# COGS 108 - Analysis on possible relationship between outage duration time in the US and potential factors

(https://youtu.be/3F8wyYpwGhE)

## Permissions

Place an `X` in the appropriate bracket below to specify if you would like your group's project to be made available to the public. (Note that student names will be included (but PIDs will be scraped from any groups who include their PIDs).

- [ X ] YES - make available
- [ ] NO - keep private

# Overview

In the United States, power outages are always an interesting topic to research. California, a big state which has more serious power issues than all other states. There are many factors responsible for putting almost a million Americans into darkness: environmental impact, climate change and population growth. 2020 as a special year, faces the worst power outage. **[1]** There are a variety of reasons that power outages happen. **With our analysis, we will find the most common weather disaster that causes the outage in certain year, we also try to measure the relationship between power outage duration and Sales Price, Density of population and nature land/water resource, based on power outage information in the US from 2000-2016. Using z-test, we found that the cause and year with the most amount of outage occurrences is significantly related to outage duration.** As a result, we find that outage duration negatively correlates with the occurence of outage, and doesn't have correlation with any other single factors or combination of multiple factors.

# Names


- Chi, Yunxiang
- Liu, Ziyan
- Ni, Jiajun
- Wu, Peicong
- Zhang, Xiaoxuan


# Research Question

What is the most common reason that causes those major power outage in the continential U.S? In addition, is there a relationship between the last time of outage with the electricity consumption and number of consumer served in that state (and would that be specifically related to a certain category, like residential, commercial, or industrial)? Is the land composition (percent of land and water) also a potential factor?

## Background & Prior Work

Nowadays, with the technological revolutions during the recent decades, various kinds of electrical products have already become the indispensable part of our life. With even small power outage for a few hours, people's normal life pace can be greatly affected.

From our prior works, we found that 2020 was actually the worst power outage year in the U.S., "on average, a person in the US went through eight hours without electricity in 2020", twice as long as the average outage time in 2013. **[1]** We can tell that power outage is becoming a serious problem we need to face and resolve. Therefore, we want to find what factors are related to number of occurrences of outages and what factors are related to duration of outages.

The primary cause of the outages is the severe weather condition. Heavy rain, strong wind, lightening, etc. can all lead to power outage. Even dirt in the air between electricity lines can be turned into conductor when light rain comes, making short circuit more likely to happen. **[2]**

Total power demand in an area could be deterministic in the duration of outage events, as higher power demand in a area makes the electrical more delicate and likely to melt and fail. **[3]** The more severe damage the electrical equiments receive, the longer time is needed to fix the outage.

References (include links):


- 1) "2020 was the worst year yet for power outages in the US", https://www.theverge.com/2021/11/10/22774266/power-outages-worse-united-states-electricity-grid-climate-change

- 2) "Light Rain in Bay Area, Weather Causes Power Outages in 7 Cities in East Bay", https://www.nbcbayarea.com/news/local/light-rain-in-bay-area-weather-causes-power-outages-in-7-cities-in-east-bay/2081312/
- 3) "8 Common Causes of Outages", https://energized.edison.com/stories/8-common-causes-of-outages

# Hypothesis

We believe that the severe weather are possibly the most common reason for the power outage events in U.S based on our background research, as we think the heavy rain could short-cuts the lines and strong wind could knock down power lines and blow objects into overhead lines.

For the relationship, we think greater electricity consumption and more consumer served in that state cause longer outage event because more comsumption will make it harder to repair, and we also agree that there is a positive relationship between the land percentage and outage duration since larger land percentage will make time of going between two outage area become longer.

# Dataset(s)

- Dataset Name: Data on major power outage events in the continental U.S.
- Link to the dataset: https://www.semanticscholar.org/paper/Data-on-major-power-outage-events-in-the-U.S.-Mukherjee-Nateghi/73b7b0b2f79960fee01dad12e45fe5f87d7685b9
- Number of observations: 1534

This dataset includes all major power outage events from 2000-2016 in U.S.. It has each outage's start time, duration, cause,etc. for outage information; it also contains happening place's climate region, electricity price, electricity consumption, etc.

# Setup

In [1]:

```
!pip install pandas
%pip install seaborn
!pip install openpyxl
```

```
WARNING: You are using pip version 22.0.4; however, version 22.1.2 is available.
You should consider upgrading via the 'C:\Users\16001\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.Py
thon.3.8_qbz5n2kfra8p0\python.exe -m pip install --upgrade pip' command.
Requirement already satisfied: pandas in c:\users\16001\appdata\local\packages\pythonsoftwarefoundation.python.3.8_qb
z5n2kfra8p0\localcache\local-packages\python38\site-packages (1.4.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\16001\appdata\local\packages\pythonsoftwarefoundation.python.
3.8_qbz5n2kfra8p0\localcache\local-packages\python38\site-packages (from pandas) (2022.1)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\16001\appdata\local\packages\pythonsoftwarefoundati
on.python.3.8_qbz5n2kfra8p0\localcache\local-packages\python38\site-packages (from pandas) (2.8.2)
Requirement already satisfied: numpy>=1.18.5 in c:\users\16001\appdata\local\packages\pythonsoftwarefoundation.pytho
n.3.8_qbz5n2kfra8p0\localcache\local-packages\python38\site-packages (from pandas) (1.22.3)
Requirement already satisfied: six>=1.5 in c:\users\16001\appdata\local\packages\pythonsoftwarefoundation.python.3.8_
qbz5n2kfra8p0\localcache\local-packages\python38\site-packages (from python-dateutil>=2.8.1->pandas) (1.15.0)
Requirement already satisfied: seaborn in c:\users\16001\appdata\local\packages\pythonsoftwarefoundation.python.3.8_q
bz5n2kfra8p0\localcache\local-packages\python38\site-packages (0.11.2)
Requirement already satisfied: pandas>=0.23 in c:\users\16001\appdata\local\packages\pythonsoftwarefoundation.python.
3.8_qbz5n2kfra8p0\localcache\local-packages\python38\site-packages (from seaborn) (1.4.2)
Requirement already satisfied: numpy>=1.15 in c:\users\16001\appdata\local\packages\pythonsoftwarefoundation.python.
3.8_qbz5n2kfra8p0\localcache\local-packages\python38\site-packages (from seaborn) (1.22.3)
Requirement already satisfied: matplotlib>=2.2 in c:\users\16001\appdata\local\packages\pythonsoftwarefoundation.pyth
on.3.8_qbz5n2kfra8p0\localcache\local-packages\python38\site-packages (from seaborn) (3.5.1)
Requirement already satisfied: scipy>=1.0 in c:\users\16001\appdata\local\packages\pythonsoftwarefoundation.python.3.
8_qbz5n2kfra8p0\localcache\local-packages\python38\site-packages (from seaborn) (1.8.0)
Requirement already satisfied: packaging>=20.0 in c:\users\16001\appdata\local\packages\pythonsoftwarefoundation.pyth
on.3.8_qbz5n2kfra8p0\localcache\local-packages\python38\site-packages (from matplotlib>=2.2->seaborn) (21.3)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\16001\appdata\local\packages\pythonsoftwarefoundation.py
thon.3.8_qbz5n2kfra8p0\localcache\local-packages\python38\site-packages (from matplotlib>=2.2->seaborn) (4.32.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\16001\appdata\local\packages\pythonsoftwarefoundation.pytho
n.3.8_qbz5n2kfra8p0\localcache\local-packages\python38\site-packages (from matplotlib>=2.2->seaborn) (9.1.0)
Note: you may need to restart the kernel to use updated packages.Requirement already satisfied: python-dateutil>=2.7
in c:\users\16001\appdata\local\packages\pythonsoftwarefoundation.python.3.8_qbz5n2kfra8p0\localcache\local-packages
\python38\site-packages (from matplotlib>=2.2->seaborn) (2.8.2)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\16001\appdata\local\packages\pythonsoftwarefoundation.pyt
hon.3.8_qbz5n2kfra8p0\localcache\local-packages\python38\site-packages (from matplotlib>=2.2->seaborn) (3.0.8)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\16001\appdata\local\packages\pythonsoftwarefoundation.py
thon.3.8_qbz5n2kfra8p0\localcache\local-packages\python38\site-packages (from matplotlib>=2.2->seaborn) (1.4.2)
Requirement already satisfied: cycler>=0.10 in c:\users\16001\appdata\local\packages\pythonsoftwarefoundation.python.
3.8_qbz5n2kfra8p0\localcache\local-packages\python38\site-packages (from matplotlib>=2.2->seaborn) (0.11.0)
Requirement already satisfied: pytz>=2020.1 in c:\users\16001\appdata\local\packages\pythonsoftwarefoundation.python.
3.8_qbz5n2kfra8p0\localcache\local-packages\python38\site-packages (from pandas>=0.23->seaborn) (2022.1)
Requirement already satisfied: six>=1.5 in c:\users\16001\appdata\local\packages\pythonsoftwarefoundation.python.3.8_
qbz5n2kfra8p0\localcache\local-packages\python38\site-packages (from python-dateutil>=2.7->matplotlib>=2.2->seaborn)
```

```
(1.15.0)

WARNING: You are using pip version 22.0.4; however, version 22.1.2 is available.
You should consider upgrading via the 'C:\Users\16001\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.Py
thon.3.8_qbz5n2kfra8p0\python.exe -m pip install --upgrade pip' command.
Requirement already satisfied: openpyxl in c:\users\16001\appdata\local\packages\pythonsoftwarefoundation.python.3.8_
qbz5n2kfra8p0\localcache\local-packages\python38\site-packages (3.0.9)
Requirement already satisfied: et-xmlfile in c:\users\16001\appdata\local\packages\pythonsoftwarefoundation.python.3.
8_qbz5n2kfra8p0\localcache\local-packages\python38\site-packages (from openpyxl) (1.1.0)
WARNING: You are using pip version 22.0.4; however, version 22.1.2 is available.
You should consider upgrading via the 'C:\Users\16001\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.Py
thon.3.8_qbz5n2kfra8p0\python.exe -m pip install --upgrade pip' command.
```

In [2]:

```
import pandas as pd
import numpy as np

# Import seaborn and apply its plotting styles
import seaborn as sns
sns.set(style="white", font_scale=2)

import patsy
import statsmodels.api as sm

import statsmodels.formula.api as smf

# import matplotlib
import matplotlib as mpl
import matplotlib.pyplot as plt
# set plotting size parameter
plt.rcParams['figure.figsize'] = (17, 7)
pd.set_option('display.max_columns', 30)
pd.set_option('display.max_rows', 50)
import warnings
warnings.filterwarnings('ignore')

from scipy.stats import skewnorm
from scipy import stats
from statsmodels.stats.weightstats import ztest as ztest
```

# Data Cleaning

Describe your data cleaning steps here.

## 1. Converting all data files into csv/excel, and read excel to dataframe

In [3]:

```
outage = pd.read_excel('Datasets/outage.xlsx',sheet_name='Masterdata',header=5)
```

In [4]:

```
outage
```

Out[4]:

| | variables | OBS | YEAR | MONTH | U.S._STATE | POSTAL.CODE | NERC.REGION | CLIMATE.REGION | ANOMALY.LEVEL | CLIMATE.CATEGORY |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Units | NaN | NaN | NaN | NaN | NaN | NaN | NaN | numeric | NaN |
| **1** | NaN | 1.0 | 2011.0 | 7.0 | Minnesota | MN | MRO | East North Central | -0.3 | normal |
| **2** | NaN | 2.0 | 2014.0 | 5.0 | Minnesota | MN | MRO | East North Central | -0.1 | normal |
| **3** | NaN | 3.0 | 2010.0 | 10.0 | Minnesota | MN | MRO | East North Central | -1.5 | cold |
| **4** | NaN | 4.0 | 2012.0 | 6.0 | Minnesota | MN | MRO | East North Central | -0.1 | normal |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **1530** | NaN | 1530.0 | 2011.0 | 12.0 | North Dakota | ND | MRO | West North Central | -0.9 | cold |
| **1531** | NaN | 1531.0 | 2006.0 | NaN | North Dakota | ND | MRO | West North Central | NaN | NaN |
| **1532** | NaN | 1532.0 | 2009.0 | 8.0 | South Dakota | SD | RFC | West North Central | 0.5 | warm |
| **1533** | NaN | 1533.0 | 2009.0 | 8.0 | South Dakota | SD | MRO | West North Central | 0.5 | warm |
| **1534** | NaN | 1534.0 | 2000.0 | NaN | Alaska | AK | ASCC | NaN | NaN | NaN |

1535 rows × 57 columns

## 2. Adjusting the dataframe to make it easier to handle and analyze

### 2.1 Clean columns and rows to delete introductory lines and columns

In [5]:
```
outage = outage.iloc[1:,1:]
outage.head()
```

Out[5]:

| | OBS | YEAR | MONTH | U.S._STATE | POSTAL.CODE | NERC.REGION | CLIMATE.REGION | ANOMALY.LEVEL | CLIMATE.CATEGORY | OUTAGE.START.D |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1.0 | 2011.0 | 7.0 | Minnesota | MN | MRO | East North Central | -0.3 | normal | 2011-07-01 00:0 |
| 2 | 2.0 | 2014.0 | 5.0 | Minnesota | MN | MRO | East North Central | -0.1 | normal | 2014-05-11 00:0 |
| 3 | 3.0 | 2010.0 | 10.0 | Minnesota | MN | MRO | East North Central | -1.5 | cold | 2010-10-26 00:0 |
| 4 | 4.0 | 2012.0 | 6.0 | Minnesota | MN | MRO | East North Central | -0.1 | normal | 2012-06-19 00:0 |
| 5 | 5.0 | 2015.0 | 7.0 | Minnesota | MN | MRO | East North Central | 1.2 | warm | 2015-07-18 00:0 |

5 rows × 56 columns

### 2.2 change type of OBS to int and make it the index for outage

In [6]:
```
outage['OBS'] = outage['OBS'].astype(int)
```

In [7]:
```
outage = outage.set_index('OBS')
outage.index.name = ""
```

In [8]:
```
outage.head()
```

Out[8]:

| | YEAR | MONTH | U.S._STATE | POSTAL.CODE | NERC.REGION | CLIMATE.REGION | ANOMALY.LEVEL | CLIMATE.CATEGORY | OUTAGE.START.DATE |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2011.0 | 7.0 | Minnesota | MN | MRO | East North Central | -0.3 | normal | 2011-07-01 00:00:00 |
| 2 | 2014.0 | 5.0 | Minnesota | MN | MRO | East North Central | -0.1 | normal | 2014-05-11 00:00:00 |
| 3 | 2010.0 | 10.0 | Minnesota | MN | MRO | East North Central | -1.5 | cold | 2010-10-26 00:00:00 |
| 4 | 2012.0 | 6.0 | Minnesota | MN | MRO | East North Central | -0.1 | normal | 2012-06-19 00:00:00 |
| 5 | 2015.0 | 7.0 | Minnesota | MN | MRO | East North Central | 1.2 | warm | 2015-07-18 00:00:00 |

5 rows × 55 columns

### 2.3 change column to correct type

Now let's take a look at the `outage` dataframe.

