS.AFFECTED	PCT
557	0.0	12:00:00 AM	0.016667	0.0	
545	2000.0	12:00:00 AM	214.833333	650000.0	
544	0.0	12:00:00 AM	0.033333	0.0	

Null Hypothesis: The start times between outages where the amount of peak demand lost **is** missing, and outages where the amount of peak demand lost is **not** missing, have **the same** distribution. Any observed difference is due to chance alone.

Alternative Hypothesis: The outage start times by missingness of peak demand lost have **different** distributions. The observed difference is **unlikely** due to chance alone.

Test statistic: difference in group median start time

Significance level: 5%

Method: shuffle DEMAND.MISSING (status of missing) column to simulate under null hypothesis

In [ ]:
```
# Run simulation 5000 times

N = 5000
differences = []

for _ in range(N):
    aom['shuffled'] = np.random.permutation(aom['DEMAND.MISSING'])
    w_missing_median = aom[aom['shuffled']]['START.TIME'].median()
    wo_missing_median = aom[~aom['shuffled']]['START.TIME'].median()
    test_stat = abs(w_missing_median-wo_missing_median)
    differences.append(test_stat)
```

In [ ]:
```
# Get observed test statistic

w_missing_median_obs = aom[aom['DEMAND.MISSING']]['START.TIME'].median()
wo_missing_median_obs = aom[~aom['DEMAND.MISSING']]['START.TIME'].median()
observed = abs(w_missing_median_obs-wo_missing_median_obs)
observed = observed.total_seconds()
```

In [ ]:
```
# Get p-value

results = []
differences_secs = []
```

```
for val in differences:
    val = val.total_seconds()
    differences_secs.append(val)
    if val >= observed:
        results.append(True)
        continue
    results.append(False)

pval = np.mean(results)
pval
```

Out[ ]: 0.1796

In [ ]:
```
# Plot distribution of results

perm1 = px.histogram(pd.DataFrame({'simulated differences': differences_secs}),\
    x='simulated differences', histnorm='probability',
    title='Simulated differences of median outage start times',
        # <br><sup>Calculation: absolute difference between median outage start tim
    labels={'simulated differences': 'simulated differences (seconds)'})
perm1.add_vline(x=observed, line_color='rgb(0,100,80)',
            annotation_text='observed', annotation_position='top left')
p_95 = np.percentile(differences_secs, 95)
perm1.add_vline(x=p_95, line_color='rgb(0,176,246)',
            annotation_text='significance level (5%)', annotation_position='top rig
perm1
```

In [ ]:
```
# Write to html file

path = os.path.join('assets', 'aom_perm1_results.html')
perm1.write_html(path, include_plotlyjs='cdn')
```

Result: **Fail to reject** the null hypothesis at a 5% significance level

- Missingness of peak demand lost is likely to not depend on outage start times

In [ ]:
```
aom.head(3)
```

Out[ ]:

	DEMAND.LOSS.MW	OUTAGE.START.TIME	DURATION.HR	CUSTOMERS.AFFECTED	PCT_
557	0.0	12:00:00 AM	0.016667	0.0	
545	2000.0	12:00:00 AM	214.833333	650000.0	
544	0.0	12:00:00 AM	0.033333	0.0	

PCT_WATER_INLAND

- Percentage of inland water area in the U.S. state as compared to the overall inland water area in the continental U.S. (in %)

In [ ]:
```
# Histogram of state water percentage by missingness of peak-hours demand loss

aom_fig4 = px.histogram(aom, x='PCT_WATER_INLAND', color='DEMAND.MISSING', \
    histnorm='probability', barmode='overlay', marginal='box', opacity=0.7,
    nbins=20, title="State water percentage by missingness of peak demand lost",
    labels={'PCT_WATER_INLAND': 'state water percentage',
    'DEMAND.MISSING': 'demand lost missing'}
)

aom_fig4.add_vline(x=aom['PCT_WATER_INLAND'].mean(), annotation_text='mean', \
    line_color='#AB63FA')
aom_fig4.add_vline(x=aom['PCT_WATER_INLAND'].median(), annotation_text='median', \
    line_color='#00CC96')
aom_fig4.update_layout(yaxis_range=[0, 0.37])
aom_fig4
```

