A Study on Trends in Frequency and Intensity of Winter Storms in Colorado

Aimé Fournier
May 16, 2022
Resolve
Team ρ

Contributions

David Carlton: David served as the lead coder for this project. He was responsible for data processing, storm calculations, variable creation, designing the coding regressions, and interpreting the results. He also visualized the results in graphs and assisted with documentation of the project.

Walker Jones: Walker oversaw all communication. This meant communicating with sponsors and taking notes during the meetings and presentations. On top the communication, Walker also worked to gather and organize the initial data as well as working to clean data and assisted with some of the code writing. Walker was also in charge of writing the contributions, abstract, and helped with describing the visualizations and methods.

Sydney Jenkinson: Sydney was the leader of this project. She helped make the schedule and keep ensured that we followed the schedule. She assisted with some of the coding for visualizations as well as helping with simple linear regressions to find trends in the data. Sydney also wrote the Introduction and Conclusion.

Abstract

This report uses data provided by the United States Department of Agriculture's National Water and Climate Center, which is then analyzed for the Colorado Avalanche Information Center. The goal of this report is to determine if we see any statistically significant change in the frequency and intensity of winter storms in Colorado over time. This project effectively calculates and analyzes trends in winter storms across Colorado. The motive behind this project is to better understand trends in snowstorms to provide useful information about the relationship between winter storms and climate.

Introduction

The Colorado Avalanche Information Center (CAIC) is a program within the Colorado Department of Natural Resources, a state government agency. They provide information about snowpack stability throughout the state of Colorado for motorists and backcountry recreational use in order to reduce the number of avalanche related injuries, and economic damages within the state. They are interested in trends in frequency and intensity of winter storms.

We have worked with the CAIC for the past few months receiving Snow Water Equivalent (SWE) data across 115 stations, the majority of which span the last 30 years. SWE is a common snowpack measurement used by hydrologists and water managers to gauge the amount of water contained within snowpack. It is equal to the amount of depth contained within the snowpack when it melts. We worked with the stations that contain data for the full 30 years as this is a benchmark for climate questions. With this information, we have been able to look at trends in frequency and intensity of snowstorms in Colorado.

The CAIC defines a storm as beginning when SWE increases over a 24 hour period, and ending when there is no increase over a 24 hour period. We define frequency for this project as the number of winter storms in a given year, and intensity as the the number of days taken for a storm to surpass a given SWE threshold, with a shorter duration indicating a more intense storm assuming the same threshold. These thresholds will be defined later in the notebook.

We hope that the impact of the project is to provide insight to the CAIC about trends in winter storms, so that they can use them to better understand the climate in Colorado, possibly extend the study to other locations, and ultimately make useful conclusions about winter storm trends.

Methods

We received data directly from the CAIC. The file includes daily SWE measurements from 1990 to 2021 across 115 Colorado stations, as well as the daily average temperature for the same time period. For this project, we decided to work with stations that have data for the full time period, as about 30% do not. We did this because 30 years is a benchmark for climate questions according to the CAIC, so those with less are not appropriate to include. The file contains 1,048,405 rows and 4 columns containing the timestamp, station name, SWE value in inches, and average temperature in degrees Fahrenheit, respectively. We also filter the dataset to only include the winter months because we are dealing with winter storms.This dataset and similar datasets can be generated using the US Department of Agriculture's website, found here: https://wcc.sc.egov.usda.gov/reportGenerator/

The dataset we worked with is organized in a stacked format (all stations in one column). We formatted it this way to allow for creation of new columns necessary to our analysis.

