Exploratory Data Analysis of Major Outages

Name(s): Diego Silva

Website Link: https://dsilva019.github.io/EDA-of-Major-Outages/

Code

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

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

import plotly.figure_factory as ff
import plotly.graph_objects as go

from scipy.stats import ks_2samp

# Used for plotting examples.
def create_kde_plotly(df, group_col, group1, group2, vals_col, title=''):
    fig = ff.create_distplot(
        hist_data=[df.loc[df[group_col] == group1, vals_col], df.loc[df[group_col]
        group_labels=[group1, group2],
        show_rug=False, show_hist=False,
        colors=['#ef553b', '#636efb'],
    )
    return fig.update_layout(
                            xaxis_title= "",
    yaxis=dict(showgrid=False, tickfont = dict(size=18)),
    xaxis = dict(showgrid=False, tickfont = dict(size=10.5)),
    font=dict(family="Lato",color="black", size = 20),
    plot_bgcolor='rgba(0,0,0,0)',
    title={
        'text': title,
        'y':0.9,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'}
    )
```

Code Cited:

create_kde_ploty function is from Lecture 12 – Identifying Missingness Mechanisms, with a sligthly modified output.

Introduction:

In this project, I cleaned and analyzed a data set containing major outages reported by different states in the United States from January 2000-July 2016. The main question I want to answer in this analysis, is there a significant difference between the outages distributions of the seasons in the SPP Region and the Overall outages distributions of the seasons of the NERC Regions? This data set and analysis provide an understanding of major outage patterns and how in the future they can be avoided to improve our national electrical infrastructure. Moving forward I will reference the data set as Outages.

Original Data Set

In [2]:
```
outages_fp = os.path.join('data', 'outage.xlsx')
outages = pd.read_excel(outages_fp)
outages
```

Out[2]:

	Major power outage events in the continental U.S.	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	
0	Time period: January 2000 - July 2016	NaN	NaN	NaN	NaN	NaN	NaN	
1	Regions affected: Outages reported in this dat...	NaN	NaN	NaN	NaN	NaN	NaN	
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
4	variables	OBS	YEAR	MONTH	U.S._STATE	POSTAL.CODE	NERC.REGION	CLIM
...	...	...	...	...	...	...	...	
1535	NaN	1530	2011	12	North Dakota	ND	MRO	
1536	NaN	1531	2006	NaN	North Dakota	ND	MRO	
1537	NaN	1532	2009	8	South Dakota	SD	RFC	
1538	NaN	1533	2009	8	South Dakota	SD	MRO	
1539	NaN	1534	2000	NaN	Alaska	AK	ASCC	

1540 rows × 57 columns

Cleaning and EDA

In [3]:

```
#This renames the columns to the proper column names
outages.columns = list(outages.iloc[4])

# This drops the 2 unecessary columns in the df
outages = outages.drop(outages.columns[[0, 1]], axis = 1)

# This drops the 5 unecessary rows in the df
outages = outages.drop([0, 1, 2, 3, 4, 5], axis = 0)
# This converts the object types of the columns to best possible types
outages = outages.infer_objects()
```

```
#This resets the index
outages.reset_index(drop=True, inplace=True)
outages
```

Out[3]:

	YEAR	MONTH	U.S._STATE	POSTAL.CODE	NERC.REGION	CLIMATE.REGION	ANOMALY.LEVE
0	2011	7.0	Minnesota	MN	MRO	East North Central	-0.
1	2014	5.0	Minnesota	MN	MRO	East North Central	-0.
2	2010	10.0	Minnesota	MN	MRO	East North Central	-1.
3	2012	6.0	Minnesota	MN	MRO	East North Central	-0.
4	2015	7.0	Minnesota	MN	MRO	East North Central	1.
...	...	...	...	...	...	...	
1529	2011	12.0	North Dakota	ND	MRO	West North Central	-0.
1530	2006	NaN	North Dakota	ND	MRO	West North Central	Na
1531	2009	8.0	South Dakota	SD	RFC	West North Central	0.
1532	2009	8.0	South Dakota	SD	MRO	West North Central	0.
1533	2000	NaN	Alaska	AK	ASCC	NaN	Na

1534 rows × 55 columns

In [4]:

```
# TODO
#OUTAGE.Start Column Creation
#This converts the colummns to string types then they get converted to datetime and
outages['OUTAGE.START.DATE'] = outages['OUTAGE.START.DATE'].map(str)
outages['OUTAGE.START.TIME'] = outages['OUTAGE.START.TIME'].map(str)
outages['OUTAGE.START.DATE'] = pd.to_datetime(outages['OUTAGE.START.DATE'])
outages['OUTAGE.START.TIME'] = pd.to_timedelta(outages['OUTAGE.START.TIME'])

#This combines the two columns inoformation into one column
outages['OUTAGE.START'] = outages['OUTAGE.START.DATE'] + outages['OUTAGE.START.TIME