In [ ]:
```
# Write to html file

path = os.path.join('assets', 'aom_perm2_observed_dist.html')
aom_fig4.write_html(path, include_plotlyjs='cdn')
```

In [ ]:
```
# Plot the CDFs of state water percentage by missingness of peak demand lost

def create_cdf(df, group_col, group, stat_col):
    return df.loc[df[group_col] == group, stat_col].value_counts(normalize=True).so

cdf_fig = go.Figure()

cdf_fig.add_trace(
    go.Scatter(x=create_cdf(aom, 'DEMAND.MISSING', True, 'PCT_WATER_INLAND').index,
        y=create_cdf(aom, 'DEMAND.MISSING', True, 'PCT_WATER_INLAND'),
        name='CDF with missing demand')
)

cdf_fig.add_trace(
    go.Scatter(x=create_cdf(aom, 'DEMAND.MISSING', False, 'PCT_WATER_INLAND').index
        y=create_cdf(aom, 'DEMAND.MISSING', False, 'PCT_WATER_INLAND'),
        name='CDF without missing demand')
)

cdf_fig.update_layout(title='CDFs of state water percentage by missingness of peak
cdf_fig.update_xaxes(title='state water percentage')
cdf_fig.update_yaxes(title='cumulative probability')
```

In [ ]:
```
# Write to html file

path = os.path.join('assets', 'aom_perm2_cdf.html')
cdf_fig.write_html(path, include_plotlyjs='cdn')
```

Null Hypothesis: The distribution of state inland water percentage in outages where peak demand lost **is** missing is **the same** as in outages where peak demand lost **is not** missing. Any observed difference is due to chance alone.

Alternative Hypothesis: The distributions of state inland water percentage between the two groups are different. The observed difference is **unlikely** due to chance alone.

Test statistic: K-S statistic

Significance level: 5%

Method: shuffle DEMAND.MISSING (status of missing) column to simulate under null hypothesis

In [ ]:
```
# Run simulation 5000 times

N = 10000
results = []

for _ in range(N):
    aom['shuffled'] = np.random.permutation(aom['DEMAND.MISSING'])
    groups = aom.groupby('shuffled')['PCT_WATER_INLAND']
    ks_stat = ks_2samp(groups.get_group(True), groups.get_group(False)).statistic
    results.append(ks_stat)

results[:5]
```

Out[ ]:
```
[0.02988305144196631,
 0.038723917562816006,
 0.046476571790330996,
 0.050458126940943974,
 0.03833038181522641]
```

In [ ]:
```
# Get observed K-S statistic

observed_ks = ks_2samp(aom.loc[aom['DEMAND.MISSING'], 'PCT_WATER_INLAND'], \
    aom.loc[~aom['DEMAND.MISSING'], 'PCT_WATER_INLAND']).statistic
observed_ks
```

Out[ ]:
```
0.08352539588840695
```

In [ ]:
```
# Get p-value

pval = (np.array(results) >= observed_ks).mean()
pval
```

Out[ ]:
```
0.0036
```

In [ ]:
```
# Plot simulated K-S stats

perm2 = px.histogram(pd.DataFrame(data={'simulated K-S statistics': results}), \
    x='simulated K-S statistics', histnorm='probability',
    title='Empirical distribution of the K-S Statistic, state water percentage by m
perm2.add_vline(x=observed_ks, line_color='rgb(0,100,80)', \
    annotation_text='observed K-S')
p_95 = np.percentile(results, 95)
perm2.add_vline(x=p_95, line_color='rgb(0,176,246)', \
```

```
    annotation_text='significance level (5%)', annotation_position='top left')
perm2
```