Time series analysis, autoregression, and linear regression are our primary statistical methods for this project. We begin by conducting time series analysis on the raw SWE data before calculating the number of winter storms per year. Next, we build a dataframe containing the number of storms per station per year, and use linear regression to discover the relationship

Data Processing

Required files: 'CAICData.xlsx' (in Math Clinic Folder)

```
# If you get an error message in the next cell, uncomment below and install, then 
!pip install statsmodels --upgrade
```

```
Requirement already satisfied: statsmodels i
Collecting statsmodels
  Downloading statsmodels-0.13.2-cp37-cp37m-
     |████████████████████████████████| 9.8
Requirement already satisfied: packaging>=21
Requirement already satisfied: patsy>=0.5.2
Requirement already satisfied: pandas>=0.25
Requirement already satisfied: scipy>=1.3 in
Requirement already satisfied: numpy>=1.17 i
Requirement already satisfied: pyparsing!=3.
Requirement already satisfied: python-dateut
Requirement already satisfied: pytz>=2017.3
Requirement already satisfied: six in /usr/l
Installing collected packages: statsmodels
  Attempting uninstall: statsmodels
    Found existing installation: statsmodels
    Uninstalling statsmodels-0.10.2:
      Successfully uninstalled statsmodels-0
Successfully installed statsmodels-0.13.2
```

```
# Importing libraries

import matplotlib
import pandas as pd
import numpy as np
from google.colab import drive
from google.colab import files
```

```python
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import matplotlib.colors as mcolors
import os
import seaborn as sns
from IPython.core.pylabtools import figsize
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
import statsmodels.api as sm
import joblib
import sys
sys.modules['sklearn.externals.joblib'] = joblib
from mlxtend.feature_selection import SequentialFeatureSelector as SFS
from mlxtend.plotting import plot_sequential_feature_selection as plot_sfs
from scipy import stats
from pandas.plotting import lag_plot
from statsmodels.tsa.api import acf, graphics, pacf
from statsmodels.tsa.ar_model import AutoReg, ar_select_order
from pandas.plotting import scatter_matrix

%matplotlib inline
sns.set_theme()
plt.rcParams['axes.grid'] = True
```

```python
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

```python
cwd = os.getcwd()       # Assumes no cd commands were executed
pathTeam = cwd + '/gdrive/My Drive/'
pathProfessor = 'Colab Notebooks/Math Clinic/2022sp/CAIC/'
if os.path.exists(pathTeam + pathProfessor):
 pathTeam += pathProfessor
#os.listdir(pathTeam)    # Just a handy check that we see the expected files
```

The following block takes about 2 minutes to run. We import the data then convert it to a Pandas DataFrame.

```python
swe_data = pd.read_excel(pathTeam+'CAICData.xlsx', sheet_name=None, parse_dates=Tru
```

```python
df = pd.DataFrame.from_dict({(i,j): swe_data[i][j]
                             for i in swe_data.keys()
                             for j in swe_data[i].keys()})
```

```python
df
```

	Sheet1		
	Date	Station Name	Snow Water
0	1990-01-01	Apishapa	
1	1990-01-02	Apishapa	
2	1990-01-03	Apishapa	
3	1990-01-04	Apishapa	
4	1990-01-05	Apishapa	
...	...	...	
1046716	2021-12-27	Vallecito	
1046717	2021-12-28	Vallecito	
1046718	2021-12-29	Vallecito	
1046719	2021-12-30	Vallecito	
1046720	2021-12-31	Vallecito	

1046721 rows × 4 columns

In the following two cells, we organize the columns and add variables containing the daily change in SWE, as well as lagged (one day behind) versions of both of those variables. The lagged version will be used to check certain conditions and accurately calculate frequency of storms.

```python
df['Date'] = df[('Sheet1', 'Date')]
df['Station Name'] = df[('Sheet1', 'Station Name')]
df['SWE (in)'] = df[('Sheet1', 'Snow Water Equivalent (in) Start of Day Values')]
df['Air Temp Avg'] = df[('Sheet1', 'Air Temperature Average (degF)')]

df = df.drop(columns=[('Sheet1', 'Date'),
              ('Sheet1', 'Station Name'),
              ('Sheet1', 'Snow Water Equivalent (in) Start of Day Values'),
              ('Sheet1', 'Air Temperature Average (degF)')])
```

```python
df['SWE Difference'] = df['SWE (in)'].diff()
df['Lagged SWE'] = df['SWE (in)'].shift()
df['Lagged SWE Difference'] = df['SWE Difference'].shift()

# Indicator of positive change in SWE
df['Storm Indicator'] = np.where(df['SWE Difference'] > 0, True, False)
```

This next block creates a variable that counts the number of consecutive days that the storm indicator variable has been true. This calculates the duration in days of each storm, which is key for measuring intensity.