#OUTAGE.RESTORATION Column Creation
#This converts the colummns to string types then they get converted to datetime and
outages['OUTAGE.RESTORATION.DATE'] = outages['OUTAGE.RESTORATION.DATE'].map(str)
outages['OUTAGE.RESTORATION.TIME'] = outages['OUTAGE.RESTORATION.TIME'].map(str)
outages['OUTAGE.RESTORATION.DATE'] = pd.to_datetime(outages['OUTAGE.RESTORATION.DAT
outages['OUTAGE.RESTORATION.TIME'] = pd.to_timedelta(outages['OUTAGE.RESTORATION.TI

#This combines the two columns inoformation into one column
outages['OUTAGE.RESTORATION'] = outages['OUTAGE.RESTORATION.DATE'] + outages['OUTAG
```

```
#Dropping the Date and Time Columns for OUTAGE and OUTAGE.RESTORATION
outages = outages.drop(columns = ['OUTAGE.START.DATE', 'OUTAGE.START.TIME', 'OUTAGE
```

Cleaned DataFrame

In [5]:
```
outages
```

Out[5]:

	YEAR	MONTH	U.S._STATE	POSTAL.CODE	NERC.REGION	CLIMATE.REGION	ANOMALY.LEVE
0	2011	7.0	Minnesota	MN	MRO	East North Central	-0.
1	2014	5.0	Minnesota	MN	MRO	East North Central	-0.
2	2010	10.0	Minnesota	MN	MRO	East North Central	-1.
3	2012	6.0	Minnesota	MN	MRO	East North Central	-0.
4	2015	7.0	Minnesota	MN	MRO	East North Central	1.
...	...	...	...	...	...	...	...
1529	2011	12.0	North Dakota	ND	MRO	West North Central	-0.
1530	2006	NaN	North Dakota	ND	MRO	West North Central	Na
1531	2009	8.0	South Dakota	SD	RFC	West North Central	0.
1532	2009	8.0	South Dakota	SD	MRO	West North Central	0.
1533	2000	NaN	Alaska	AK	ASCC	NaN	Na

1534 rows × 53 columns

Main Question:

Is there a significant difference between the outages distributions of the seasons in the SPP Region and the Overall outages distributions of the seasons of the NERC Regions?

Univariate Analysis

Counts of Outage Causes

In [6]:
```
#This gets the count of each CAUSE.CATEGORY.DETAIL and sorts them by decsencing ord
detail_counts = outages.groupby('CAUSE.CATEGORY.DETAIL').count().sort_values(by = '

#This plots the results in a bar graph
fig = px.bar(detail_counts, y='YEAR' ,barmode="group", color_discrete_sequence=px.c

#This code makes changes the layout of the graph to make it more professional looki
fig.update_layout(


    yaxis_title= 'Count',
    xaxis_title="Cause Category Detail",
    yaxis=dict(showgrid=False, tickfont = dict(size=18)),
    xaxis = dict(showgrid=False, tickfont = dict(size=10.5)),
    font=dict(family="Lato",color="black", size = 20),
    plot_bgcolor='rgba(0,0,0,0)',
    title={
        'text': 'Counts of Each Cause Category Detail',
        'y':0.9,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'}


)
```

Counts Outages in Each State

In [7]:
```
#This gets the count of outages in each state and sorts them by decsencing order
num_outages = outages.groupby('U.S._STATE').count().sort_values(by = 'YEAR', ascend

#This plots the results in a bar graph
fig = px.bar(num_outages, y='YEAR' ,barmode="group", color_discrete_sequence=px.col

#This code makes changes the layout of the graph to make it more professional looki
fig.update_layout(