In [ ]:
```
# Write to html file

path = os.path.join('assets', 'aom_perm2_results.html')
perm2.write_html(path, include_plotlyjs='cdn')
```

Result: **Reject** the null hypothesis at a 5% significance level

- Missingness of peak demand lost **could possibly** depend on the state proportion of inland water relative to continental U.S. (MAR dependent)

In [ ]:
```
# sanity check

aom['PCT_WATER_INLAND'].min()
```

Out[ ]: 0.240151328

Hypothesis Testing

Test the observed distribution of anomaly level by cause category

In [ ]:
```
df['ANOMALY.LEVEL'].plot()
```

In [ ]:
```
fig7
```

In [ ]:
```
group2
```

Out[ ]:

CAUSE.CATEGORY.DETAIL	Coal	Hydro	Natural Gas	100 MW loadshed	Coal	HVSubstation interruption	Hyd
CAUSE.CATEGORY							
equipment failure	NaN	NaN	NaN	NaN	NaN	NaN	Na
fuel supply emergency	-0.29	-0.4	-0.514286	NaN	-0.471429	NaN	-0.
intentional attack	NaN	NaN	NaN	NaN	NaN	NaN	Na
severe weather	NaN	NaN	NaN	NaN	NaN	NaN	Na
system operability disruption	NaN	NaN	NaN	-0.2	NaN	-0.1	Na

In [ ]:
```
group2.abs().sum(axis=1).plot(kind='bar')
```

In [ ]:
```
# Inspect the distribution of anomaly levels (observed)
```

```
oni_fig = df[['ANOMALY.LEVEL', 'CAUSE.CATEGORY']].plot(kind='hist', \
    x='ANOMALY.LEVEL', histnorm='probability',
    labels={'ANOMALY.LEVEL': 'anomaly level (ONI Index)', 'CAUSE.CATEGORY':
    'cause category'},
    title='Distribution of anomaly level (ONI Index), subset by cause category',
    color='CAUSE.CATEGORY')
oni_fig.add_vline(x=df['ANOMALY.LEVEL'].mean(), annotation_text='mean')
oni_fig.add_vline(x=df['ANOMALY.LEVEL'].median(), annotation_text='median', \
    annotation_position='top left')
oni_fig.update_layout(yaxis_range=[0, 0.9])
```

In [ ]:
```
# Write to html file

path = os.path.join('assets', 'hyp1_oni_dist.html')
oni_fig.write_html(path, include_plotlyjs='cdn')
```

In [ ]:
```
# Inspect the distribution of anomaly levels (observed) separately by cause categor

for cat in group2.index:
    data = df[df['CAUSE.CATEGORY'] == cat]
    plot = data['ANOMALY.LEVEL'].plot(kind='hist', histnorm='probability', labels={
        'value': 'anomaly level (ONI Index)'},
        title=f'Distribution of anomaly levels for {cat}')
    plot.add_vline(x=data['ANOMALY.LEVEL'].mean(), annotation_text='mean')
    plot.add_vline(x=data['ANOMALY.LEVEL'].median(), annotation_text='median', \
        annotation_position='top left')
    plot.show()

    # Write to html file
    path = os.path.join('assets', f'hyp1_oni_obs_dist_{cat}.html')
    plot.write_html(path, include_plotlyjs='cdn')
```

Conclusion from above: A non-parametric test (permutation) seems appropriate because the observed distribution of anomaly level is non-normal

Test begin

Null Hypothesis: The distribution of outage cause categories with **extreme** anomaly levels is **the same** as in outage cause categories with **regular** anomaly levels. Any observed difference is due to chance alone.

Alternative Hypothesis: The distributions of outage cause categories between the two groups are different. The observed difference is **unlikely** due to chance alone.