```
g1 = df['Storm Indicator'].ne(df['Storm Indicator'].shift()).cumsum()
df['Duration'] = df.groupby(g1)['Storm Indicator'].transform('size') * np.where(df[
df['Duration Lag'] = df['Duration'].shift()
```

Here, we unstack the data now that our new variables are created, but keep a stacked version for calculation of storms.

```
dfStacked = df.copy()
df = df.pivot(index='Date', columns='Station Na
df
```

Aimé Fournier Resolve
May 17, 2022
(edited 12:51 PM Yesterday)

Note that averaging a difference implies all but a couple of values cancel each other.

	SWE (in)				
Station Name	Apishapa	Arapaho Ridge	Bear Lake	Bear River	Bea
Date					
1990-01-	1.4	NaN	5.1	NaN	
02	1.4	NaN	5.2	NaN	
03	1.4	NaN	5.2	NaN	
1990-01-					

Here, we filter the data to only include winter months.

```
df = df[df.index.month.isin([10, 11, 12, 1, 2, 3])]
```

In the next two cells, we create columns containing the averages of our variables of interest across all stations.

```
df['Average SWE'] = df['SWE (in)'].mean(axis=1)
df['Average SWE Difference'] = df['SWE Difference'].mean(axis=1)
df['Average Air Temp'] = df['Air Temp Avg'].mean(axis=1)
df['Average Storm Duration'] = df['Duration'].mean(axis=1)
```

```
/usr/local/lib/python3.7/dist-packages/ipyke
A value is trying to be set on a copy of a s
Try using .loc[row_indexer,col_indexer] = va

See the caveats in the documentation: https:
  """Entry point for launching an IPython ke
/usr/local/lib/python3.7/dist-packages/ipyke
A value is trying to be set on a copy of a s
Try using .loc[row_indexer,col_indexer] = va

See the caveats in the documentation: https:

/usr/local/lib/python3.7/dist-packages/ipyke
A value is trying to be set on a copy of a s
Try using .loc[row_indexer,col_indexer] = va

See the caveats in the documentation: https:
  This is separate from the ipykernel packag
/usr/local/lib/python3.7/dist-packages/ipyke
A value is trying to be set on a copy of a s
Try using .loc[row_indexer,col_indexer] = va

See the caveats in the documentation: https:
  after removing the cwd from sys.path.
```

```
# Used for scatter matrix later in notebook
dfAverages = df[['Average SWE', 'Average SWE Difference', 'Average Air Temp',
                 'Average Storm Duration']].copy()
dfAverages = pd.pivot_table(dfAverages, index='Date')
```

Frequency and Intensity Calculations

The variables we have calculated thus far tell us about changes in daily SWE values, but not necessarily about winter storms as defined by the CAIC. The CAIC defines a storm as beginning when SWE increases over 24 hours, and ending when SWE remains constant or decreases over 24 hours. The following code blocks loop through the data to check conditions that will calculate frequency of storms, that is, the number of storms each year. 11,688 is length of columns that contain the full 30 year data, and we wish to only use these for our calculation, which is why that number is used to filter the data. The 'Date1' variable is used because of an unresolved issue we had with the pandas groupby function.

```
dfStacked = dfStacked.groupby('Station Name').filter(lambda x: len(x) == 11688)
dfStacked.insert(0, 'Date1', dfStacked['Date'])
dfStacked.set_index('Date', inplace=True)
```

In the following cell, the loop checks for storms that exceed a given SWE threshold over their timeframe. We check those conditions and then create a dataframe containing the results.