    yaxis_title= 'Count',
    xaxis_title="U.S States",
    yaxis=dict(showgrid=False, tickfont = dict(size=18)),
    xaxis = dict(showgrid=False, tickfont = dict(size=10.5)),
    font=dict(family="Lato",color="black", size = 20),
    plot_bgcolor='rgba(0,0,0,0)',
    title={
        'text': 'Count of Outages in Each State',
        'y':0.9,
        'x':0.5,
        'xanchor': 'center',
```

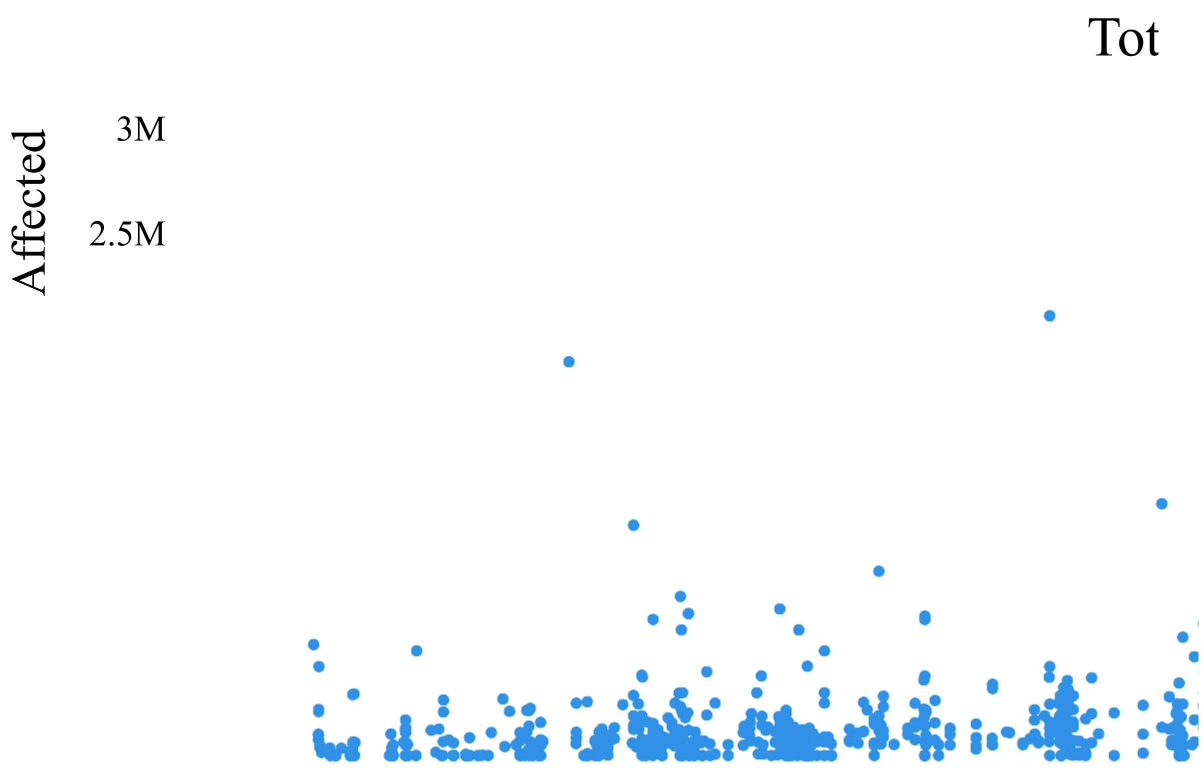
```
        'yanchor': 'top'}


)

fig.show()
```

Bivariate Analysis

Total Customers vs. Customers Affected

In [8]:
```
#This is a scatter plot with TOTAL.CUSTOMERS vs. CUSTOMERS.AFFECTED
fig = outages.plot.scatter(x = 'TOTAL.CUSTOMERS', y = 'CUSTOMERS.AFFECTED', color_d

#This code makes changes the layout of the graph to make it more professional looki
fig.update_layout(
    yaxis_title= 'Number Customoers Affected',
    xaxis_title="Total Customers",
    yaxis=dict(showgrid=False, tickfont = dict(size=18)),
    xaxis = dict(showgrid=False, tickfont = dict(size=10.5)),
    font=dict(family="Lato",color="black", size = 20),
    plot_bgcolor='rgba(0,0,0,0)',
```

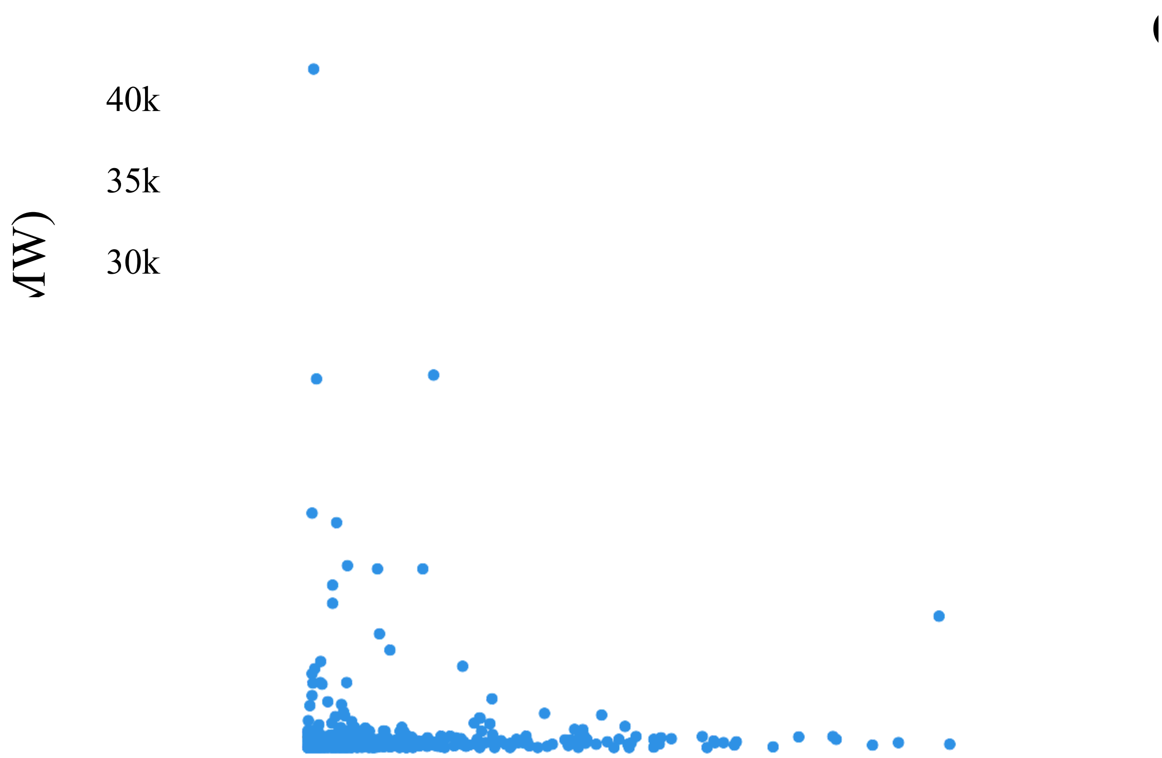
```
    title={
        'text': 'Total Customers Vs. Customers Affected',
        'y':0.9,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'}