In [9]:
```
pd.set_option('display.max_columns', 100)
outage.head()
```

Out[9]:

| | YEAR | MONTH | U.S._STATE | POSTAL.CODE | NERC.REGION | CLIMATE.REGION | ANOMALY.LEVEL | CLIMATE.CATEGORY | OUTAGE.START.DATE |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2011.0 | 7.0 | Minnesota | MN | MRO | East North Central | -0.3 | normal | 2011-07-01 00:00:00 |
| 2 | 2014.0 | 5.0 | Minnesota | MN | MRO | East North Central | -0.1 | normal | 2014-05-11 00:00:00 |
| 3 | 2010.0 | 10.0 | Minnesota | MN | MRO | East North Central | -1.5 | cold | 2010-10-26 00:00:00 |
| 4 | 2012.0 | 6.0 | Minnesota | MN | MRO | East North Central | -0.1 | normal | 2012-06-19 00:00:00 |
| 5 | 2015.0 | 7.0 | Minnesota | MN | MRO | East North Central | 1.2 | warm | 2015-07-18 00:00:00 |

#### 2.3.1 replace NA with 0 in some columns for further changing type

In [10]:
```
outage['CUSTOMERS.AFFECTED'] = outage['CUSTOMERS.AFFECTED'].fillna(0)
```

#### 2.3.2 change type of some columns

In [11]:
```
outage['YEAR'] = outage['YEAR'].astype(int)
#outage['MONTH'] = outage['MONTH'].astype(int)
outage['CUSTOMERS.AFFECTED'] = outage['CUSTOMERS.AFFECTED'].astype(int)
outage['RES.CUSTOMERS'] = outage['RES.CUSTOMERS'].astype(int)
outage['COM.CUSTOMERS'] = outage['COM.CUSTOMERS'].astype(int)
outage['IND.CUSTOMERS'] = outage['IND.CUSTOMERS'].astype(int)
```

```
outage['TOTAL.CUSTOMERS'] = outage['TOTAL.CUSTOMERS'].astype(int)
outage['POPULATION'] = outage['POPULATION'].astype(int)
outage.head()
```

Out[11]:

| | YEAR | MONTH | U.S._STATE | POSTAL.CODE | NERC.REGION | CLIMATE.REGION | ANOMALY.LEVEL | CLIMATE.CATEGORY | OUTAGE.START.DATE |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2011 | 7.0 | Minnesota | MN | MRO | East North Central | -0.3 | normal | 2011-07-01 00:00:00 |
| 2 | 2014 | 5.0 | Minnesota | MN | MRO | East North Central | -0.1 | normal | 2014-05-11 00:00:00 |
| 3 | 2010 | 10.0 | Minnesota | MN | MRO | East North Central | -1.5 | cold | 2010-10-26 00:00:00 |
| 4 | 2012 | 6.0 | Minnesota | MN | MRO | East North Central | -0.1 | normal | 2012-06-19 00:00:00 |
| 5 | 2015 | 7.0 | Minnesota | MN | MRO | East North Central | 1.2 | warm | 2015-07-18 00:00:00 |

# 3 Filtering useful columns

## 3.1 Handling NA

### 3.1.1 Checking types of every columns

In [12]:

```
pd.set_option('display.max_rows', 60)
outage.dtypes
```

Out[12]:

```
YEAR                       int32
MONTH                    float64
U.S._STATE                object
POSTAL.CODE               object
NERC.REGION               object
CLIMATE.REGION            object
ANOMALY.LEVEL             object
CLIMATE.CATEGORY          object
OUTAGE.START.DATE         object
OUTAGE.START.TIME         object
OUTAGE.RESTORATION.DATE   object
OUTAGE.RESTORATION.TIME   object
CAUSE.CATEGORY            object
CAUSE.CATEGORY.DETAIL     object
HURRICANE.NAMES           object
OUTAGE.DURATION           object
DEMAND.LOSS.MW            object
CUSTOMERS.AFFECTED         int32
RES.PRICE                 object
COM.PRICE                 object
IND.PRICE                 object
TOTAL.PRICE               object
RES.SALES                 object
COM.SALES                 object
IND.SALES                 object
TOTAL.SALES               object
RES.PERCEN                object
COM.PERCEN                object
IND.PERCEN                object
RES.CUSTOMERS              int32
COM.CUSTOMERS              int32
IND.CUSTOMERS              int32
TOTAL.CUSTOMERS            int32
RES.CUST.PCT              object
COM.CUST.PCT              object
IND.CUST.PCT              object
PC.REALGSP.STATE          object
PC.REALGSP.USA            object
PC.REALGSP.REL            object
PC.REALGSP.CHANGE         object
UTIL.REALGSP              object
TOTAL.REALGSP             object
UTIL.CONTRI               object
PI.UTIL.OFUSA             object
POPULATION                 int32
POPPCT_URBAN              object
POPPCT_UC                 object
POPDEN_URBAN              object
POPDEN_UC                 object
POPDEN_RURAL              object
AREAPCT_URBAN             object
AREAPCT_UC                object
PCT_LAND                  object
PCT_WATER_TOT             object
PCT_WATER_INLAND          object
dtype: object
```

3.1.2 fillna with every related numeric columns and change type into int/float

In [13]:

```
# outage['ANOMALY.LEVEL'] = outage['ANOMALY.LEVEL'].fillna(0)


# change numeric columns' type
outage['RES.CUST.PCT'] = outage['RES.CUST.PCT'].astype(float)
outage['COM.CUST.PCT'] = outage['COM.CUST.PCT'].astype(float)
outage['IND.CUST.PCT'] = outage['IND.CUST.PCT'].astype(float)
outage['PC.REALGSP.STATE'] = outage['PC.REALGSP.STATE'].astype(int)
outage['PC.REALGSP.USA'] = outage['PC.REALGSP.USA'].astype(int)
outage['PC.REALGSP.REL'] = outage['PC.REALGSP.REL'].astype(float)
outage['PC.REALGSP.CHANGE'] = outage['PC.REALGSP.CHANGE'].astype(float)
outage['UTIL.REALGSP'] = outage['UTIL.REALGSP'].astype(int)
outage['TOTAL.REALGSP'] = outage['TOTAL.REALGSP'].astype(int)
outage['UTIL.CONTRI'] = outage['UTIL.CONTRI'].astype(float)
outage['PI.UTIL.OFUSA'] = outage['PI.UTIL.OFUSA'].astype(float)
outage['POPPCT_URBAN'] = outage['POPPCT_URBAN'].astype(float)
outage['POPPCT_UC'] = outage['POPPCT_UC'].astype(float)
outage['POPDEN_URBAN'] = outage['POPDEN_URBAN'].astype(float)
outage['POPDEN_UC'] = outage['POPDEN_UC'].astype(float)
outage['POPDEN_RURAL'] = outage['POPDEN_RURAL'].astype(float)
outage['AREAPCT_URBAN'] = outage['AREAPCT_URBAN'].astype(float)
outage['AREAPCT_UC'] = outage['AREAPCT_UC'].astype(float)
outage['PCT_LAND'] = outage['PCT_LAND'].astype(float)
outage['PCT_WATER_TOT'] = outage['PCT_WATER_TOT'].astype(float)
outage['PCT_WATER_INLAND'] = outage['PCT_WATER_INLAND'].astype(float)
```

3.1.3 dropna with subset of related numeric columns and change those types to int/float

In [14]:

```
outage = outage.dropna(subset=['RES.PRICE','COM.PRICE','IND.PRICE','TOTAL.PRICE','RES.SALES','COM.SALES','IND.SALES',
outage['RES.PRICE'] = outage['RES.PRICE'].astype(float)
outage['COM.PRICE'] = outage['COM.PRICE'].astype(float)
outage['IND.PRICE'] = outage['IND.PRICE'].astype(float)
outage['TOTAL.PRICE'] = outage['TOTAL.PRICE'].astype(float)
outage['RES.SALES'] = outage['RES.SALES'].astype(int)
outage['COM.SALES'] = outage['COM.SALES'].astype(int)
outage['IND.SALES'] = outage['IND.SALES'].astype(int)
outage['TOTAL.SALES'] = outage['TOTAL.SALES'].astype(int)
outage['RES.PERCEN'] = outage['RES.PERCEN'].astype(float)
outage['COM.PERCEN'] = outage['COM.PERCEN'].astype(float)
outage['IND.PERCEN'] = outage['IND.PERCEN'].astype(float)
outage['MONTH'] = outage['MONTH'].astype(int)
outage['OUTAGE.DURATION'] = outage['OUTAGE.DURATION'].astype(float)
```

## 3.2 dropping unrelated columns

Drop the columns of data that are not considered in our analysis.

In [15]:

```
outage = outage.drop(['POSTAL.CODE','CLIMATE.REGION','NERC.REGION','ANOMALY.LEVEL','HURRICANE.NAMES','CUSTOMERS.AFFEC
outage = outage.drop(['PC.REALGSP.STATE','PC.REALGSP.USA','UTIL.REALGSP','TOTAL.REALGSP','UTIL.CONTRI','PI.UTIL.OFUSA
```

Take a brief look at `outage` dataframe after filtering.

In [16]:

```
outage
```

Out[16]:

| | YEAR | MONTH | U.S._STATE | CLIMATE.CATEGORY | OUTAGE.START.DATE | OUTAGE.START.TIME | OUTAGE.RESTORATION.DATE | OUTAGE.RES |
|---|---|---|---|---|---|---|---|---|
| **1** | 2011 | 7 | Minnesota | normal | 2011-07-01 00:00:00 | 17:00:00 | 2011-07-03 00:00:00 | |
| **2** | 2014 | 5 | Minnesota | normal | 2014-05-11 00:00:00 | 18:38:00 | 2014-05-11 00:00:00 | |
| **3** | 2010 | 10 | Minnesota | cold | 2010-10-26 00:00:00 | 20:00:00 | 2010-10-28 00:00:00 | |
| **4** | 2012 | 6 | Minnesota | normal | 2012-06-19 00:00:00 | 04:30:00 | 2012-06-20 00:00:00 | |
| **5** | 2015 | 7 | Minnesota | warm | 2015-07-18 00:00:00 | 02:00:00 | 2015-07-19 00:00:00 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **1526** | 2011 | 6 | Idaho | normal | 2011-06-15 00:00:00 | 16:00:00 | 2011-06-16 00:00:00 | |
| **1527** | 2016 | 3 | Idaho | warm | 2016-03-08 00:00:00 | 00:00:00 | 2016-03-08 00:00:00 | |
| **1530** | 2011 | 12 | North Dakota | cold | 2011-12-06 00:00:00 | 08:00:00 | 2011-12-06 00:00:00 | |
| **1532** | 2009 | 8 | South Dakota | warm | 2009-08-29 00:00:00 | 22:54:00 | 2009-08-29 00:00:00 | |
| **1533** | 2009 | 8 | South | warm | 2009-08-29 00:00:00 | 11:00:00 | 2009-08-29 00:00:00 | |

| YEAR | MONTH | U.S._STATE | CLIMATE.CATEGORY | OUTAGE.START.DATE | OUTAGE.START.TIME | OUTAGE.RESTORATION.DATE | OUTAGE.RES |
|---|---|---|---|---|---|---|---|
| | | Dakota | | | | | |

1464 rows × 41 columns

## 4. Combining columns and adjusting format to prepare for EDA

### 4.1 Combining start.date&start.time and restoration.date&restoration.time

In [17]:
```
outage['OUTAGE.START.DATE'] = outage['OUTAGE.START.DATE'].astype(str).str.slice(start=0, stop=10)
outage['OUTAGE.RESTORATION.DATE'] = outage['OUTAGE.RESTORATION.DATE'].astype(str).str.slice(start=0, stop=10)
```

In [18]:
```
outage['OUTAGE.START.DATE'] = outage['OUTAGE.START.DATE'] + ' ' +outage['OUTAGE.START.TIME'].astype(str)
outage['OUTAGE.RESTORATION.DATE'] = outage['OUTAGE.RESTORATION.DATE'] + ' ' +outage['OUTAGE.RESTORATION.TIME'].astype
```

In [19]:
```
outage = outage.drop(['OUTAGE.START.TIME','OUTAGE.RESTORATION.TIME'], axis=1)
outage.columns = outage.columns.str.replace('OUTAGE.START.DATE','OUTAGE.START')
outage.columns = outage.columns.str.replace('OUTAGE.RESTORATION.DATE','OUTAGE.RESTORATION')
outage.head()
```

Out[19]:

| | YEAR | MONTH | U.S._STATE | CLIMATE.CATEGORY | OUTAGE.START | OUTAGE.RESTORATION | CAUSE.CATEGORY | CAUSE.CATEGORY.DETAIL | OU |
|---|---|---|---|---|---|---|---|---|---|
| **1** | 2011 | 7 | Minnesota | normal | 2011-07-01 17:00:00 | 2011-07-03 20:00:00 | severe weather | NaN | |
| **2** | 2014 | 5 | Minnesota | normal | 2014-05-11 18:38:00 | 2014-05-11 18:39:00 | intentional attack | vandalism | |
| **3** | 2010 | 10 | Minnesota | cold | 2010-10-26 20:00:00 | 2010-10-28 22:00:00 | severe weather | heavy wind | |
| **4** | 2012 | 6 | Minnesota | normal | 2012-06-19 04:30:00 | 2012-06-20 23:00:00 | severe weather | thunderstorm | |
| **5** | 2015 | 7 | Minnesota | warm | 2015-07-18 02:00:00 | 2015-07-19 07:00:00 | severe weather | NaN | |

### 4.2 sorted by year then month

In [20]:
```
outage.sort_values(by=['YEAR','MONTH'], ignore_index=True)
```

Out[20]:

| | YEAR | MONTH | U.S._STATE | CLIMATE.CATEGORY | OUTAGE.START | OUTAGE.RESTORATION | CAUSE.CATEGORY | CAUSE.CATEGORY.DETAIL |
|---|---|---|---|---|---|---|---|---|
| **0** | 2000 | 1 | South Carolina | cold | 2000-01-23 08:00:00 | 2000-01-28 12:00:00 | severe weather | winter storm |
| **1** | 2000 | 1 | South Carolina | cold | 2000-01-29 22:00:00 | 2000-02-03 12:00:00 | severe weather | winter storm |
| **2** | 2000 | 3 | Texas | cold | 2000-03-18 16:00:00 | 2000-03-18 17:10:00 | system operability disruption | transmission interruption |
| **3** | 2000 | 3 | New Mexico | cold | 2000-03-18 19:08:00 | 2000-03-18 19:08:00 | system operability disruption | transmission interruption |
| **4** | 2000 | 5 | Texas | cold | 2000-05-02 04:00:00 | 2000-05-02 12:00:00 | severe weather | NaN |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **1459** | 2016 | 5 | New York | warm | 2016-05-31 07:30:00 | 2016-06-13 07:27:00 | fuel supply emergency | NaN |
| **1460** | 2016 | 5 | Louisiana | warm | 2016-05-20 00:00:00 | 2016-05-22 05:00:00 | system operability disruption | transmission interruption |
| **1461** | 2016 | 6 | Utah | normal | 2016-06-07 12:00:00 | 2016-06-07 12:15:00 | intentional attack | sabotage |
| **1462** | 2016 | 6 | New Jersey | normal | 2016-06-14 07:59:00 | 2016-06-14 08:00:00 | intentional attack | vandalism |
| **1463** | 2016 | 6 | Delaware | normal | 2016-06-17 04:30:00 | 2016-06-17 04:31:00 | intentional attack | vandalism |

1464 rows × 39 columns

### 4.3 Standardize the column names

In [21]:
```
def standardize_name(string):
    string = string.lower()
    string = string.replace('.','_')
    return string
```

In [22]:
```
outage.columns = outage.columns.to_series().apply(standardize_name)
outage.rename({'u_s_state':'state'}, axis=1, inplace = True)
outage
```

Out[22]:

| | year | month | u_s__state | climate_category | outage_start | outage_restoration | cause_category | cause_category_detail | outage_duration |
|---|---|---|---|---|---|---|---|---|---|
| **1** | 2011 | 7 | Minnesota | normal | 2011-07-01 17:00:00 | 2011-07-03 20:00:00 | severe weather | NaN | 3060.0 |
| **2** | 2014 | 5 | Minnesota | normal | 2014-05-11 18:38:00 | 2014-05-11 18:39:00 | intentional attack | vandalism | 1.0 |
| **3** | 2010 | 10 | Minnesota | cold | 2010-10-26 20:00:00 | 2010-10-28 22:00:00 | severe weather | heavy wind | 3000.0 |
| **4** | 2012 | 6 | Minnesota | normal | 2012-06-19 04:30:00 | 2012-06-20 23:00:00 | severe weather | thunderstorm | 2550.0 |
| **5** | 2015 | 7 | Minnesota | warm | 2015-07-18 02:00:00 | 2015-07-19 07:00:00 | severe weather | NaN | 1740.0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **1526** | 2011 | 6 | Idaho | normal | 2011-06-15 16:00:00 | 2011-06-16 06:30:00 | intentional attack | vandalism | 870.0 |
| **1527** | 2016 | 3 | Idaho | warm | 2016-03-08 00:00:00 | 2016-03-08 00:00:00 | intentional attack | sabotage | 0.0 |
| **1530** | 2011 | 12 | North Dakota | cold | 2011-12-06 08:00:00 | 2011-12-06 20:00:00 | public appeal | NaN | 720.0 |
| **1532** | 2009 | 8 | South Dakota | warm | 2009-08-29 22:54:00 | 2009-08-29 23:53:00 | islanding | NaN | 59.0 |
| **1533** | 2009 | 8 | South Dakota | warm | 2009-08-29 11:00:00 | 2009-08-29 14:01:00 | islanding | NaN | 181.0 |

1464 rows × 39 columns

# Data Analysis & Results (EDA)

In [23]:
```
outage.describe()
```

Out[23]:

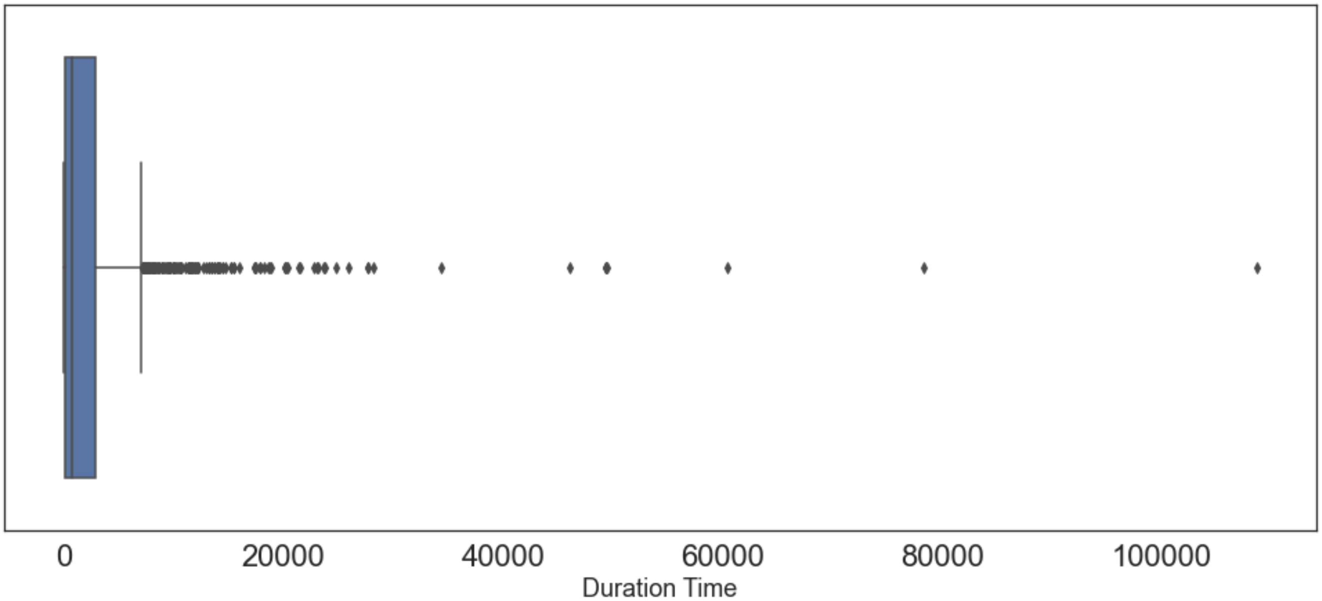
| | year | month | outage_duration | res_price | com_price | ind_price | total_price | res_sales | com_sales | i |
|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 1464.000000 | 1464.000000 | 1464.000000 | 1464.000000 | 1464.000000 | 1464.000000 | 1464.000000 | 1.464000e+03 | 1.464000e+03 | 1.464 |
| **mean** | 2010.080601 | 6.245902 | 2639.651639 | 11.970963 | 10.133805 | 7.344952 | 10.121523 | 4.342385e+06 | 4.410675e+06 | 2.787 |
| **std** | 3.645472 | 3.273736 | 5964.139426 | 3.090077 | 2.823382 | 2.457176 | 2.902191 | 3.387925e+06 | 3.487297e+06 | 2.206 |
| **min** | 2000.000000 | 1.000000 | 0.000000 | 5.650000 | 4.700000 | 3.200000 | 4.700000 | 1.444170e+05 | 1.525170e+05 | 1.552 |
| **25%** | 2008.000000 | 4.000000 | 99.500000 | 9.577500 | 8.010000 | 5.730000 | 7.970000 | 2.047998e+06 | 1.905618e+06 | 1.188 |
| **50%** | 2011.000000 | 6.000000 | 711.000000 | 11.500000 | 9.480000 | 6.720000 | 9.430000 | 3.466457e+06 | 3.182069e+06 | 2.295 |
| **75%** | 2013.000000 | 9.000000 | 2880.000000 | 13.850000 | 11.325000 | 8.590000 | 11.740000 | 5.978200e+06 | 6.956379e+06 | 3.998 |
| **max** | 2016.000000 | 12.000000 | 108653.000000 | 34.580000 | 32.140000 | 27.850000 | 31.290000 | 1.862066e+07 | 1.404697e+07 | 9.588 |

# Inferential Analysis

## 1. Look at the distribution of outage duration time in our dataframe.

In [24]:
```
sns.boxplot(x=outage['outage_duration'])
plt.xlabel('Duration Time', size=18)
```

Out[24]:
```
Text(0.5, 0, 'Duration Time')
```




Now, we can see that there is an extreme outlier in the distribution, where the duration time is greater than 100000 minutes (around 70 days). Let's dig deeper into that.

In [25]:
```
outlier = outage[outage['outage_duration'] == outage['outage_duration'].max()]
outlier
```

Out[25]:

| | year | month | u_s__state | climate_category | outage_start | outage_restoration | cause_category | cause_category_detail | outage_duration | res |
|---|---|---|---|---|---|---|---|---|---|---|
| **54** | 2014 | 1 | Wisconsin | cold | 2014-01-24 00:00:00 | 2014-04-09 11:53:00 | fuel supply emergency | Coal | 108653.0 | |

**If we google on that date and event, [https://www.twincities.com/2014/01/26/propane-emergency-declared-in-wisconsin-2/](https://www.twincities.com/2014/01/26/propane-emergency-declared-in-wisconsin-2/), we can see the "frigid weather bearing down on Wisconsin and 'dangerously low' supplies of propane". It makes more sense why this outage has last for such a long time.**

Before we carry out the analysis, we'll remove the outlier from the dataframe.

In [26]:
```
q1 = outage['outage_duration'].quantile(0.25)
q3 = outage['outage_duration'].quantile(0.75)
diff = q3-q1
outage = outage[(outage['outage_duration'] <= q3 + 1.5 * diff) & (outage['outage_duration'] >= q1 - 1.5 * diff)]
outage
```

Out[26]:

| | year | month | u_s__state | climate_category | outage_start | outage_restoration | cause_category | cause_category_detail | outage_duration |
|---|---|---|---|---|---|---|---|---|---|
| **1** | 2011 | 7 | Minnesota | normal | 2011-07-01 17:00:00 | 2011-07-03 20:00:00 | severe weather | NaN | 3060.0 |
| **2** | 2014 | 5 | Minnesota | normal | 2014-05-11 18:38:00 | 2014-05-11 18:39:00 | intentional attack | vandalism | 1.0 |
| **3** | 2010 | 10 | Minnesota | cold | 2010-10-26 20:00:00 | 2010-10-28 22:00:00 | severe weather | heavy wind | 3000.0 |
| **4** | 2012 | 6 | Minnesota | normal | 2012-06-19 04:30:00 | 2012-06-20 23:00:00 | severe weather | thunderstorm | 2550.0 |
| **5** | 2015 | 7 | Minnesota | warm | 2015-07-18 02:00:00 | 2015-07-19 07:00:00 | severe weather | NaN | 1740.0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **1526** | 2011 | 6 | Idaho | normal | 2011-06-15 16:00:00 | 2011-06-16 06:30:00 | intentional attack | vandalism | 870.0 |
| **1527** | 2016 | 3 | Idaho | warm | 2016-03-08 00:00:00 | 2016-03-08 00:00:00 | intentional attack | sabotage | 0.0 |
| **1530** | 2011 | 12 | North Dakota | cold | 2011-12-06 08:00:00 | 2011-12-06 20:00:00 | public appeal | NaN | 720.0 |
| **1532** | 2009 | 8 | South Dakota | warm | 2009-08-29 22:54:00 | 2009-08-29 23:53:00 | islanding | NaN | 59.0 |
| **1533** | 2009 | 8 | South | warm | 2009-08-29 | 2009-08-29 | islanding | NaN | 181.0 |

| | year | month | u_s__state | climate_category | outage_start | outage_restoration | cause_category | cause_category_detail | outage_duration |
|---|---|---|---|---|---|---|---|---|---|
| | | | Dakota | | 11:00:00 | 14:01:00 | | | |

1320 rows × 39 columns

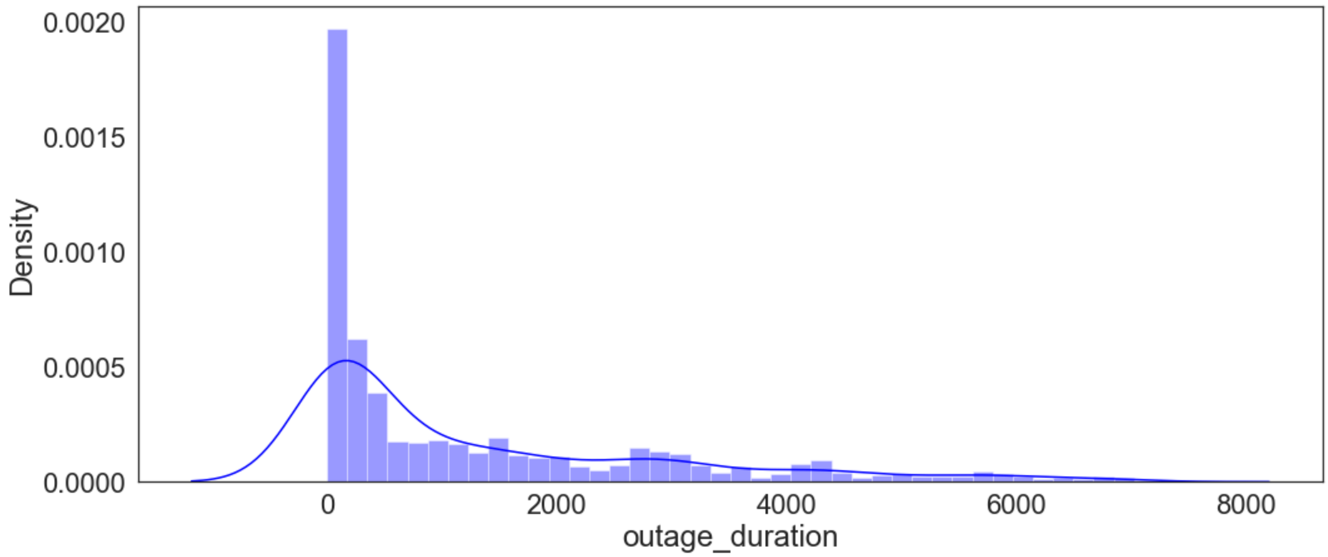
Let's take a brief look at the distribution of outage duration after removing the outlier.

In [27]:

```
sns.distplot(outage['outage_duration'],
             kde=True, bins=40, color="blue")
```

Out[27]:

```
<AxesSubplot:xlabel='outage_duration', ylabel='Density'>
```




The distributions of outage duration is skewed right. This suggests that we may need to transform these data to use linear regression to ensure that the large outlier values are not driving our relationship.

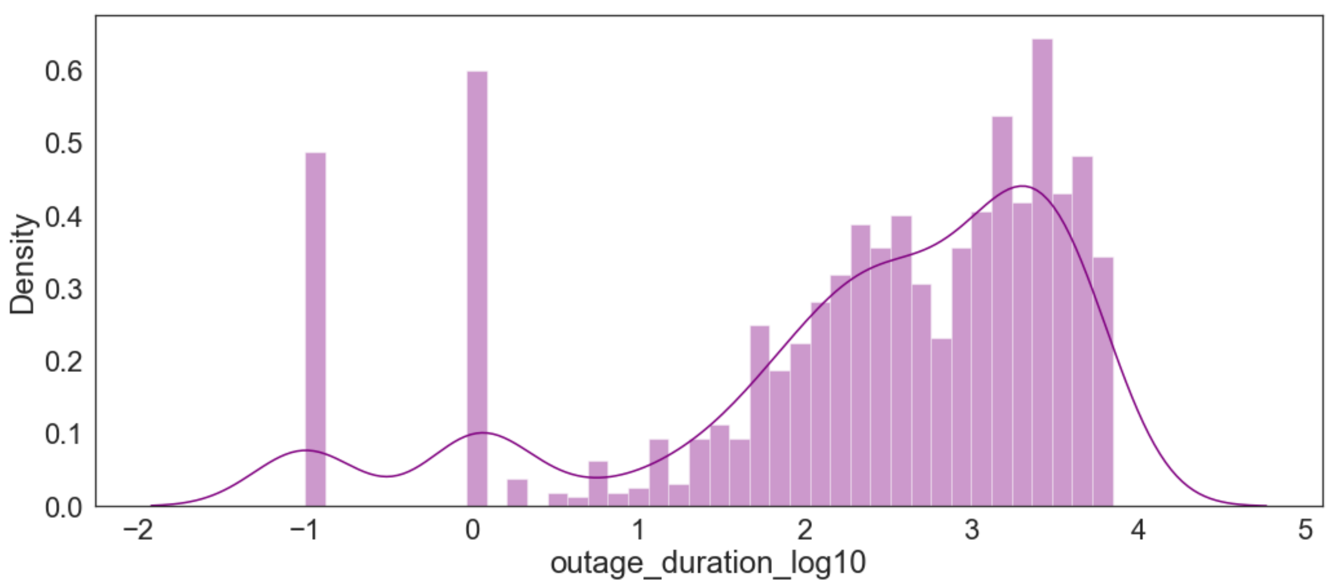
So, as we learned in class, we can apply a log10-transformation the outage duration data, with an offset of 0.1. This will shift the values away from being centered near zero when put on the log scale. Then, we store this in a new column outage_duration_log10. (We do that in the later linear regression)

In [28]:

```
outage['outage_duration_log10'] = np.log10(outage['outage_duration'] + 0.1)

sns.distplot(outage['outage_duration_log10'],
             kde=True, bins=40, color="purple")
```

Out[28]:

```
<AxesSubplot:xlabel='outage_duration_log10', ylabel='Density'>
```




The distribution seems less skewed, but there is a value around -1. This is because there were zeroes in the original dataset. Due to this we used an offset of 0.1 in the log transformation above. These zeroes all show up at -1 based on $log10(0+0.1)=-1$ .

Now let's use the transformed data to do the linear regression analysis.

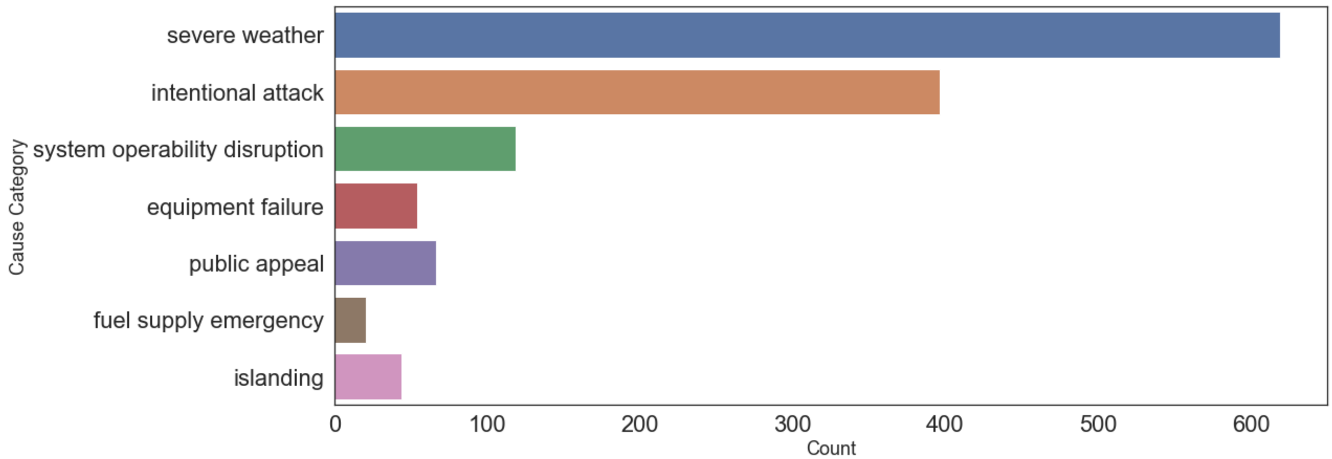
# 2. Find the relationship between the duration time of outage and different factors.