```
dfGrouped = dfStacked.groupby([pd.Grouper(key='Date1', freq='Y'), 'Station Name'])
g = [group for frame, group in dfGrouped]

stormsList = []
count = 0
sweInitial = 0
sweFinal = 0

# Here, the SWE threshold is defined. It may be adjusted as necessary.
threshold = 2

for year in g: # Loop through each year of dataframe
  for swe, sweLag, sweDiff, sweDiffLag in zip(
                         year['SWE (in)'],
                         year['Lagged SWE'],
                         year['SWE Difference'],
                         year['Lagged SWE Difference']):
    if sweDiff > 0 and sweDiffLag <= 0: # Set initial value when storm begins
      sweInitial = sweLag
    elif sweDiff > 0 and sweDiffLag > 0: # Continue if storm in progress
      continue
    elif sweDiff <= 0 and sweDiffLag > 0: # Set final value when storm ends
      sweFinal = sweLag
      if (sweFinal - sweInitial) > threshold: # Count a storm if threshold is excee
        count += 1
  stormsList.append(count) # Add count to list of storms
  count = 0 # Reset storm count for each year
```

The next cell loops through the grouped dataframe and calculates stores the duration of storms for each year, skipping days where there is no storm.

```
intensityList = []

for year in g:
  for duration in year['Duration']:
    if duration > 0:
      intensityList.append(duration)
```

We take the list of storms and durations calculated in the previous blocks and split them by station and year. The resulting dataframes contain the number of winter storms that exceed the given SWE threshold for each station, each year, and the duration of each storm organized by year, respectively.

```
dfGroupedName = dfStacked.groupby('Station Name')
stormsListSplit = np.array_split(stormsList, 32)
dfStorms = pd.DataFrame(stormsListSplit, columns=dfGroupedName.groups.keys())
dfStorms.insert(0, 'Year', np.arange(1990, 2022, 1))
dfStorms.set_index(np.arange(1990, 2022, 1), inplace=True)
dfStorms['Average Storms Across Stations'] = dfStorms.drop('Year', axis=1).mean(nur
```

```
intensityListSplit = np.array_split(intensityList, 32)
dfIntensity = pd.DataFrame(intensityListSplit)
```

```
dfIntensity.insert(0, 'Year', np.arange(1990, 2022, 1))
dfIntensity.set_index(np.arange(1990, 2022, 1), inplace=True)
dfIntensity['Average Storm Duration'] = dfIntensity.drop('Year', axis=1).mean(numer
```

```
dfStorms
```

	Year	Apishapa	Bear Lake	Beartown	Berthou Summi
1990	1990	1	4	3	
1991	1991	1	1	5	
1992	1992	2	0	2	
1993	1993	1	4	5	
1994	1994	0	2	7	
1995	1995	1	4	4	
1996	1996	0	4	5	
1999	1999	3	4	4	
2005	2005	1	4	4	
2010	2010	0	2	4	

Autoregression

First, we run an autoregression on each of our variables of interest to see if there are statistically significant trends over time without considering to their relationships to other variables. The period we use is 365 days, implying that each value is being compared to itself exactly a year before.

To run this regression for individual stations, uncomment the first line of the following cell, comment out the second line, replace Average SWE with the variable of interest, and replace Apishapa with the station of interest.

```
# mod = AutoReg(df[('Average SWE','','Apishapa')], period=365, lags=10)
mod = AutoReg(df['Average SWE'], period=365, lags=10)
mod2 = AutoReg(df['Average SWE Difference'], period=365, lags=10)
mod3 = AutoReg(df['Average Air Temp'], period=365, lags=10)
mod4 = AutoReg(df['Average Storm Duration'], period=365, lags=10)
res = mod.fit()
res2 = mod2.fit()
res3 = mod3.fit()
res4 = mod4.fit()
```

```
/usr/local/lib/python3.7/dist-packages/stats
  self._init_dates(dates, freq)
/usr/local/lib/python3.7/dist-packages/stats
```

```
  self._init_dates(dates, freq)
/usr/local/lib/python3.7/dist-packages/stats
  self._init_dates(dates, freq)
/usr/local/lib/python3.7/dist-packages/stats
  self._init_dates(dates, freq)
```

Linear Regression

Next, we run a linear regression to see if there is a relationship between air temperature and any of our variables. Professor Fournier requested that we use air temperature because our ultimate goal to provide useful climate information, and we know that air temperature is one of the most important factors in these types of questions. We run separate linear regressions on each variable, using average air temperature as the predictor.

Similar to the autoregression, this regression can be run for individual stations. The commented-out first line is a template which may be edited as desired.