)
```



Outage Duration Vs. MegaWatts Demand Loss

In [9]:
```
#This is a scatter plot with 'OUTAGE.DURATION' vs. 'DEMAND.LOSS.MW'
fig = outages.plot.scatter(x = 'OUTAGE.DURATION', y = 'DEMAND.LOSS.MW', color_discr

#This code makes changes the layout of the graph to make it more professional looki

fig.update_layout(


    yaxis_title= 'Demand Loss (MW)',
    xaxis_title="Outage Duration (Minutes)",
    yaxis=dict(showgrid=False, tickfont = dict(size=18)),
```

```
    xaxis = dict(showgrid=False, tickfont = dict(size=10.5)),
    font=dict(family="Lato",color="black", size = 20),
    plot_bgcolor='rgba(0,0,0,0)',
    title={
        'text': 'Outage Duration Vs. Demand Loss',
        'y':0.9,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'}


)
```



Interesting Aggregates

The Number of Each Cause Category in Each State

In [10]:
```
#To create this table I used made a pd.pivot_table() with index being U.S_STATE, co
#and the values for this table did not necessarly matter as long as it contained no
#chose RES.CUSTOMERS. Lastly, the aggfunc I chose was count. As a result, this gave
#for each state.
```

```
aggregate = pd.pivot_table(outages, index = 'U.S._STATE', columns = 'CAUSE.CATEGORY
aggregate.head(10)
```

Out[10]:

CAUSE.CATEGORY U.S._STATE	equipment failure	fuel supply emergency	intentional attack	islanding	public appeal	severe weather	system operability disruption
Alabama	0	0	1	0	0	5	0
Alaska	1	0	0	0	0	0	0
Arizona	4	0	18	0	0	4	2
Arkansas	1	0	6	1	7	10	0
California	21	17	24	28	9	70	41
Colorado	0	1	5	1	0	4	4
Connecticut	0	0	8	0	0	10	0
Delaware	1	0	37	0	0	2	1
District of Columbia	1	0	0	0	0	9	0
Florida	4	0	2	0	3	26	10

Assessment of Missingness

NMAR Analysis:

In the Outages data frame, the 'CAUTEGORY.CAUSE.DETAIL' column is supposed to give a specific reason for the 'CATEGORY.CAUSE' of the outage. This column could be NMAR because the reason why a value would be missing can be due to negligence of the person logging the data because they may have felt the specific reason may not be significant enough or if an investigation was not done. This column would be MAR if another column provided information whether an investigation was done to figure out the specific cause of the outage.