## 2.1 The number of Outage V.S. Duration time

### 2.1.1 outage cause

In [29]:
```
common_reason = sns.countplot(y='cause_category', data=outage)
plt.ylabel('Cause Category', size=18)
plt.xlabel('Count', size=18)
```

Out[29]:
```
Text(0.5, 0, 'Count')
```

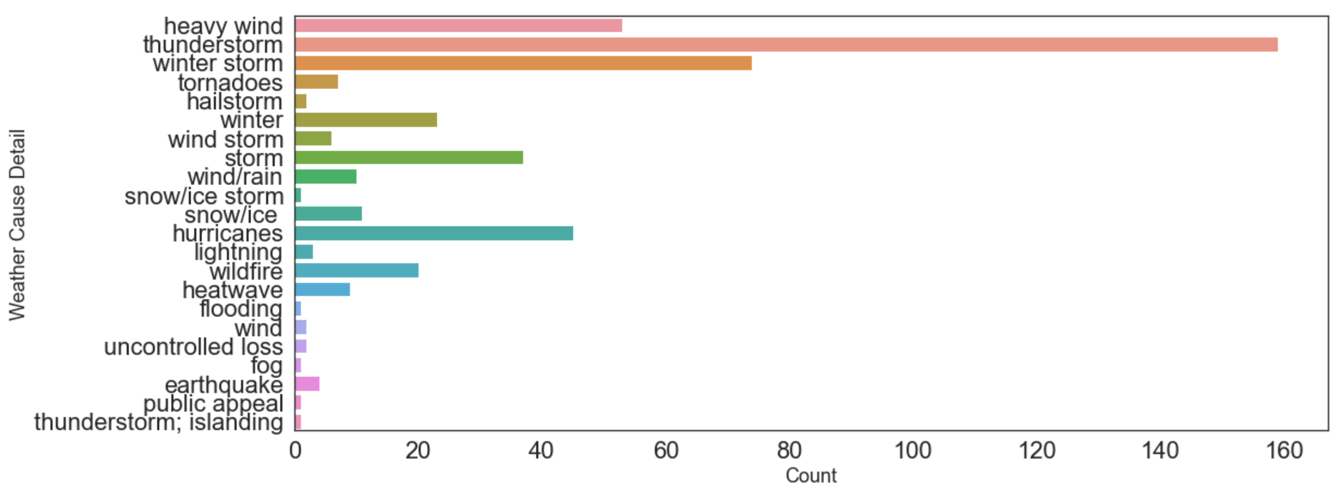



From the plot above, we can clearly tell the fact that the `severe weather`, `intentional attack` and `system operability disruption` are the most common cause of outage in the U.S., which also fits our background research.

Then let's find out what's the most common weather disaster that causes the outage.

In [30]:
```
common_weather = sns.countplot(y='cause_category_detail', data=outage.loc[outage['cause_category']=='severe weather']

plt.ylabel('Weather Cause Detail', size=18)
plt.xlabel('Count', size=18)
```

Out[30]:
```
Text(0.5, 0, 'Count')
```



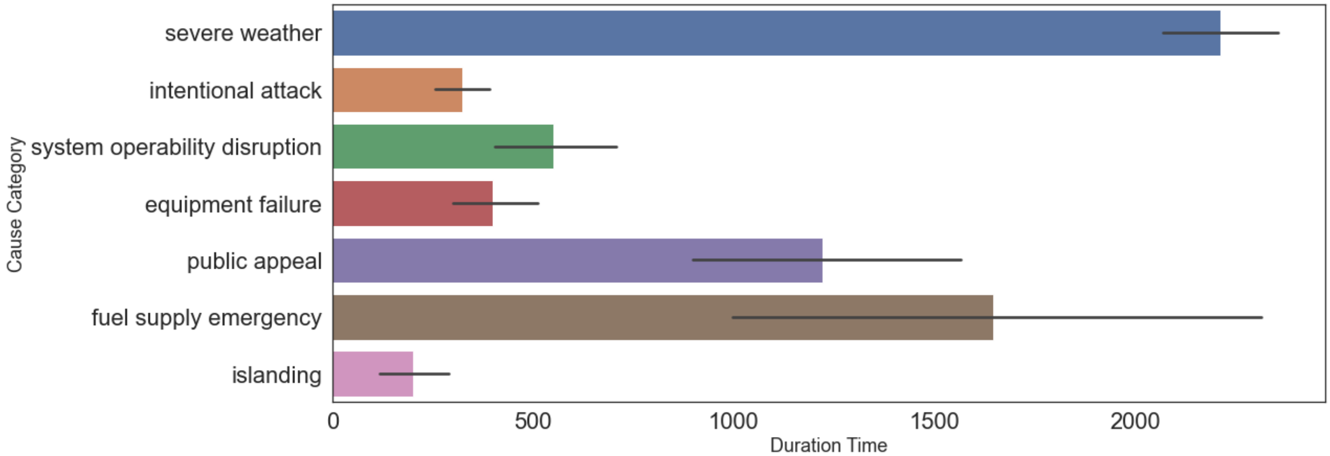

Looking into the `severe weather`, we can find that `thunderstorm`, `winter storm` and `heavy wind` are the top 3 factors in the category.

Use bar plot to represent the relationship between the outage duration time and different outage causes.

In [31]:
```
cause_TIME = sns.barplot(y='cause_category',
                         x='outage_duration',
                         data=outage)
```

```
plt.xlabel('Duration Time', size=18)
```

Out[31]: Text(0, 0.5, 'Cause Category')




From the generated plot above and the information about causes we get in part 1, we know that even though severe weather is the most common cause of power outage, outages due to fuel supply emergency occurs the least likely and usually last for a longer time.

From the first plot we generated above, `severe weather` is obviously the cause that has the most amount of power outage events. Now we want to check if the outage duration of severe weather is significantly smaller than the average duration time across the records. We'll do a z-test on this.

In [32]:
```
outage['outage_duration'].describe()
# mean = 1295.778030
ztest(outage[outage['cause_category']=='severe weather'].outage_duration, value=1295.778030)
```

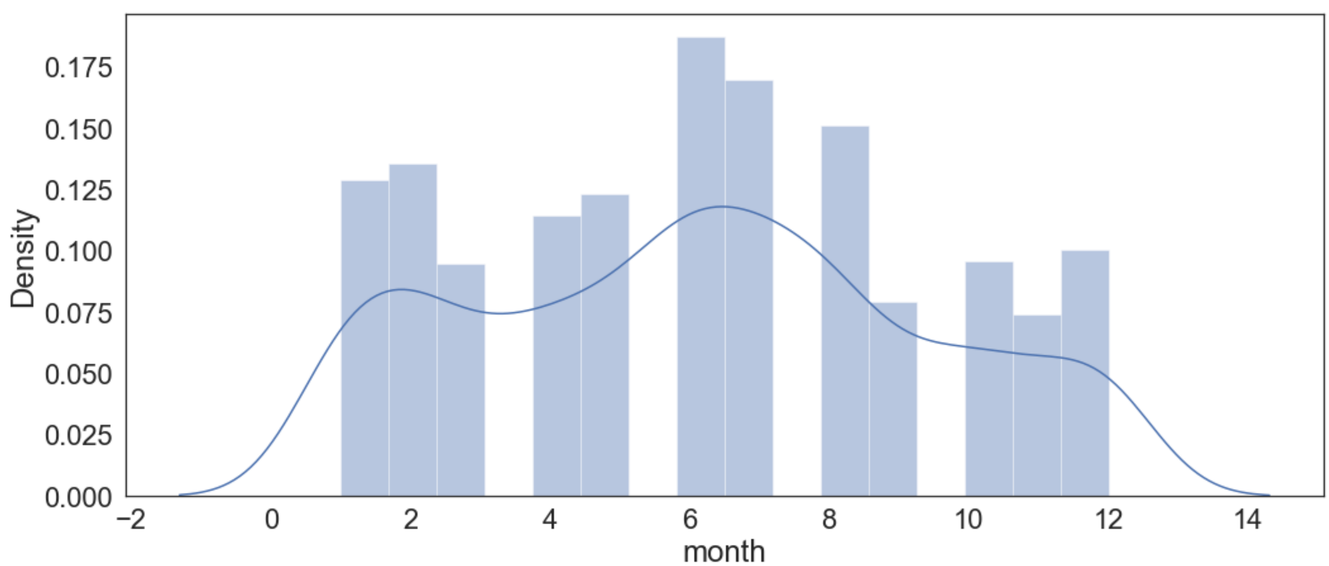
Out[32]: (12.457077191077335, 1.2797546769023965e-35)

Since `p-value` is around `1.28e-35` , which is obviously smaller than `0.1` , we have sufficient evidence to reject the null hypothesis. In other words, the `severe weather` does significantly affect `outage duration` .

### 2.1.2 MONTH

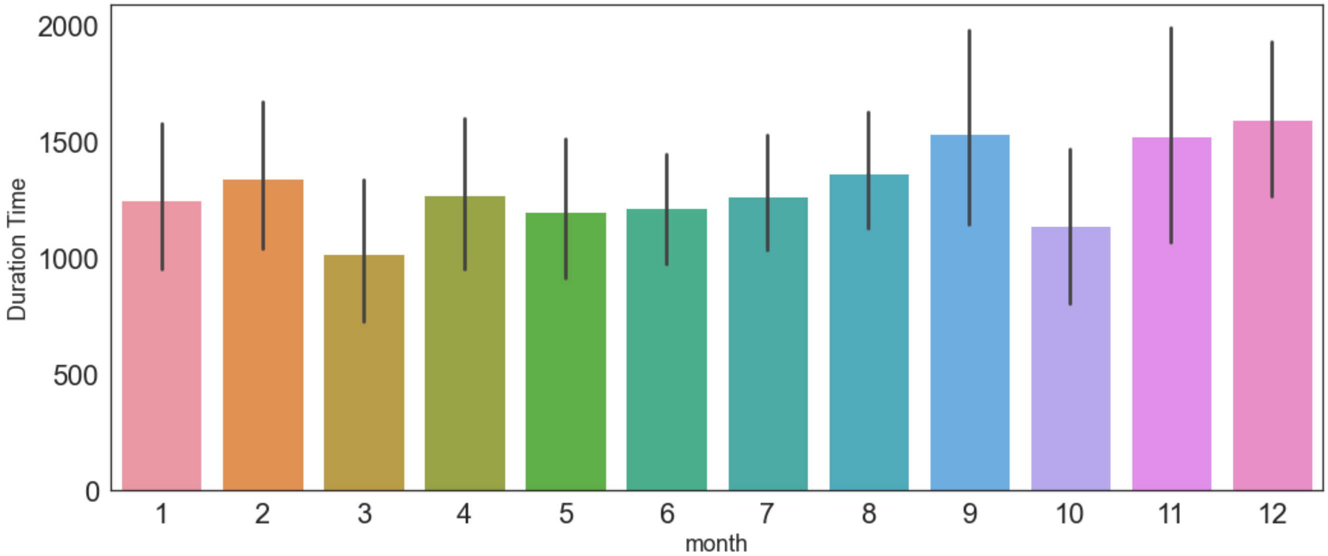
In [33]:
```
sns.distplot(outage['month'])
```

Out[33]: <AxesSubplot:xlabel='month', ylabel='Density'>




In [34]:
```
sns.barplot(x='month', y='outage_duration', data=outage)

plt.ylabel('Duration Time', size=18)
plt.xlabel('month', size=18)
```

Out[34]: Text(0.5, 0, 'month')




From the first plot we generated above, June is obviously the month that has the most amount of power outage events. Now we want to check if the outage duration of June is significantly smaller than the average duration time across the records. We'll do a z-test on this.

In [35]:
```
outage['outage_duration'].describe()
# mean = 1295.778030
ztest(outage[outage['month']==6].outage_duration, value=1295.778030)
```

Out[35]:
```
(-0.6300552996085758, 0.5286584042104481)
```

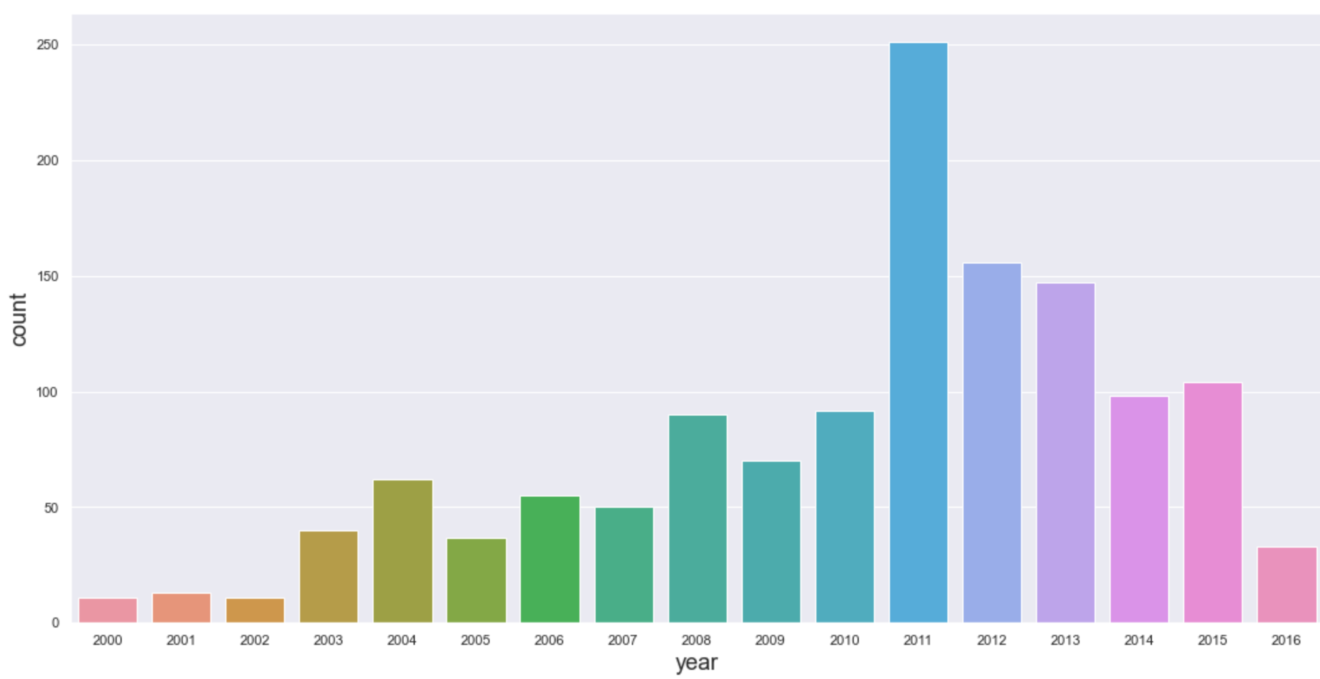
Since `p-value` is around `0.53`, which is obviously greater than `0.1`, we do not have sufficient evidence to reject the null hypothesis. In other words, the `month` does not significantly affect `outage duration`.

### 2.1.3 YEAR

Use a countplot to see which years have the most power outage events.

In [36]:
```
outage['outage_duration'] = outage['outage_duration'].astype(float)
sns.set(rc={'figure.figsize':(18,9)})
sns.countplot(x='year', data=outage)

plt.ylabel('count', size=18)
plt.xlabel('year', size=18)
```

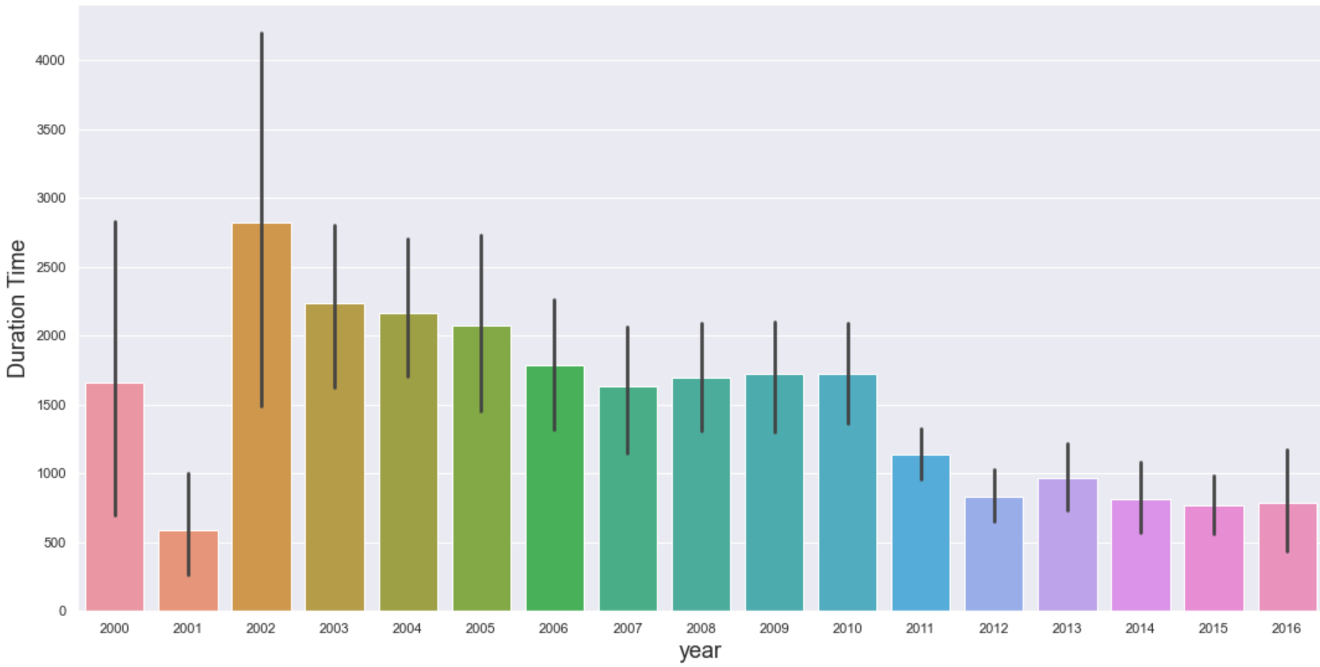
Out[36]:
```
Text(0.5, 0, 'year')
```




Then we use a bar plot to have a look at the distribution of the outage duration for different years.