```
# X = df[('Average Air Temp','','Apishapa')]
X = df['Average Air Temp']
Y = df['Average SWE']
X = sm.add_constant(X)

model = sm.OLS(Y, X).fit()
predictions = model.predict(X)

print_model = model.summary()
```

```
X2 = df['Average Air Temp']
Y2 = df['Average SWE Difference']
X2 = sm.add_constant(X2)

model2 = sm.OLS(Y2, X2).fit()
predictions2 = model.predict(X2)

print_model2 = model2.summary()
```

```
X3 = df['Average Air Temp']
Y3 = df['Average Storm Duration']
X3 = sm.add_constant(X3)

model3 = sm.OLS(Y3, X3).fit()
predictions3 = model.predict(X3)

print_model3 = model3.summary()
```

Aimé Fournier
May 19, 2022
Resolve
So the time axis *also* excludes 6 mo/y? That's weird.

Here, we calculate the average temperature for each year to match the frequency and intensity data.

```
dfTemp = dfStacked.groupby([pd.Grouper(key='Date1', freq='Y')]).agg('mean')
dfTemp = dfTemp.reset_index()
```

```
/usr/local/lib/python3.7/dist-packages/panda
  obj = obj._drop_axis(labels, axis, level=l
```

For the following regression, we used average storms across stations as the response. This may be replaced with any station of interest. To do this, replace 'Average Storms Across Stations' with the station name of interest.

```
X4 = list(dfTemp['Air Temp Avg'])

# Enter station name between the quotes on next line
Y4 = dfStorms['Average Storms Across Stations']
X4 = sm.add_constant(X4)

model4 = sm.OLS(Y4, X4).fit()
predictions4 = model4.predict(X4)

print_model4 = model4.summary()
```

```
X5 = list(dfTemp['Air Temp Avg'])
Y5 = dfIntensity['Average Storm Duration']
X5 = sm.add_constant(X5)

model5 = sm.OLS(Y5, X5).fit()
predictions5 = model5.predict(X5)

print_model5 = model5.summary()
```

Results and Discussion

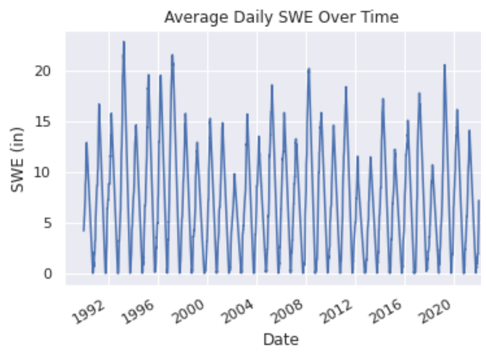
Autoregression

The autoregressions served as our first look at trends in the data without consideration to other variables. The ten coefficients are the lags of the regression, that is, 365 days plus 1, 2, 3, ..., 10 days. The highest correlation for all variables was the first level of the regression. They were also all statistically significant and were slightly positively correlated.

```
print(res.summary())
```

```
                                   AutoReg Model Re
=============================================
Dep. Variable:              Average SWE   No.
Model:                      AutoReg(10)   Log
Method:                 Conditional MLE   S.D.
Date:                  Mon, 16 May 2022   AIC
Time:                          22:05:42   BIC
Sample:                              10   HQIC
                                   5832
=============================================
                      coef    std err
---------------------------------------------
const               0.1870      0.026       7
Average SWE.L1      0.9923      0.013      75
Average SWE.L2     -0.0040      0.018      -0
Average SWE.L3     -0.0016      0.018      -0
Average SWE.L4     -0.0009      0.018      -0
Average SWE.L5      0.0009      0.018       0
Average SWE.L6      0.0018      0.018       0
Average SWE.L7     -0.0006      0.018      -0
Average SWE.L8     -0.0006      0.018      -0
Average SWE.L9      0.0011      0.018       0
Average SWE.L10    -0.0156      0.013      -1
                                       Roots
=============================================
                  Real          Imaginary
---------------------------------------------
AR.1            1.0335           -0.0000j
AR.2            1.3716           -0.0000j
AR.3            1.0880           -1.0690j
AR.4            1.0880           +1.0690j
AR.5            0.1851           -1.6022j
AR.6            0.1851           +1.6022j
AR.7           -1.5693           -0.5476j
AR.8           -1.5693           +0.5476j
AR.9           -0.8711           -1.3925j
AR.10          -0.8711           +1.3925j
---------------------------------------------
```

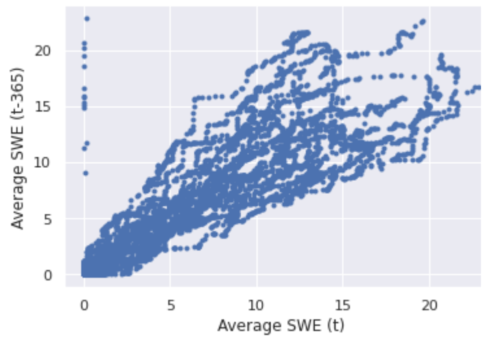
```
df['Average SWE'].plot(ylabel='SWE (in)', title='Average Daily SWE Over Time');
```



These (t-365) scatter plots show the correlation between a given measurement and its value a year before.