Missingness Dependency

Distributions of YEAR by Missingness of CUSTOMERS.AFFECTED

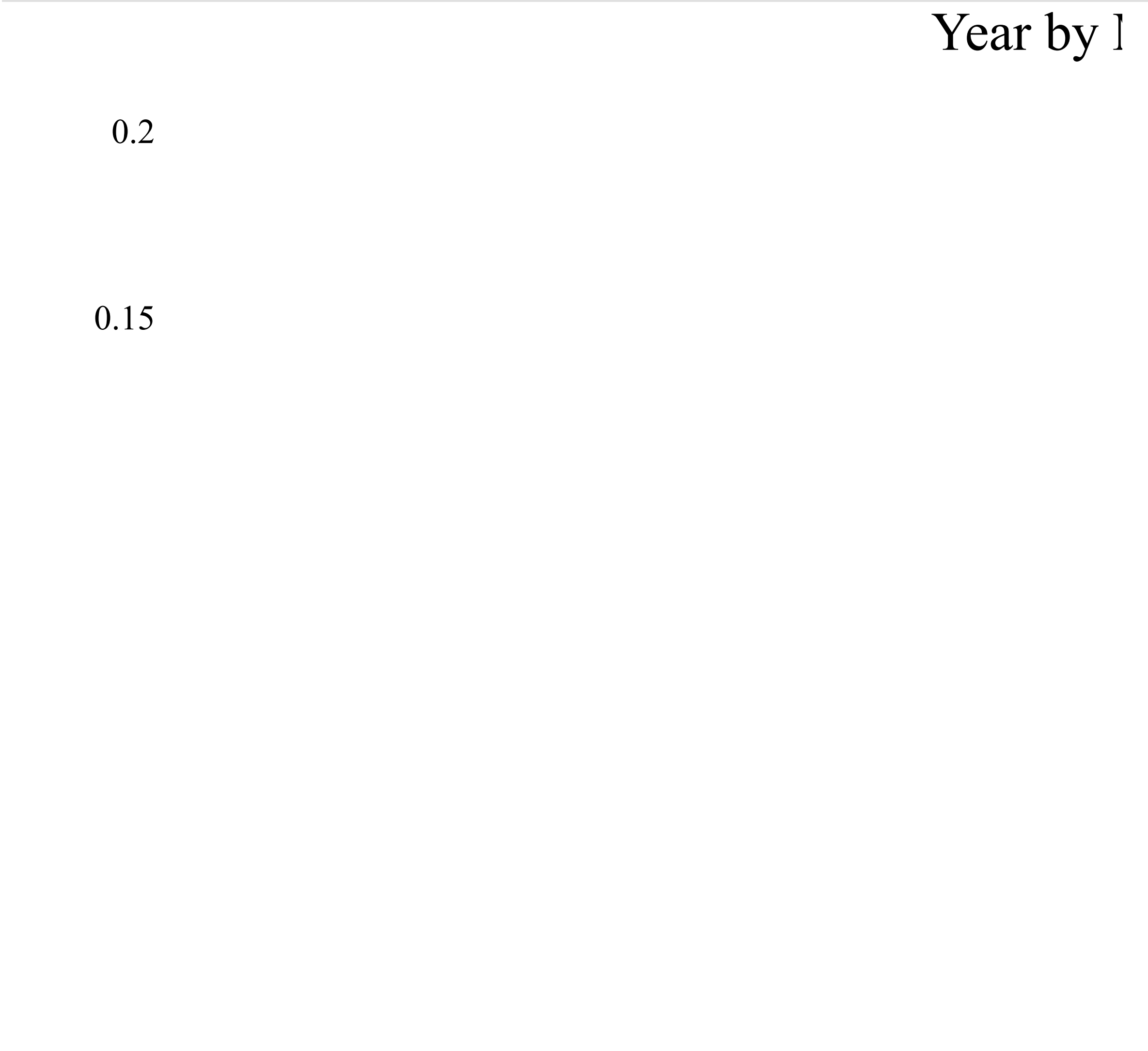
In [11]:
```
outages_copy = outages.copy()

#This line creates a df with number of CUSTOMER.AFFECTED values for each year that
CUSTOMERS_AFFECTED_dist = (outages_copy.assign(CUSTOMERS_AFFECTED_Missing = outages

#This reanmes the column names to make it obvious which column represents when CUST
CUSTOMERS_AFFECTED_dist.columns = ['CUSTOMERS AFFECTED Missing = False', 'CUSTOMERS

#This converts the counts of the missing and non missing values into distributions
CUSTOMERS_AFFECTED_dist = CUSTOMERS_AFFECTED_dist / CUSTOMERS_AFFECTED_dist.sum()
CUSTOMERS_AFFECTED_dist
```

Out[11]:

YEAR	CUSTOMERS AFFECTED Missing = False	CUSTOMERS AFFECTED Missing = True
2000	0.019248	0.011287
2001	0.007333	0.015801
2002	0.014665	0.002257
2003	0.039413	0.006772
2004	0.058662	0.015801
2005	0.046746	0.009029
2006	0.053162	0.020316
2007	0.037580	0.033860
2008	0.088909	0.031603
2009	0.047663	0.058691
2010	0.067828	0.072235
2011	0.190651	0.137698
2012	0.111824	0.117381
2013	0.070577	0.171558
2014	0.028414	0.182844
2015	0.071494	0.092551
2016	0.045830	0.020316

In [12]:
```
# This plots the distribtion table of Year by Missingness of CUSTOMERS.AFFECTED
fig = px.bar(CUSTOMERS_AFFECTED_dist,barmode="group", color_discrete_sequence=px.co

#This changes the layout of the plot to make it more professional looking
fig.update_layout(
    yaxis_title= 'Value',
    xaxis_title="Year",
    yaxis=dict(showgrid=False, tickfont = dict(size=18)),
    xaxis = dict(showgrid=False, tickfont = dict(size=10.5)),
    font=dict(family="Lato",color="black", size = 20),
    plot_bgcolor='rgba(0,0,0,0)',
```