In [37]:
```
sns.barplot(x='year', y='outage_duration', data=outage)

plt.ylabel('Duration Time', size=18)
plt.xlabel('year', size=18)
```

Out[37]:
```
Text(0.5, 0, 'year')
```




From the first plot we generated above, `2011` is obviously the year that has the most amount of power outage events. Now we want to check if the outage duration of 2011 is significantly smaller than the average duration time across the records. We'll do a z-test on this.

In [38]:
```
outage['outage_duration'].describe()
# mean = 1295.778030
ztest(outage[outage['year']==2011].outage_duration, value=1295.778030)
```

Out[38]:
```
(-1.695895878580138, 0.08990560032514401)
```

Since `p-value` is around `0.09`, which is smaller than `0.1`, we can conclude that the duration time of 2011 is significantly shorter than average duration time from 2000-2016.

## Setup for Sales Price

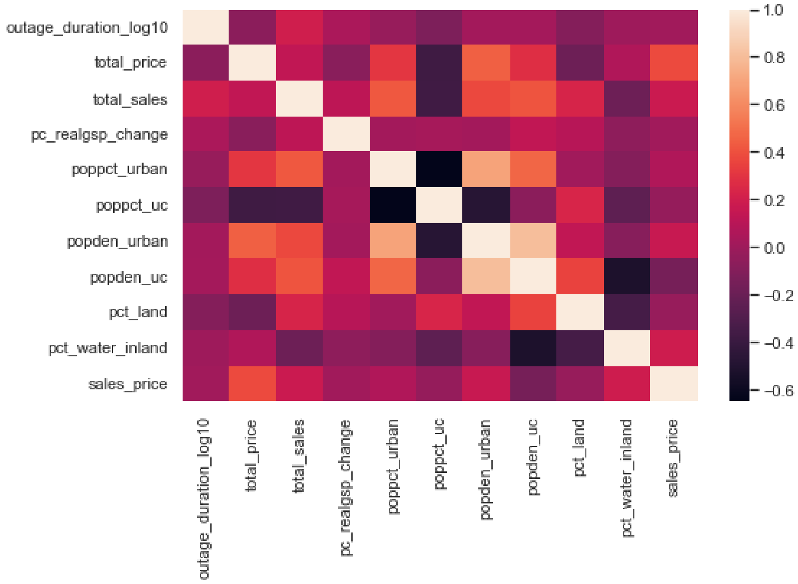
In [39]:
```
outage['sales_price'] = (outage['total_sales'] * outage['total_price'] * 1000)/outage['total_customers']
outage['sales_price']
```

Out[39]:
```
1       23461.986920
2       18569.688568
3       16452.187228
4       20401.585446
5       23291.533096
            ...
1526    16401.897802
1527    15702.872742
1530    25181.431969
1532    16247.134349
1533    16247.134349
Name: sales_price, Length: 1320, dtype: float64
```

## *Overview before actually analyzing the numeric factors

In [40]:
```
outage_corr = outage[['outage_duration_log10','total_price','total_sales','pc_realgsp_change','poppct_urban','poppct_
fig, ax = plt.subplots(figsize=(8, 5))
sns.heatmap(outage_corr.corr())
```

Out[40]:
```
<AxesSubplot:>
```




From the heatmap, we can see that the relation between outage_duration(after log10 transform) and other numeric factors are not quite obvious. However, they are related at some small values, so that we will dig more deeply into each factors.

## 2.2 Electricity Consumption & Electricity Price V.S. Duration time

In [41]:

```
sns.lmplot(y = 'outage_duration_log10',
           x = 'sales_price',
           data = outage,
           fit_reg = True,
           height = 6,
           aspect = 1.7,
           line_kws={'color': 'green'},
           scatter_kws={'color': 'green'})
```

Out[41]: <seaborn.axisgrid.FacetGrid at 0x26401935250>


In [42]:

```
duration_priceSale = outage[['sales_price', 'outage_duration_log10']]
duration_priceSale.columns = ['SP', 'duration_log10']

outcome, predictors = patsy.dmatrices('duration_log10 ~ SP', duration_priceSale)

# Now use statsmodels to intialize an OLS linear model
# This step initializes the model, and provides the data (but does not actually compute the model)
mod_log = sm.OLS(outcome, predictors)

# fit the model
res_log = mod_log.fit()
```

```
# Check out the results
print(res_log.summary())
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:         duration_log10   R-squared:                       0.000
Model:                            OLS   Adj. R-squared:                 -0.001
Method:                 Least Squares   F-statistic:                    0.1080
Date:                Tue, 07 Jun 2022   Prob (F-statistic):              0.743
Time:                        03:23:41   Log-Likelihood:                -2214.6
No. Observations:                1320   AIC:                             4433.
Df Residuals:                    1318   BIC:                             4444.
Df Model:                           1                                         
Covariance Type:            nonrobust                                         
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      2.2843      0.144     15.850      0.000       2.002       2.567
SP          2.185e-06   6.65e-06      0.329      0.743   -1.09e-05    1.52e-05
==============================================================================
Omnibus:                      223.446   Durbin-Watson:                   1.552
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              344.614
Skew:                          -1.212   Prob(JB):                     1.47e-75
Kurtosis:                       3.622   Cond. No.                     8.75e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 8.75e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

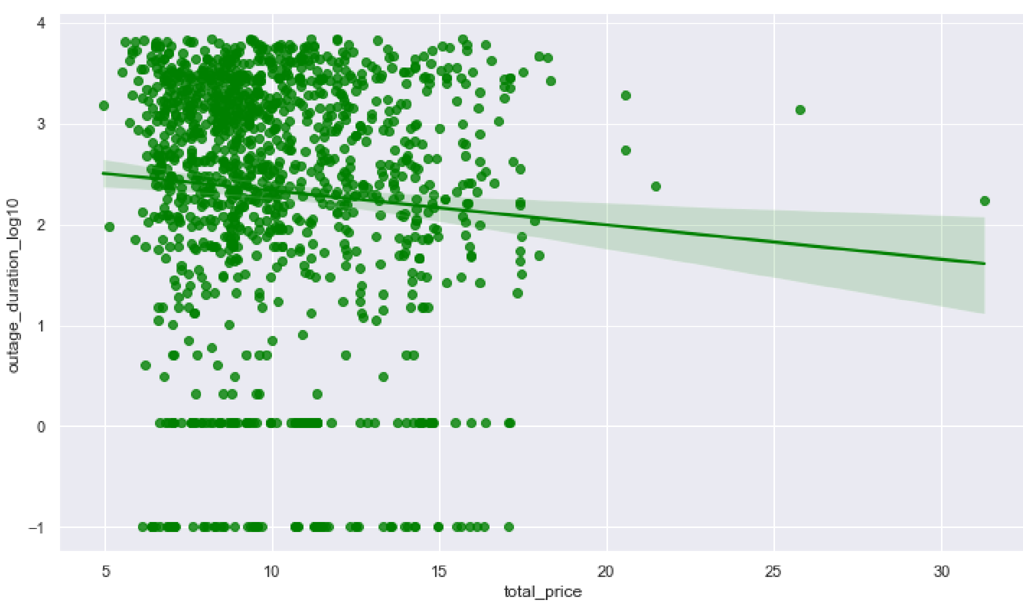
Summary of `Total electricity consumption in the U.S. state` V.S. `Duration time`

- The `Cond. No. is 8.75e+04`, which is a collinearity problem.
- the `p-value is 0.743 > 0.05`, which suggests that we lack evidence to reject the null hypothesis thus cannot make inferential analysis.
- the `slope of sales_price is 2.185e-06`, which suggest very slight proportional relationship between `Total electricity consumption in the U.S. state` V.S. `Duration`
- the `R-squared is 0.000`, which means 0% of the data variability is explained by the regression model.

Overall, we **cannot reject the null hypothesis** since p-value is too high, and we conclude there is a **collinearity probelm between** `Total electricity consumption` V.S. `Duration time`, but the data `doesn't fit the model`.

In [43]:
```
sns.lmplot(y = 'outage_duration_log10',
           x = 'total_price',
           data = outage,
           fit_reg = True,
           height = 6,
           aspect = 1.7,
           line_kws={'color': 'green'},
           scatter_kws={'color': 'green'})
```

Out[43]: `<seaborn.axisgrid.FacetGrid at 0x2640191aeb0>`



In [44]:
```
duration_price = outage[['total_price', 'outage_duration_log10']]
duration_price.columns = ['P', 'duration_log10']

outcome, predictors = patsy.dmatrices('duration_log10 ~ P', duration_price)

# Now use statsmodels to intialize an OLS linear model
# This step initializes the model, and provides the data (but does not actually compute the model)
mod_log = sm.OLS(outcome, predictors)

# fit the model
res_log = mod_log.fit()

# Check out the results
print(res_log.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:         duration_log10   R-squared:                       0.006
Model:                            OLS   Adj. R-squared:                  0.005
Method:                 Least Squares   F-statistic:                     7.496
Date:                Tue, 07 Jun 2022   Prob (F-statistic):            0.00627
Time:                        03:23:42   Log-Likelihood:                -2210.9
No. Observations:                1320   AIC:                             4426.
Df Residuals:                    1318   BIC:                             4436.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      2.6740      0.131     20.487      0.000       2.418       2.930
P             -0.0340      0.012     -2.738      0.006      -0.058      -0.010
==============================================================================
Omnibus:                      225.790   Durbin-Watson:                   1.550
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              349.636
Skew:                          -1.217   Prob(JB):                     1.20e-76
Kurtosis:                       3.657   Cond. No.                         38.9
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Summary of `Average monthly electricity price` V.S. `Duration`

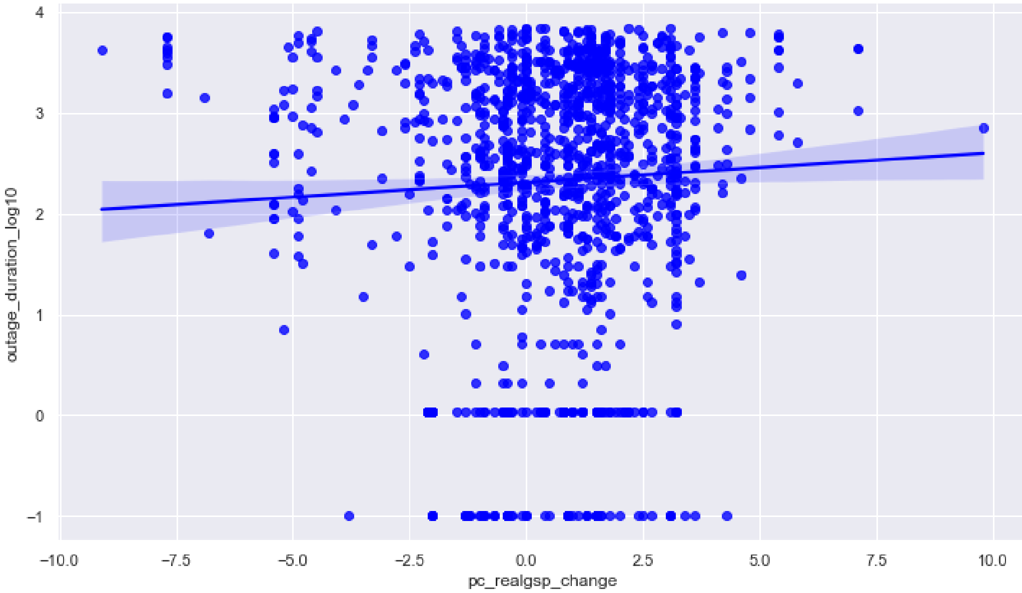
- The `Cond. No. is 38.9`, which is in a reasonable range.
- the `p-value is 0.006 < 0.05`, which suggest that these results happen due to random chance alone is approximately 0.6% of the time, which makes is inferential.
- the `slope of total_price is -0.0340`, which suggest very slight negative peropertional relationshion between `Percentage change of per capita real GSP` V.S. `Duration`
- the `R-squared is 0.006`, which means 0.6% of the data variability is explained by the regression model.

Overall, we conclude there is a `negative proportional relationship` between `Average monthly electricity price` V.S. `Duration`, but the data `doesn't fit into the model well`.

## 2.3 Percentage change of per capita real GSP V.S. Duration

In [45]:
```
sns.lmplot(y = 'outage_duration_log10',
           x = 'pc_realgsp_change',
           data = outage,
           fit_reg = True,
           height = 6,
           aspect = 1.7,
           line_kws={'color': 'blue'},
           scatter_kws={'color': 'blue'})
```

Out[45]:
```
<seaborn.axisgrid.FacetGrid at 0x26401c20b20>
```




In [46]:

```
duration_price = outage[['pc_realgsp_change', 'outage_duration_log10']]
duration_price.columns = ['PRC', 'duration_log10']

outcome, predictors = patsy.dmatrices('duration_log10 ~ PRC', duration_price)

# Now use statsmodels to intialize an OLS linear model
# This step initializes the model, and provides the data (but does not actually compute the model)
mod_log = sm.OLS(outcome, predictors)

# fit the model
res_log = mod_log.fit()

# Check out the results
print(res_log.summary())
```