Test statistic: TVD

Significance level: 5%

Method: shuffle the binary column that indicates whether the associated anomaly level is extreme to simulate under null hypothesis

- Anomaly levels above 0.5 or below -0.5 are considered extreme and are indicative of El Nino and La Nina, respectively

In [ ]:
```
def get_tvd(df, groups, cats, show_steps=False):

    '''
        Function to calculate the Total Variation Distance between groups

        groups: the binary column
        cats: the categorical column
    '''

    # Get count of each pair of catogory and group
    counts = df.pivot_table(index=cats, columns=groups, aggfunc='size')

    # Normalize each column
    distr = counts / counts.sum()

    if show_steps:
        print("\nSTEPS:")
        print(counts)
        print()
        print(distr)
        print()
        print(distr.diff(axis=1))
        print()
        print(distr.diff(axis=1).iloc[:, -1])
        print()
        print(distr.diff(axis=1).iloc[:, -1].abs())
        print()
        print(distr.diff(axis=1).iloc[:, -1].abs().sum() / 2)
        print("END\n")

    # Compute and return the total variation distance
    return distr.diff(axis=1).iloc[:, -1].abs().sum() / 2
```

In [ ]:
```
# Assign an "extreme" column that indicates whether, or not, the associated
# anomaly level is extreme

oni = df[['ANOMALY.LEVEL', 'CAUSE.CATEGORY']]
oni['extreme'] = (oni['ANOMALY.LEVEL'] < -0.5) | (oni['ANOMALY.LEVEL'] > 0.5)
oni.head(10)
```

```
C:\Users\phuro\AppData\Local\Temp\ipykernel_3788\45707634.py:5: SettingWithCopyWarni
ng:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
```

Out[ ]:

	ANOMALY.LEVEL	CAUSE.CATEGORY	extreme
0	-0.3	severe weather	False
1	-0.1	intentional attack	False
2	-1.5	severe weather	True
3	-0.1	severe weather	False
4	1.2	severe weather	True
5	-1.4	severe weather	True
6	-0.9	severe weather	True
7	0.2	severe weather	False
8	0.6	intentional attack	True
9	-0.2	severe weather	False

In [ ]:
```
# Get observed test statistic

observed_tvd = get_tvd(oni, groups='extreme', cats='CAUSE.CATEGORY')
observed_tvd
```

Out[ ]:
```
0.09399855386840203
```

In [ ]:
```
# Run simulation 5000

N = 5000
results = []

for _ in range(N):
    oni['shuffled'] = np.random.permutation(oni['extreme'])
    tvd = get_tvd(oni, groups='shuffled', cats='CAUSE.CATEGORY')
    results.append(tvd)

results[:5]
```

```
C:\Users\phuro\AppData\Local\Temp\ipykernel_3788\3431769426.py:7: SettingWithCopyWar
ning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
```

Out[ ]:
```
[0.02090156380701231,
 0.04463513532672648,
 0.039967107595026426,
 0.030191541547927924,
 0.028766676591099202]
```

In [ ]:
```
# Get p-value

pval = np.mean(np.array(results) >= observed_tvd)
pval
```

Out[ ]: 0.0014

In [ ]:
```
# Plot observed tvd and simulated tvds

hyp1 = px.histogram(pd.DataFrame({'simulated tvds': results}), \
    x='simulated tvds', histnorm='probability',
    title='Empirical Distribution of TVD')
hyp1.add_vline(x=observed_tvd, line_color='rgb(0,100,80)',
              annotation_text='observed', annotation_position='top right')
p_95 = np.percentile(results, 95)
hyp1.add_vrect(x0=p_95, x1=np.max(results), line_color='rgb(0,176,246)',
              annotation_text='significance level', annotation_position='top left')
hyp1.update_layout(xaxis_range=[0, 0.11])
```

In [ ]:
```
# Write to html file

path = os.path.join('assets', 'hyp1_results.html')
hyp1.write_html(path, include_plotlyjs='cdn')
```

Conclusion: Reject the null hypothesis at a 5% significance level.

The distribution of cause categories in outages with extreme anomaly levels are statistically significantly **different** than outages with regular anomaly levels