```
    title={
        'text': 'Year by Missingness of CUSTOMERS AFFECTED',
        'y':0.9999,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'}


)
```



Permutation Test to Verify Whether the Missingness of CUSTOMERS.AFFECTED is Dependent on YEAR Column

Null Hypothesis: The distribution of 'YEAR' when 'CUSTOMERS' is missing is the same as the distribution of 'YEAR' when 'CUSTOMERS.AFFECTED' is not missing.

Alternative Hypothesis: The distribution of 'YEAR' when 'CUSTOMERS' is missing is not the same as the distribution of 'YEAR' when 'CUSTOMERS.AFFECTED' is not missing.

In [13]:
```
n_repetitions = 500
CUSTOMERS_AFFECTED_Missing = outages.copy()
#This creates a boolean column of whether CUSTOMERS.AFFECTED is missing for that ro
```

```
CUSTOMERS_AFFECTED_Missing['CUSTOMERS_AFFECTED_Missing'] = CUSTOMERS_AFFECTED_Missi
shuffled = CUSTOMERS_AFFECTED_Missing

tvds = []

#This for loop repeats a single permutation experiment
for _ in range(n_repetitions):

    #This shuffles the CUSTOMERS.AFFECTED column
    shuffled['CUSTOMERS_AFFECTED_Missing'] = np.random.permutation(shuffled['CUSTOM

    # This Computes and stores the TVD.
    pivoted = (
        shuffled
        .pivot_table(index='YEAR', columns='CUSTOMERS_AFFECTED_Missing', aggfunc='s
        .apply(lambda x: x / x.sum())
    )

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

Code Cited:

This code comes from Lecutre 12 Identifying Missingness Mechanisms, with a sligthly modified output.

In [14]:
```
#This calculates the observed tvd in the original data set
observed_tvd = CUSTOMERS_AFFECTED_dist.diff(axis=1).iloc[:, -1].abs().sum() / 2

#This calculates the p value for this permutation test
p_value = (np.array(tvds) >= observed_tvd).mean()
p_value
```

Out[14]: 0.0

In [15]:
```
observed_tvd
```

Out[15]: 0.3059280424900634

In [16]:
```
#This plots the empirical distrubtion of the TVD
fig = px.histogram(pd.DataFrame(tvds), x=0, nbins=50, histnorm='probability',
                   title='Empirical Distribution of the TVD', color_discrete_sequen
fig.add_vline(x=observed_tvd, line_color='green')
fig.add_annotation(text=f'<span style="color:red">Observed TVD = {round(observed_tv
                   x=2.3 * observed_tvd, showarrow=False, y=0.16)
fig.update_layout(yaxis_range=[0, 0.15],
                  xaxis_range=[0, 0.35],
                 yaxis_title= 'Probability',
    xaxis_title="",
    yaxis=dict(showgrid=False, tickfont = dict(size=18)),
    xaxis = dict(showgrid=False, tickfont = dict(size=10.5)),
    font=dict(family="Lato",color="black", size = 20),
```

```
    plot_bgcolor='rgba(0,0,0,0)',
    title={
        'text': 'Empirical Distribution of the TVD',
        'y':0.9,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'}
)
```

Conclusion:

We reject the null, there is strong enough evidence to suggest the distribution of 'YEAR' when 'CUSTOMERS.AFFECTED' is missing is not the same as the distribution of 'YEAR' when 'CUSTOMERS.AFFECTED' is not missing. As a result, the evidence suggests that the 'CUSTOMERS.AFFECTED' column is dependent on the 'YEAR' column.

Is the Missingness of 'CUSTOMERS.AFFECTED' is Dependent on 'PC.REALGSP.CHANGE' Column

In [17]:
```
details_missing = outages.copy()
#This creates a boolean column that states whether the CUSTOMERS.AFFECTED value is
```

```
details_missing['CUSTOMERS_AFFECTED_Missing'] = details_missing['CUSTOMERS.AFFECTED

#This calculates the differences of means between the distributions of the 'PC.REAL
details_missing.groupby('CUSTOMERS_AFFECTED_Missing')['PC.REALGSP.CHANGE'].mean().d
```

Out[17]: 0.05779339682565965

In [18]:

```
outages_copy = outages.copy()

#This line creates a df with number of CUSTOMER.AFFECTED values for each year that
CUSTOMERS_AFFECTED_dist = (outages_copy.assign(CUSTOMERS_AFFECTED_Missing = outages

#This reanmes the column names to make it obvious which column represents when CUST
CUSTOMERS_AFFECTED_dist.columns = ['CUSTOMERS AFFECTED Missing = False', 'CUSTOMERS

#This converts the counts of the missing and non missing values into distributions
CUSTOMERS_AFFECTED_dist = CUSTOMERS_AFFECTED_dist / CUSTOMERS_AFFECTED_dist.sum()
```