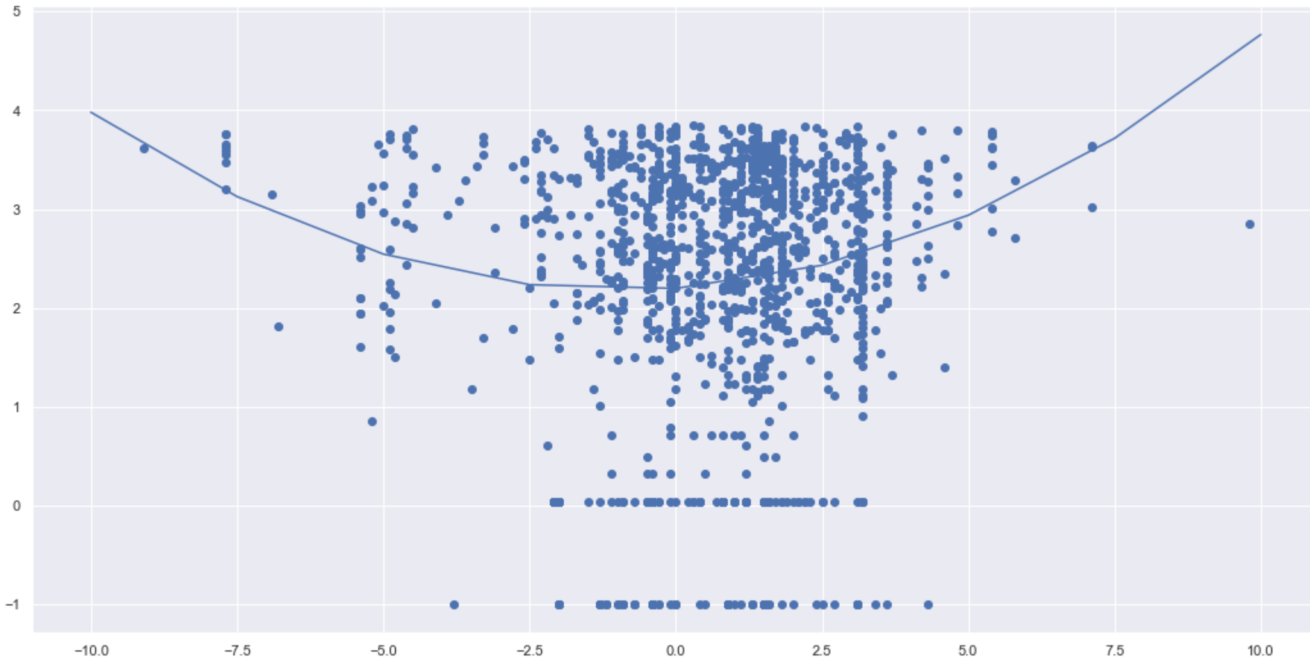
```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:         duration_log10   R-squared:                       0.002
Model:                            OLS   Adj. R-squared:                  0.002
Method:                 Least Squares   F-statistic:                     2.988
Date:                Tue, 07 Jun 2022   Prob (F-statistic):             0.0841
Time:                        03:23:42   Log-Likelihood:                -2213.1
No. Observations:                1320   AIC:                             4430.
Df Residuals:                    1318   BIC:                             4441.
Df Model:                           1                                         
Covariance Type:            nonrobust                                         
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      2.3103      0.037     61.692      0.000       2.237       2.384
PRC            0.0293      0.017      1.728      0.084      -0.004       0.063
==============================================================================
Omnibus:                      220.415   Durbin-Watson:                   1.561
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              338.053
Skew:                          -1.202   Prob(JB):                     3.92e-74
Kurtosis:                       3.609   Cond. No.                         2.37
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Summary of `Percentage change of per capita real GSP` V.S. `Duration`

- The `Cond. No. is 2.37`, which is in a reasonable range.
- the `p-value is 0.084 > 0.05`, which suggest that these results happen due to random chance alone is approximately 8.4% of the time, which makes it too high to have inferential meaning.
- the `slope of pc_realgsp_change is 0.0293`, which suggest very slight proportional relationship between `Percentage change of per capita real GSP` V.S. `Duration` in theory.
- the `R-squared is 0.002`, which means 0.2% of the data variability is explained by the regression model.

Overall, we conclude there is no relationship between `Percentage change of per capita real GSP` V.S. `Duration`.

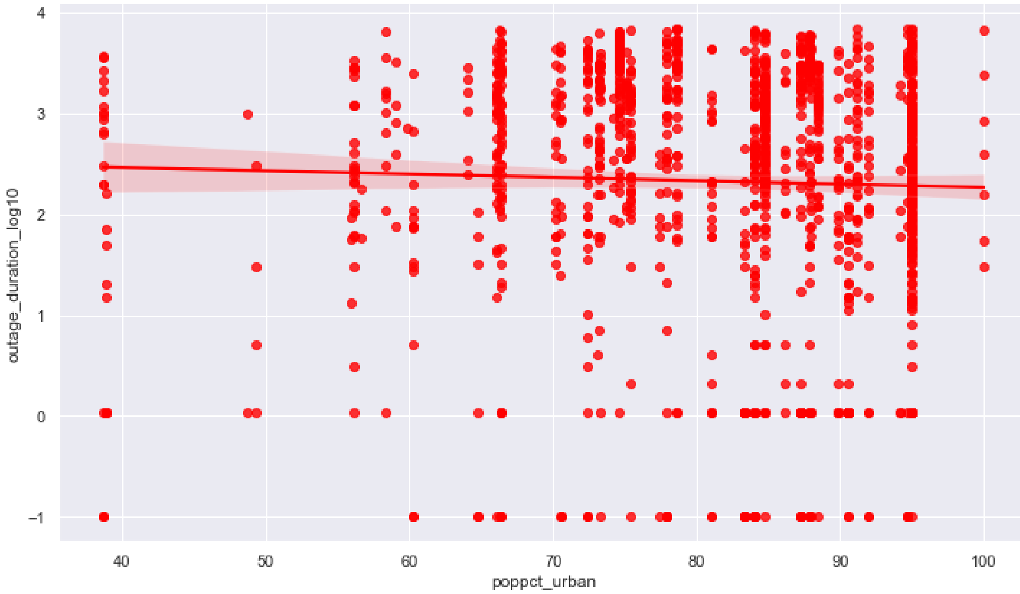
In [47]:
```
model = np.poly1d(np.polyfit(outage['pc_realgsp_change'], outage['outage_duration_log10'], 2))
polyline = np.linspace(-10, 10, 9)
plt.scatter(outage['pc_realgsp_change'],outage['outage_duration_log10'])
plt.plot(polyline, model(polyline))
plt.show()
```

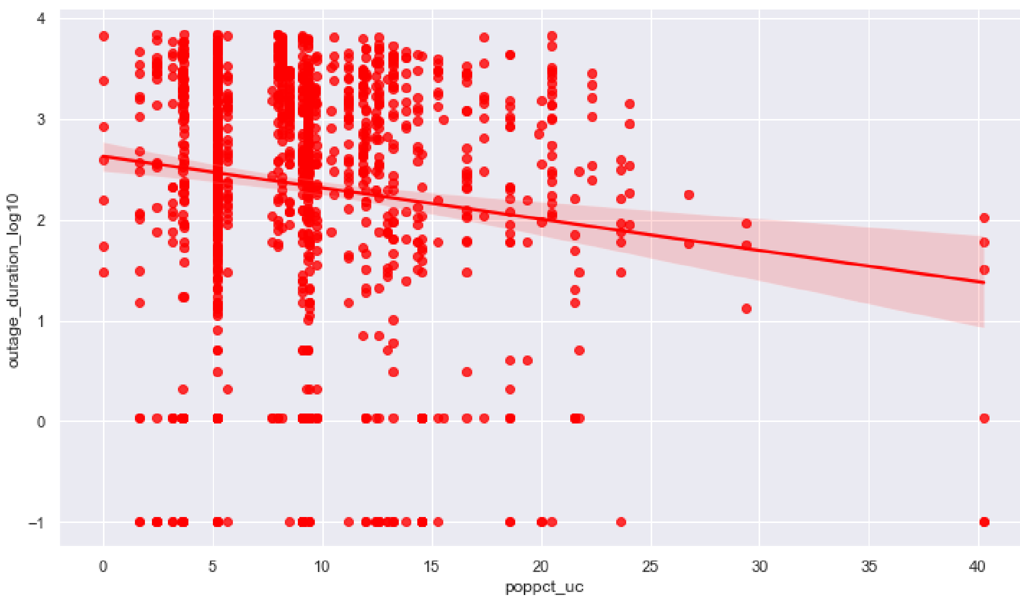



## 2.4 Percentage of urbanization V.S. Duration

In [48]:
```
sns.lmplot(y = 'outage_duration_log10',
           x = 'poppct_urban',
           data = outage,
           fit_reg = True,
           height = 6,
           aspect = 1.7,
           line_kws={'color': 'red'},
           scatter_kws={'color': 'red'})
sns.lmplot(y = 'outage_duration_log10',
           x = 'poppct_uc',
           data = outage,
           fit_reg = True,
           height = 6,
           aspect = 1.7,
           line_kws={'color': 'red'},
           scatter_kws={'color': 'red'})
```

Out[48]: `<seaborn.axisgrid.FacetGrid at 0x26401c98490>`




In [49]:
```
duration_price = outage[['poppct_urban', 'outage_duration_log10']]
duration_price.columns = ['PU', 'duration_log10']

outcome, predictors = patsy.dmatrices('duration_log10 ~ PU', duration_price)

# Now use statsmodels to intialize an OLS linear model
# This step initializes the model, and provides the data (but does not actually compute the model)
mod_log = sm.OLS(outcome, predictors)

# fit the model
res_log = mod_log.fit()

# Check out the results
print(res_log.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:         duration_log10   R-squared:                       0.001
Model:                            OLS   Adj. R-squared:                  0.000
Method:                 Least Squares   F-statistic:                     1.245
Date:                Tue, 07 Jun 2022   Prob (F-statistic):              0.265
Time:                        03:23:44   Log-Likelihood:                -2214.0
No. Observations:                1320   AIC:                             4432.
Df Residuals:                    1318   BIC:                             4442.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      2.5977      0.242     10.717      0.000       2.122       3.073
```

```
PU             -0.0033      0.003     -1.116      0.265      -0.009       0.003
==============================================================================
Omnibus:                      224.908   Durbin-Watson:                   1.553
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              347.782
Skew:                          -1.217   Prob(JB):                     3.02e-76
Kurtosis:                       3.633   Cond. No.                         556.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

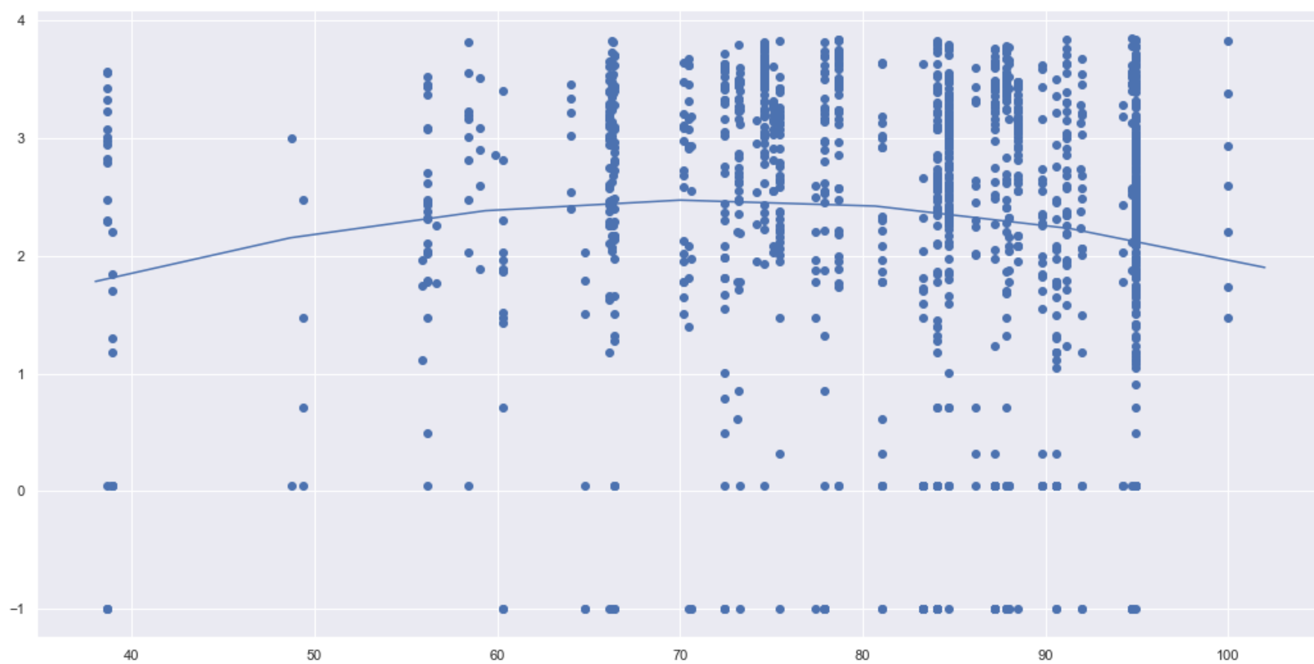
Summary of `Density of population in urban` V.S. `Duration`

- The `Cond. No. is 556`, which is in a reasonable range.
- the `p-value is 0.265 > 0.05`, which suggest that these results happen due to random chance alone is approximately 26.5% of the time, which is inferential meaning.
- the `slope of poppct_urban is -0.0033`, which suggest very slight negative proportional relationship between `Density of population in urban` V.S. `Duration` in theory.
- the `R-squared is 0.002`, which means 0.2% of the data variability is explained by the regression model.

Overall, we conclude there is no relationship between `Density of population in urban` V.S. `Duration`.

In [50]:

```
model = np.poly1d(np.polyfit(outage['poppct_urban'], outage['outage_duration_log10'], 2))
polyline = np.linspace(38, 102, 7)
plt.scatter(outage['poppct_urban'],outage['outage_duration_log10'])
plt.plot(polyline, model(polyline))
plt.show()
```




In [51]:

```
duration_price = outage[['poppct_uc', 'outage_duration_log10']]
duration_price.columns = ['PUC', 'duration_log10']

outcome, predictors = patsy.dmatrices('duration_log10 ~ PUC', duration_price)

# Now use statsmodels to intialize an OLS linear model
# This step initializes the model, and provides the data (but does not actually compute the model)
mod_log = sm.OLS(outcome, predictors)

# fit the model
res_log = mod_log.fit()

# Check out the results
print(res_log.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:         duration_log10   R-squared:                       0.016
Model:                            OLS   Adj. R-squared:                  0.015
Method:                 Least Squares   F-statistic:                     21.22
Date:                Tue, 07 Jun 2022   Prob (F-statistic):           4.48e-06
Time:                        03:23:44   Log-Likelihood:                -2204.1
```

```
No. Observations:                1320   AIC:                             4412.
Df Residuals:                    1318   BIC:                             4423.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      2.6278      0.074     35.668      0.000       2.483       2.772
PUC           -0.0311      0.007     -4.607      0.000      -0.044      -0.018
==============================================================================
Omnibus:                      224.296   Durbin-Watson:                   1.578
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              346.371
Skew:                          -1.212   Prob(JB):                     6.12e-76
Kurtosis:                       3.650   Cond. No.                         22.8
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Summary of `Density of population in urban clusters` V.S. `Duration`

- The `Cond. No. is 22.8`, which is in reasonable range.
- the `p-value is 0.000 < 0.05`, which suggest that these results happen due to random chance alone is approximately 0% of the time, which is inferential meaning.
- the `slope of poppct_uc is -0.0311`, which suggest very slight negative proportional relationship between `Density of population in urban clusters` V.S. `Duration`
- the `R-squared is 0.016`, which means 1.6% of the data variability is explained by the regression model.

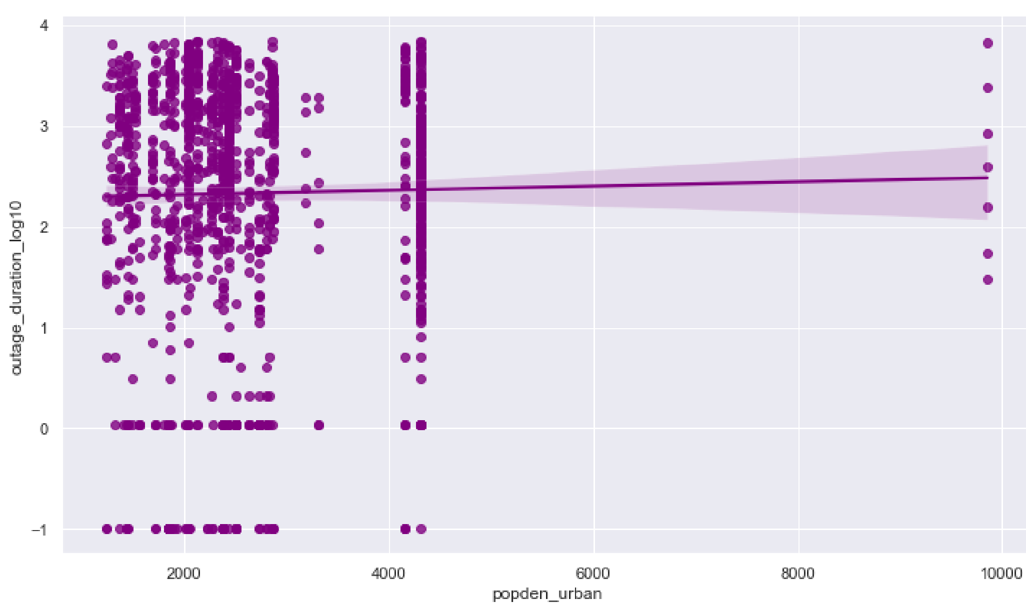
Overall, we conclude there is a negative proportional relationship between `Density of population in urban clusters` V.S. `Duration`, but the data `doesn't fit into the model well`.

## 2.5 Density of urban population V.S. Duration

In [52]:

```python
sns.lmplot(y = 'outage_duration_log10',
           x = 'popden_urban',
           data = outage,
           fit_reg = True,
           height = 6,
           aspect = 1.7,
           line_kws={'color': 'purple'},
           scatter_kws={'color': 'purple'})
sns.lmplot(y = 'outage_duration_log10',
           x = 'popden_uc',
           data = outage,
           fit_reg = True,
           height = 6,
           aspect = 1.7,
           line_kws={'color': 'purple'},
           scatter_kws={'color': 'purple'})
```

Out[52]: <seaborn.axisgrid.FacetGrid at 0x26404062160>




In [53]:
```
duration_price = outage[['popden_urban', 'outage_duration_log10']]
duration_price.columns = ['PDU', 'duration_log10']

outcome, predictors = patsy.dmatrices('duration_log10 ~ PDU', duration_price)

# Now use statsmodels to intialize an OLS linear model
# This step initializes the model, and provides the data (but does not actually compute the model)
mod_log = sm.OLS(outcome, predictors)

# fit the model
res_log = mod_log.fit()

# Check out the results
print(res_log.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:          duration_log10   R-squared:                       0.000
Model:                             OLS   Adj. R-squared:                 -0.000
Method:                  Least Squares   F-statistic:                    0.3934
Date:                 Tue, 07 Jun 2022   Prob (F-statistic):              0.531
Time:                         03:23:45   Log-Likelihood:                -2214.4
No. Observations:                 1320   AIC:                             4433.
Df Residuals:                     1318   BIC:                             4443.
Df Model:                            1
Covariance Type:             nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      2.2755      0.094     24.174      0.000       2.091       2.460
PDU          2.12e-05   3.38e-05      0.627      0.531   -4.51e-05    8.75e-05
==============================================================================
Omnibus:                      220.950   Durbin-Watson:                   1.551
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              339.262
Skew:                          -1.205   Prob(JB):                     2.14e-74
Kurtosis:                       3.601   Cond. No.                     7.35e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 7.35e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Summary of `Density of urban population` V.S. `Duration`

- The `Cond. No. is 7.35e+03`, which is a collinearity problem.
- the `p-value is 0.531 > 0.05`, which suggest that these results happen due to random chance alone is approximately 53.1% of the time, which makes it too high to have inferential meaning.
- the `slope of PDU is 2.12e-05`, which suggest very slight propertional relationship between `Density of urban population` V.S. `Duration`
- the `R-squared is 0.000`, which means 0% of the data variability is explained by the regression model.