```
plt.scatter(df['Average SWE'], df['Average SWE'].shift(365), marker='.')
plt.ylabel('Average SWE (t-365)')
plt.xlabel('Average SWE (t)')
```

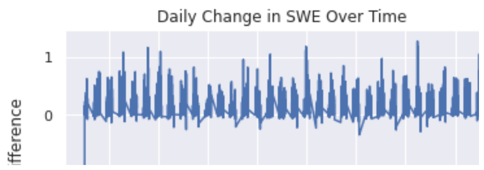
```
Text(0.5, 0, 'Average SWE (t)')
```



```
print(res2.summary())
```

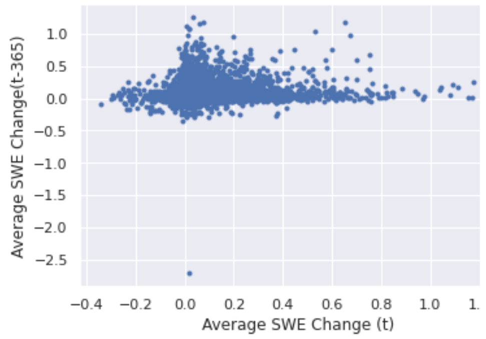
```
                              AutoReg Model
=============================================
Dep. Variable:     Average SWE Difference
Model:                        AutoReg(10)
Method:                   Conditional MLE
Date:                    Mon, 16 May 2022
Time:                            22:05:42
Sample:                                10
                                     5832
=============================================
                               coef     std
---------------------------------------------
const                        0.0379       0
Average SWE Difference.L1    0.4386       0
Average SWE Difference.L2   -0.0114       0
Average SWE Difference.L3    0.0473       0
Average SWE Difference.L4    0.0040       0
Average SWE Difference.L5    0.0138       0
Average SWE Difference.L6    0.0266       0
Average SWE Difference.L7    0.0122       0
Average SWE Difference.L8   -0.0015       0
Average SWE Difference.L9    0.0045       0
Average SWE Difference.L10   0.0214       0
                                    Roots
=============================================
                  Real           Imaginary
---------------------------------------------
AR.1            1.2320           -0.0000j
AR.2            1.1859           -0.8239j
AR.3            1.1859           +0.8239j
AR.4            0.5599           -1.3535j
AR.5            0.5599           +1.3535j
AR.6           -0.4213           -1.4076j
AR.7           -0.4213           +1.4076j
AR.8           -1.2475           -0.9496j
AR.9           -1.2475           +0.9496j
AR.10          -1.5944           -0.0000j
---------------------------------------------
```

```
df['Average SWE Difference'].plot(ylabel='SWE Difference',
                                  title='Daily Change in SWE Over Time');
```



```
plt.scatter(df['Average SWE Difference'], df['Average SWE Difference'].shift(365),
plt.ylabel('Average SWE Change(t-365)')
plt.xlabel('Average SWE Change (t)')
```