In [19]:

```
#This plots the kernel density plot to figure out the appropiate test to use
kde_plot = create_kde_plotly(details_missing[['CUSTOMERS_AFFECTED_Missing', 'PC.REA
kde_plot
```

Since 'PC.REALGSP.CHANGE' is numerical and the distributions of when CUSTOMERS.AFFECTED have similar means but the distributions look different a Kolmogorov-Smirnov test needs to be preformed

Permutation Test to test whether the missingness of category details is dependent on U.S_STATE Column

Null Hypothesis: The shape of the distribution of 'PC.REALGSP.CHANGE' when 'CUSTOMERS.AFFECTED' is missing is the same as the shape of the distribution of 'PC.REALGSP.CHANGE' when 'CUSTOMERS.AFFECTED' is not missing.

Alternative Hypothesis: The shape of the distribution of 'PC.REALGSP.CHANGE' when 'CUSTOMERS.AFFECTED' is missing is not the same as the shape of the distribution of 'PC.REALGSP.CHANGE' when 'CUSTOMERS.AFFECTED' is not missing.

In [20]:
```
# This performs the K-S test
k_s_test = ks_2samp(details_missing.loc[details_missing['CUSTOMERS_AFFECTED_Missing
observed_ks = k_s_test.statistic
observed_ks
```

Out[20]: 0.055216805672514496

In [21]:
```
p_value = k_s_test.pvalue
p_value
```

Out[21]: 0.27920234146101947

Conclusion:

Since the p-value is large. We fail to reject the null, there is not enough evidence to suggest that the shape of the distribution of 'PC.REALGSP.CHANGE' when 'CUSTOMERS.AFFECTED' is missing is not the same as the shape of the distribution of 'PC.REALGSP.CHANGE' when 'CUSTOMERS.AFFECTED' is not missing. Which suggests the missingness of 'CUSTOMERS.AFFECTED' is not dependent on 'PC.REALGSP.CHANGE'.

Hypothesis Testing

Null Hypothesis: There is not a difference between the outages distributions of the seasons in the SPP Region and the Overall outages distributions of the seasons of the NERC Regions

Alternative Hypothesis: There is a difference between the outages distributions of the seasons in the SPP Region and the Overall outages distributions of the seasons of the NERC Regions

In [22]:
```
#This is a helper function to figure out the season in which the outage occured
def season(sr):
    seasons = {1: 'Winter', 2: 'Winter', 3: 'Spring', 4: 'Spring', 5: 'Spring', 6:
    if np.isnan(sr) == False:
        return seasons[int(sr)]
```

In [23]:
```
#Creates a new categorical column that states the season in which the outage occure
outages['Season'] = outages['MONTH'].apply(season)
```

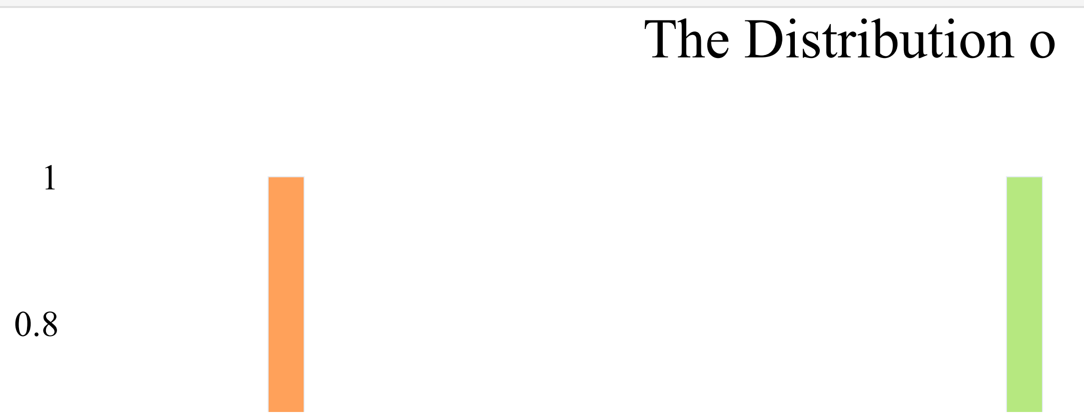
In [24]:
```
#This creates a distribution table of the outages that occured in during each seaso
NERC_Season = pd.pivot_table(outages, index = 'Season', columns = 'NERC.REGION',  v
overall_dist = NERC_Season.sum(axis = 1) /  NERC_Season.sum(axis = 1).sum()
NERC_Season = NERC_Season/NERC_Season.sum()
NERC_Season['Overall'] = overall_dist
NERC_Season.T
```

Out[24]:

Season NERC.REGION	Fall	Spring	Summer	Winter
ECAR	0.235294	0.264706	0.411765	0.088235
FRCC	0.325581	0.186047	0.325581	0.162791
FRCC, SERC	0.000000	0.000000	0.000000	1.000000
HECO	0.666667	0.000000	0.333333	0.000000
HI	1.000000	0.000000	0.000000	0.000000
MRO	0.177778	0.222222	0.422222	0.177778
NPCC	0.253333	0.280000	0.313333	0.153333
PR	0.000000	1.000000	0.000000	0.000000
RFC	0.277512	0.148325	0.394737	0.179426
SERC	0.159204	0.258706	0.358209	0.223881
SPP	0.227273	0.181818	0.469697	0.121212
TRE	0.198198	0.315315	0.333333	0.153153
WECC	0.288248	0.237251	0.286031	0.188470
Overall	0.253115	0.221639	0.346885	0.178361

In [25]:
```
# This plots the distribtions
NERC_Season_graph = px.bar(NERC_Season, barmode="group", color_discrete_sequence=px
NERC_Season_graph.update_layout(

                yaxis_title= 'Value',
```

```
    xaxis_title="Season",
    yaxis=dict(showgrid=False, tickfont = dict(size=18)),
    xaxis = dict(showgrid=False, tickfont = dict(size=10.5)),
    font=dict(family="Lato",color="black", size = 20),
    plot_bgcolor='rgba(0,0,0,0)',
    title={
        'text': 'The Distribution of Outages during Each Season in each NERC Region
        'y':0.9999,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'}
)
```



In [26]:
```
#This plots the the distribution of outages during each Season for the SPP Region a
spp_vs_overall = pd.DataFrame([NERC_Season['SPP'], NERC_Season['Overall']]).T
spp_vs_overall = px.bar(spp_vs_overall, barmode="group", color_discrete_sequence=px

spp_vs_overall.update_layout(
    yaxis_range=[0, 0.6],
    yaxis_title= 'Value',
    xaxis_title="Season",
    yaxis=dict(showgrid=False, tickfont = dict(size=18)),
    xaxis = dict(showgrid=False, tickfont = dict(size=10.5)),
    font=dict(family="Lato",color="black", size = 20),
```

```
    plot_bgcolor='rgba(0,0,0,0)',
    title={
        'text': 'The Distribution of Outages during Each Season',
        'y':0.9999,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'}
)
```

The Di

0.6

0.5

0.4

In [27]:
```
#This helper function calculates the tvd between two given distributions
def total_variation_distance(dist1, dist2):
    return np.sum(np.abs(dist1 - dist2)) / 2
```

In [28]:
```
#This calculated the oberserved total variation distance
observed_tvd = total_variation_distance(NERC_Season['SPP'], NERC_Season['Overall'])
observed_tvd
```

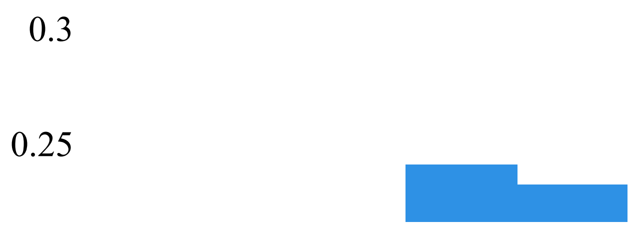
Out[28]:
```
0.12281172379533038
```

In [29]:
```
# This generates many random samples under the null and calculates the tvds of each
num_reps = 100_000
num_outages = outages['Season'].count()
```

```
season_dist = np.random.multinomial(num_outages, NERC_Season['Overall'], size=num_r
tvds = np.sum(np.abs(season_dist - NERC_Season['Overall'].to_numpy()), axis=1) / 2
```

Code Cited: The code used in the last 3 code cells comes from Lecture 9 – Hypothesis Testing, modified slightly for the scope of my specific test.

In [30]:
```
#This plots the emirical distribution of the TVD
fig = px.histogram(pd.DataFrame(tvds), x=0, nbins=20, histnorm='probability',
                   title='Empirical Distribution of the TVD', color_discrete_sequen
fig.add_vline(x=observed_tvd, line_color='green')
fig.update_layout(yaxis_range=[0, 0.3],
                 yaxis_title= 'Probability',
    xaxis_title="",
    yaxis=dict(showgrid=False, tickfont = dict(size=18)),
    xaxis = dict(showgrid=False, tickfont = dict(size=10.5)),
    font=dict(family="Lato",color="black", size = 20),
    plot_bgcolor='rgba(0,0,0,0)',
    title={
        'text': 'Empirical Distribution of the TVD',
        'y':0.9,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'}
)
```



In [31]:
```
p_value = (np.array(tvds) >= observed_tvd).mean()
p_value
```

Out[31]: 0.0

Conclusion

The probability that the observed TVD came from the distrubtion of TVDs under that the assumption that null is true is essentially 0. There is strong enough evidence to suggest that there is a difference between the outages distributions of the seasons in the SPP Region and the Overall outages distributions of the seasons of the NERC Regions