Due to the large value of Cond. No., this analysis suggest a collinearity problem between `Density of urban population` V.S. `Duration`.

In [54]:

```
model = np.poly1d(np.polyfit(outage['popden_urban'], outage['outage_duration_log10'], 2))
polyline = np.linspace(0, 10000, 5)
plt.scatter(outage['popden_urban'],outage['outage_duration_log10'])
plt.plot(polyline, model(polyline))
plt.show()
```


In [55]:

```
duration_price = outage[['popden_uc', 'outage_duration_log10']]
duration_price.columns = ['PDU', 'duration_log10']

outcome, predictors = patsy.dmatrices('duration_log10 ~ PDU', duration_price)

# Now use statsmodels to intialize an OLS linear model
# This step initializes the model, and provides the data (but does not actually compute the model)
mod_log = sm.OLS(outcome, predictors)

# fit the model
res_log = mod_log.fit()

# Check out the results
print(res_log.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:         duration_log10   R-squared:                       0.001
Model:                            OLS   Adj. R-squared:                 -0.000
Method:                 Least Squares   F-statistic:                    0.8704
Date:                Tue, 07 Jun 2022   Prob (F-statistic):              0.351
Time:                        03:23:45   Log-Likelihood:                -2204.5
No. Observations:                1313   AIC:                             4413.
Df Residuals:                    1311   BIC:                             4423.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      2.1654      0.179     12.116      0.000       1.815       2.516
PDU            0.0001      0.000      0.933      0.351      -0.000       0.000
==============================================================================
Omnibus:                      217.081   Durbin-Watson:                   1.549
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              331.694
Skew:                          -1.197   Prob(JB):                     9.41e-73
Kurtosis:                       3.578   Cond. No.                     7.92e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 7.92e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Summary of `Density of urban population` V.S. `Duration`

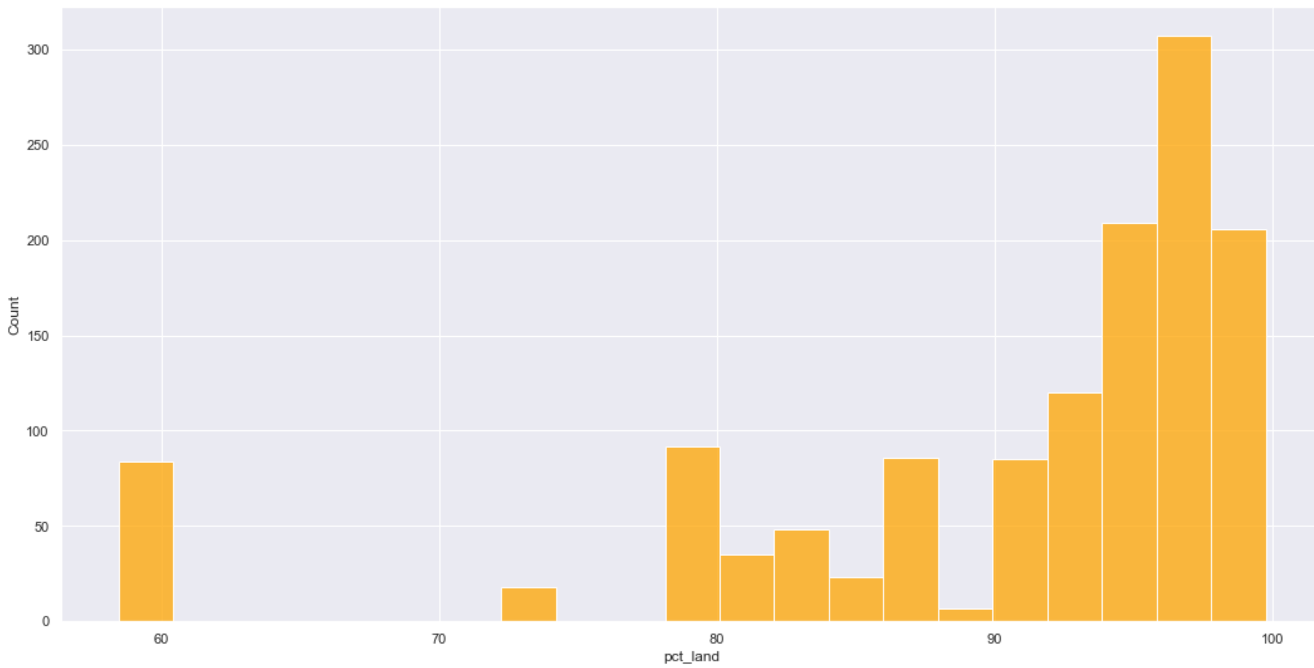
- The `Cond. No. is 7.92e+03`, which is a collinearity problem.
- the `p-value is 0.531 > 0.05`, which suggest that these results happen due to random chance alone is approximately 53.1% of the time, which makes it too high to have inferential meaning.
- the `slope of PDU is 0.0001`, which suggest very slight proportional relationship between `Density of urban population` V.S. `Duration`
- the `R-squared is 0.001`, which means 0.1% of the data variability is explained by the regression model.

Overall, thers is a collinearity problem in this analysis, and the data `doesn't fit into the model well`.

## 2.6 Land Area V.S. Duration

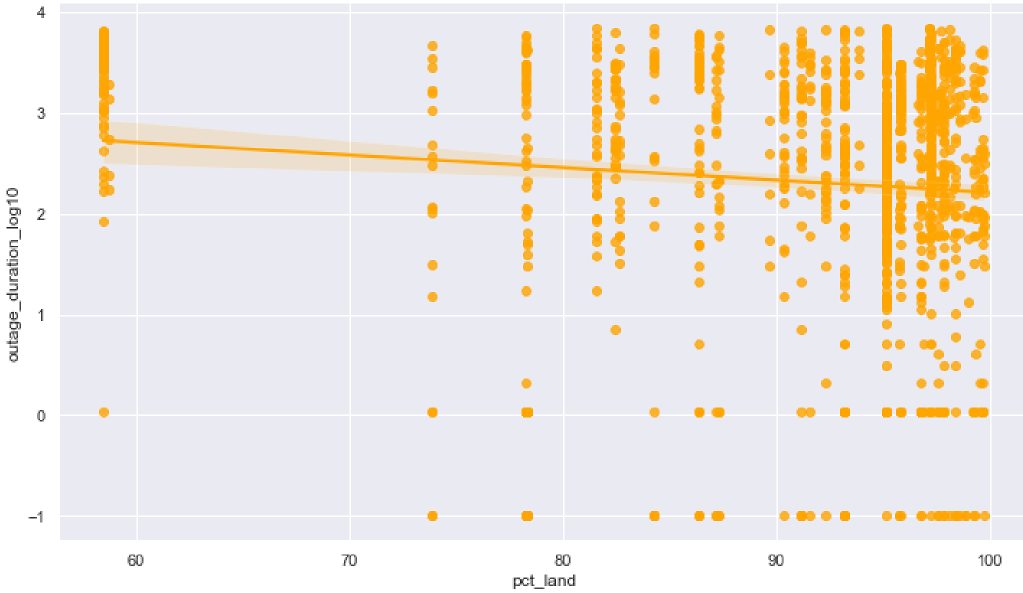
In [56]:
```
sns.histplot(outage['pct_land'], color = 'orange')
```

Out[56]: <AxesSubplot:xlabel='pct_land', ylabel='Count'>




In [57]:
```
sns.lmplot(y = 'outage_duration_log10',
           x = 'pct_land',
           data = outage,
           fit_reg = True,
           height = 6,
           aspect = 1.7,
           line_kws={'color': 'orange'},
           scatter_kws={'color': 'orange'})
```

Out[57]: <seaborn.axisgrid.FacetGrid at 0x264041f9b50>




In [58]:

```
duration_price = outage[['pct_land', 'outage_duration_log10']]
duration_price.columns = ['PL', 'duration_log10']

outcome, predictors = patsy.dmatrices('duration_log10 ~ PL', duration_price)

# Now use statsmodels to intialize an OLS linear model
# This step initializes the model, and provides the data (but does not actually compute the model)
mod_log = sm.OLS(outcome, predictors)

# fit the model
res_log = mod_log.fit()

# Check out the results
print(res_log.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:         duration_log10   R-squared:                       0.010
Model:                            OLS   Adj. R-squared:                  0.009
Method:                 Least Squares   F-statistic:                     13.45
Date:                Tue, 07 Jun 2022   Prob (F-statistic):           0.000254
Time:                        03:23:46   Log-Likelihood:                -2207.9
No. Observations:                1320   AIC:                             4420.
Df Residuals:                    1318   BIC:                             4430.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      3.4527      0.308     11.207      0.000       2.848       4.057
PL            -0.0124      0.003     -3.668      0.000      -0.019      -0.006
==============================================================================
Omnibus:                      240.248   Durbin-Watson:                   1.567
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              382.069
Skew:                          -1.263   Prob(JB):                     1.08e-83
Kurtosis:                       3.749   Cond. No.                         789.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Summary of `PCT_LAND` V.S. `Duration`

- The `Cond. No. is 789`, which is a reasonable range.
- the `p-value is 0.000 < 0.05`, which suggest that these results happen due to random chance alone is approximately 0% of the time, which is inferential.
- the `slope of PCT_LAND is -0.0124`, which suggest very slight negative propertional relationship between `PCT_LAND` V.S. `Duration`
- the `R-squared is 0.010`, which means 1% of the data variability is explained by the regression model.

Overall, the analysis suggest there is a negative propertional relationship between `PCT_LAND` V.S. `Duration`, but the data `doesn't fit the model well`.

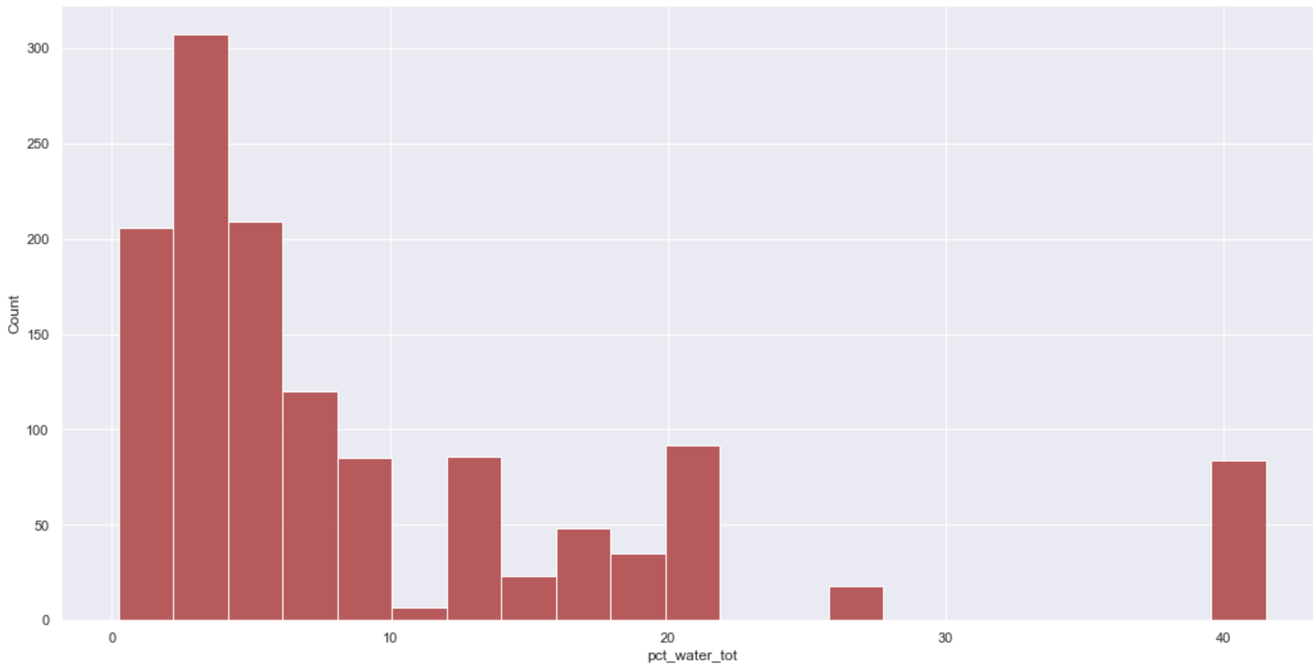
## 2.7 Water Area V.S. Duration

In [59]:

```
sns.histplot(outage['pct_water_tot'], color = "brown")
```

Out[59]:

```
<AxesSubplot:xlabel='pct_water_tot', ylabel='Count'>
```

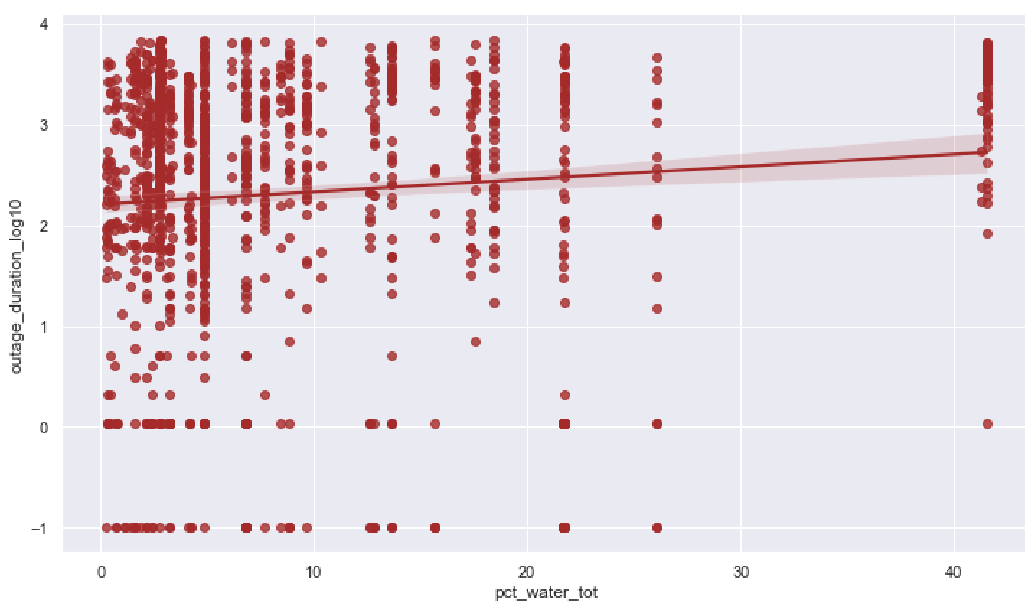



In [60]:

```
sns.lmplot(y = 'outage_duration_log10',
           x = 'pct_water_tot',
           data = outage,
           fit_reg = True,
           height = 6,
           aspect = 1.7,
           line_kws={'color': 'brown'},
           scatter_kws={'color': 'brown'})
```

Out[60]:

```
<seaborn.axisgrid.FacetGrid at 0x264044dae80>
```




In [61]:

```
duration_price = outage[['pct_water_tot', 'outage_duration_log10']]
duration_price.columns = ['PWT', 'duration_log10']

outcome, predictors = patsy.dmatrices('duration_log10 ~ PWT', duration_price)

# Now use statsmodels to intialize an OLS linear model
# This step initializes the model, and provides the data (but does not actually compute the model)
```

```
mod_log = sm.OLS(outcome, predictors)

# fit the model
res_log = mod_log.fit()

# Check out the results
print(res_log.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:         duration_log10   R-squared:                       0.010
Model:                            OLS   Adj. R-squared:                  0.009
Method:                 Least Squares   F-statistic:                     13.45
Date:                Tue, 07 Jun 2022   Prob (F-statistic):           0.000254
Time:                        03:23:47   Log-Likelihood:                -2207.9
No. Observations:                1320   AIC:                             4420.
Df Residuals:                    1318   BIC:                             4430.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      2.2090      0.048     45.554      0.000       2.114       2.304
PWT            0.0124      0.003      3.668      0.000       0.006       0.019
==============================================================================
Omnibus:                      240.245   Durbin-Watson:                   1.567
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              382.062
Skew:                          -1.263   Prob(JB):                     1.09e-83
Kurtosis:                       3.749   Cond. No.                         19.6
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Summary of `pct_water_tot` V.S. `Duration`

- The `Cond. No. is 19.6`, which is a reasonable range.
- the `p-value is 0.000 <  0.05`, which suggest that these results happen due to random chance alone is approximately 0% of the time, which is inferential.
- the `slope of pct_water_tot is 0.0124`, which suggest very slight propertional relationship between `PCT_LAND` V.S. `Duration`
- the `R-squared is 0.010`, which means 1% of the data variability is explained by the regression model.

Overall, the analysis suggest there is a propertional relationship between `pct_water_tot` V.S. `Duration`, but the data `doesn't fit into the model well`.

# 3 Check if Multiple Variables can explain data better

## 3.1Put all variable together

In [62]:
```
duration_price = outage[['sales_price','total_price','pc_realgsp_change','poppct_urban','poppct_uc','popden_uc','pct_
duration_price.columns = ['SP','P','PRC', 'PURBAN','PU','PDU','PWT','PL', 'duration_log10']
outcome, predictors = patsy.dmatrices('duration_log10 ~  PL+PWT+PDU+PU+PURBAN+PRC+P+SP', duration_price)

# Now use statsmodels to intialize an OLS linear model
# This step initializes the model, and provides the data (but does not actually compute the model)
mod_log = sm.OLS(outcome, predictors)

# fit the model
res_log = mod_log.fit()

# Check out the results
print(res_log.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:         duration_log10   R-squared:                       0.119
Model:                            OLS   Adj. R-squared:                  0.114
Method:                 Least Squares   F-statistic:                     22.05
Date:                Tue, 07 Jun 2022   Prob (F-statistic):           9.83e-32
Time:                        03:23:47   Log-Likelihood:                -2121.7
No. Observations:                1313   AIC:                             4261.
Df Residuals:                    1304   BIC:                             4308.
Df Model:                           8
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept   -388.1099   1649.789     -0.235      0.814   -3624.640    2848.420
```

```
PL              3.9271     16.497      0.238      0.812     -28.437      36.291
PWT             3.9493     16.498      0.239      0.811     -28.415      36.314
PDU             0.0015      0.000      9.139      0.000       0.001       0.002
PU             -0.1072      0.010    -10.638      0.000      -0.127      -0.087
PURBAN         -0.0410      0.005     -8.830      0.000      -0.050      -0.032
PRC             0.0043      0.017      0.259      0.796      -0.028       0.037
P              -0.1497      0.017     -8.987      0.000      -0.182      -0.117
SP           5.267e-05   8.67e-06      6.071      0.000    3.56e-05    6.97e-05
==============================================================================
Omnibus:                      233.224   Durbin-Watson:                   1.743
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              373.445
Skew:                          -1.189   Prob(JB):                     8.08e-82
Kurtosis:                       4.081   Cond. No.                     1.05e+09
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.05e+09. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Summary:

- The `Cond. No. is 1.05e+09`, which suggests a strong multicollinearity problem.
- the `R-squared is 0.119`, which means 11.9% of the data variability is explained by the regression model.

Overall, the analysis suggest there is a strong multicollinearity problem, and only a small part of variance can be explained by the model.

## 3.2 some random analysis

### 3.21 Put percentage land and percentage water togerther

In [63]:

```python
duration_price = outage[['sales_price','total_price','pc_realgsp_change','poppct_urban','poppct_uc','popden_uc','pct_
duration_price.columns = ['SP','P','PRC', 'PURBAN','PU','PDU','PWT','PL', 'duration_log10']
outcome, predictors = patsy.dmatrices('duration_log10 ~  PL+PWT', duration_price)

# Now use statsmodels to intialize an OLS linear model
# This step initializes the model, and provides the data (but does not actually compute the model)
mod_log = sm.OLS(outcome, predictors)

# fit the model
res_log = mod_log.fit()

# Check out the results
print(res_log.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:         duration_log10   R-squared:                       0.010
Model:                            OLS   Adj. R-squared:                  0.009
Method:                 Least Squares   F-statistic:                     6.954
Date:                Tue, 07 Jun 2022   Prob (F-statistic):           0.000991
Time:                        03:23:47   Log-Likelihood:                -2207.7
No. Observations:                1320   AIC:                             4421.
Df Residuals:                    1317   BIC:                             4437.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept   1101.9081   1623.494      0.679      0.497   -2083.010    4286.826
PL           -10.9970     16.235     -0.677      0.498     -42.846      20.852
PWT          -10.9847     16.235     -0.677      0.499     -42.834      20.865
==============================================================================
Omnibus:                      242.390   Durbin-Watson:                   1.567
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              387.031
Skew:                          -1.269   Prob(JB):                     9.06e-85
Kurtosis:                       3.770   Cond. No.                     4.17e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.17e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Summary:

- The `Cond. No. is 4.17e+06`, which suggests a strong multicollinearity problem.
- the `R-squared is 0.010`, which means 11.9% of the data variability is explained by the regression model.

Overall, the condition number is too large and suggests that there is a strong multicollinearity problem, and the data doesn't fit the model well.

### 3.22 Put sales price and total price together

In [64]:

```
duration_price = outage[['sales_price','total_price','pc_realgsp_change','poppct_urban','poppct_uc','popden_uc','pct_
duration_price.columns = ['SP','P','PRC', 'PURBAN','PU','PDU','PWT','PL', 'duration_log10']
outcome, predictors = patsy.dmatrices('duration_log10 ~  SP*P', duration_price)

# Now use statsmodels to intialize an OLS linear model
# This step initializes the model, and provides the data (but does not actually compute the model)
mod_log = sm.OLS(outcome, predictors)

# fit the model
res_log = mod_log.fit()

# Check out the results
print(res_log.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:         duration_log10   R-squared:                       0.010
Model:                            OLS   Adj. R-squared:                  0.008
Method:                 Least Squares   F-statistic:                     4.618
Date:                Tue, 07 Jun 2022   Prob (F-statistic):            0.00321
Time:                        03:23:47   Log-Likelihood:                -2207.7
No. Observations:                1320   AIC:                             4423.
Df Residuals:                    1316   BIC:                             4444.
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      3.2304      0.382      8.466      0.000       2.482       3.979
SP         -2.061e-05   1.69e-05     -1.221      0.222   -5.37e-05    1.25e-05
P             -0.1050      0.034     -3.094      0.002      -0.172      -0.038
SP:P        2.727e-06   1.34e-06      2.038      0.042    1.02e-07    5.35e-06
==============================================================================
Omnibus:                      226.812   Durbin-Watson:                   1.548
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              351.846
Skew:                          -1.218   Prob(JB):                     3.96e-77
Kurtosis:                       3.683   Cond. No.                     2.64e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.64e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Summary:

- The `Cond. No. is 2.64e+06`, which suggests a strong multicollinearity problem.
- the `R-squared is 0.010`, which means 11.9% of the data variability is explained by the regression model.

Overall, the condition number is too large and suggests that there is a strong multicollinearity problem, and the data doesn't fit the model well.

In [65]:

```
month_count = outage[['year','month']]
month_count.value_counts(['year','month'])
```

Out[65]:

```
year  month
2011  8        40
2014  1        34
2011  7        34
      5        29
      6        27
               ..
2007  3         1
2008  3         1
      4         1
2009  11        1
2000  1         1
Length: 167, dtype: int64
```

In [66]:

```
month_count.groupby(['year','month']).size()
```

Out[66]:

```
year  month
2000  1        1
      3        2
```

```
      5         3
      6         1
      8         3
               ..
2016  2         6
      3         5
      4         8
      5         6
      6         3
Length: 167, dtype: int64
```

### 3.3 multi variable of all negative proportional relationship with duration outage

In [67]:

```
duration_price = outage[['sales_price','total_price','pc_realgsp_change','poppct_urban','poppct_uc','popden_uc','pct_
outcome, predictors = patsy.dmatrices('outage_duration_log10 ~  total_price+poppct_uc+pct_land', duration_price)

# Now use statsmodels to intialize an OLS linear model
# This step initializes the model, and provides the data (but does not actually compute the model)
mod_log = sm.OLS(outcome, predictors)

# fit the model
res_log = mod_log.fit()

# Check out the results
print(res_log.summary())
```

```
                              OLS Regression Results
==================================================================================
Dep. Variable:     outage_duration_log10   R-squared:                       0.041
Model:                               OLS   Adj. R-squared:                  0.039
Method:                    Least Squares   F-statistic:                     18.81
Date:                   Tue, 07 Jun 2022   Prob (F-statistic):           5.98e-12
Time:                           03:23:47   Log-Likelihood:                -2186.9
No. Observations:                   1320   AIC:                             4382.
Df Residuals:                       1316   BIC:                             4403.
Df Model:                              3
Covariance Type:               nonrobust
================================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
Intercept       4.4384      0.360     12.324      0.000       3.732       5.145
total_price    -0.0693      0.013     -5.233      0.000      -0.095      -0.043
poppct_uc      -0.0400      0.007     -5.466      0.000      -0.054      -0.026
pct_land       -0.0114      0.003     -3.282      0.001      -0.018      -0.005
==============================================================================
Omnibus:                      243.252   Durbin-Watson:                   1.602
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              389.421
Skew:                          -1.263   Prob(JB):                     2.74e-85
Kurtosis:                       3.839   Cond. No.                         947.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Summary of `total_price+poppct_uc+pct_land` V.S. `Duration`

- The `Cond. No. is 947`, which is a reasonable range.
- All the `p-value is 0.000(or near 0) <  0.05`, which suggest that these results happen due to random chance alone is approximately 0% of the time, which is inferential.
- the `slope of total_price is -0.0693`, which suggest very slight negative propertional relationship between `total_price` V.S. `Duration`.
- the `slope of poppct_uc is -0.0400`, which suggest very slight negative propertional relationship between `poppct_uc` V.S. `Duration`.
- the `slope of pct_land is -0.0114`, which suggest very slight negative propertional relationship between `pct_land` V.S. `Duration`.

### 3.4 multi variable of all positive proportional relationship with duration outage

There is only one variable called `pct_water_tot` has a positive proportional relationship with duration outage based on our analysis.

## 4. Results of EDA

By establishing OLS model with both single variable as well as combined variables, we observed that density of population in urban clusters is the main feature that explains for the most variance (0.015) in the outage duration. Although the model with relative humidity percentage (1.5%) is classified as a weak model, it is the best result we get across the rest of the models.

### Conclusion based on OLS models with single variable

In conclusion of all analysis above with single varible OLS, we can see there is, but very slight, negative linear relationship between

`Average monthly electricity price` (it accounts for 0.5% of the variance) and `Duration`,

`Density of population in urban clusters`(it accounts for 1.5% of the variance) and `Duration`,

`pct_land` (it accounts for 0.9% of the variance) and `Duration`,

and slight positive relationship between `pct_water_tot` (it accounts for 0.9% of the variance) and `Duration`.

Other factors either have colinear problem or the probability it happens due to random chance alone is too high for us to reject the null hypothesis.

### Single variable vs. Multiple variables

Even thought the OLS model that combines all variables together has higher R-Square value, the model has multilinearity problem so we cannot use it to do the conclusion and find the relationship. The OLS models with some combined variables have similar or even lower adjusted R-squared value than the models with single variable. Therefore, adding more features did not help account for more variance in our data set. By comparing the results from the models established using single variables and mutiple variables, we decide that the model which has the most significant effect is the OLS model with density of population in urban clusters.

# Ethics & Privacy

We did this project in a legal way, and our project will be done for possible academic use. Based on that, our team members acknowledge that there may be some ethic issues like potential biases in our data, unintended usage of our results, and some privacy issues. We will consider all those possible concerns in the following paragraphs.

When doing research including collecting data, researcher should put the privacy of the data as their top priority. “Data science pursued in a manner so that is equitable, with respect for privacy and consent, so as to ensure that it does not cause undue harm” (lecture 3_1). The quote above is the objective we learned in class, and we keep reminding ourselves when doing our project. Since data analysis is in a perspective of god, that we collect thousands of consequences (results) to find the relationship, we should follow the objective consequentialism instead of subjective consequentialism. Objective consequentialism defines an act as morally right when it has the best overall consequences among all possibilities and subjective consequentialism defines an act as morally right when it has the best foreseeable consequences. We try to make sure we find the best overall consequences of the data. Since we are analyzing results, we should be responsible for the analyzation of data not being biased by thinking thoroughly on all possibilities. We use the multi-variable OLS analysis to cross check the confounding variables. P value and the condition number in OLS report give us the information on whether we should trust the analysis or not.

Our project data is about the electricity outage, which is a data published by the government agencies. Different from the dataset containing personal information, our dataset does not relate with the data on people, but on the natural and national scale, which does not include any unique identifier for indivisual. In addition, government dataset is supervised by the public. Every citizen has the right to supervise government’s use of power and restrict it by discipline. Public supervision is the best way to construct a panopticon structure to restrict the abuse of power. It means that it is government’s responsibility to publish its report or data without hiding anything. Then government published data has basically no privacy based on government’s function and responsibility.

# Conclusion & Discussion

In 21st century, **electricity** is one of the most important resource that people could not live without. Since electricity is not a natural resource, there might be various factors contributing to its outage. Due to the fact that possible inconvenience brought by outage will affect lives of millions of people and cause troubles for manufactures and industries, we would like to learn more about what kind of factors are related with outage to reduce its impact by precaution.

In our project, we collect our data of outage in the US from `2000` to `2016`. Firstly, we cleaned the dataset and looked at the outage duration, which is the crucial data we need to do analysis. To ensure the correct distribution and not let too large values drive the relationship, we remove the outliers and apply log_10 transformation to the outage_duration. Then, we do single-factor OLS regression to find out whether there's correlation between `outage duration` and `outage occurences`, `electricity consumption`, `economy situation`, `population`, `land&water area`. It turned out that only outage occurences may have a association with outage duration. However, the outage occurences are comprised of analysis on outage cause, year, and month. (add t-test)From those analysis, we concluded that as the increasing number of outage, the average of outage duration will decrease. And our first hypothesis that severe weather is the most common reason for outage is true according to the counts. As we look into the graph provided under 2.1.1, we can explicitly discover that there are more than 600 cases caused by 'severe weather', which exceeds the second highest cause - 'intentional attack' - about 200 cases.

We also use z-test to find out that the cause and year with the most amount of outage events has significant relationship with the outage duration. (These groups are significantly different from general groups)

However, the phenomenon that all numerical factors have no correlation with outage duration is hard to convice us, so we decided to do OLS regression with combined factors to ensure we didn't miss the possibility that there might have some factors action together to affect the outage duration. We tried land&water area, prices, all factors that may have a negative relationship with small r-squared value, and all factors together. As a result, we didn't see a relationship between the outage duration time and those combined factors. Thus, we reject our original second hypothesis. **The electricity consumption, consumer served in that state, and land percentage don't have any correlation with outage duration, but it seems that the number of outage occurence has a negative correlation with outage duration time.**

Since our data is retrieved from the government dataset, where the government might intentionally looking for outage data, not including too many perspectives of data that might be relatable. For instance, some confounding variables could bring more severe weather to the specific place and those could not be found in the dataset. In the future, we may take a further step to include the geospatial scale information into the analysis. Fixed topographical variables like altitude and latitude might also count, as they might cause the severe weather and then contributing to longer outage duration found in the area. In addition, climate data (like temperature, wind, humidity etc.) of each state is a huge dataset that can also pinpointing down to the specific time of the day, which can possibly link with the time when people encounter outage. If we can get the topographical data and combine the specific climate data, we can dive deeper into these aspects and turn in a more comprehensive analysis.

Previous related studies about this topic mainly talked about the relationship between outage occurence/duration with climate and weather factors. One of our orginal aim to do this analysis is to ensure there's no confounding variable, like electricity price, population density etc., that affects the actual relation between ourage duration and weather/climate condition. Moreover, this analysis may help each states' government to better realize the most exact reason for outage and prepare in advance for it.

# Team Contributions

- Yunxiang Chi - Holding regular meeting, finding dataset, cleaning data, doing single variable OLS regression analysis, analyzing results, and writing conclusion
- Xiaoxuan Zhang - Hypothesis; Data Cleaning & Wrangling; Data Analysis; Conclusion & Discussion; Video Editing
- Peicong Wu - Overview; Part of data cleaning; Part of EDA code; EDA analysis summary text for most of single variable and multivariables.
- Jiajun Ni - background research & prior work; EDA for year, month, cause, and doing z-test for them; Adding analysis for ols; ppt and video preparation
- Ziyan Liu - dataset research & considering related ethnic problems; EDA analysis; project conclusion; video presentation.