```
Text(0.5, 0, 'Average SWE Change (t)')
```

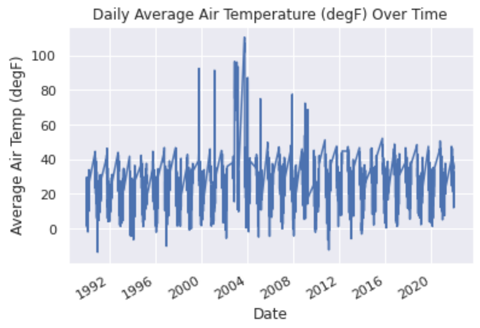


```
print(res3.summary())
```

```
                              AutoReg Model Re
=============================================
Dep. Variable:        Average Air Temp    No.
Model:                     AutoReg(10)    Log
Method:                Conditional MLE    S.D.
Date:                 Mon, 16 May 2022    AIC
Time:                         22:05:43    BIC
Sample:                             10    HQIC
                                  5832
=============================================
                          coef     std err
---------------------------------------------
const                   2.0529       0.216
Average Air Temp.L1     0.8980       0.013
Average Air Temp.L2    -0.2024       0.018
Average Air Temp.L3     0.0703       0.018
Average Air Temp.L4     0.0249       0.018
Average Air Temp.L5     0.0292       0.018
Average Air Temp.L6     0.0293       0.018
Average Air Temp.L7     0.0081       0.018
Average Air Temp.L8     0.0076       0.018
Average Air Temp.L9     0.0140       0.018
Average Air Temp.L10    0.0414       0.013
```

```
                                         Roots
=============================================
                    Real          Imaginary
---------------------------------------------
AR.1              1.0409           -0.0000j
AR.2              1.0869           -0.6971j
AR.3              1.0869           +0.6971j
AR.4              0.5423           -1.2238j
AR.5              0.5423           +1.2238j
AR.6             -0.3419           -1.4040j
AR.7             -0.3419           +1.4040j
AR.8             -1.5605           -0.0000j
AR.9             -1.1960           -0.9764j
AR.10            -1.1960           +0.9764j
---------------------------------------------
```

```
df['Average Air Temp'].plot(ylabel='Average Air Temp (degF)',
                            title='Daily Average Air Temperature (degF) Over Time')
```



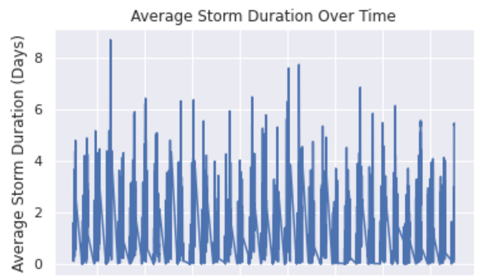
```
plt.scatter(df['Average Air Temp'], df['Average Air Temp'].shift(365), marker='.')
plt.ylabel('Average Air Temp (degF) (t-365)')
plt.xlabel('Average Air Temp (degF) (t)')
```

```
Text(0.5, 0, 'Average Air Temp (degF) (t)')
```

```
print(res4.summary())
```

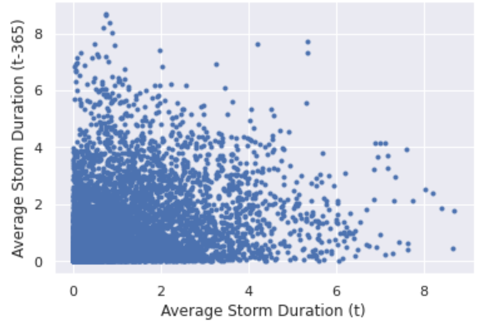
```
                                   AutoReg Model
=============================================
Dep. Variable:       Average Storm Duration
Model:                          AutoReg(10)
Method:                     Conditional MLE
Date:                      Mon, 16 May 2022
Time:                              22:05:44
Sample:                                  10
                                       5832
=============================================
                                 coef     std
---------------------------------------------
const                          0.2002       0
Average Storm Duration.L1      1.0278       0
Average Storm Duration.L2     -0.1609       0
Average Storm Duration.L3     -0.0043       0
Average Storm Duration.L4     -0.0181       0
Average Storm Duration.L5     -0.0117       0
Average Storm Duration.L6      0.0030       0
Average Storm Duration.L7      0.0182       0
Average Storm Duration.L8      0.0109       0
Average Storm Duration.L9     -0.0174       0
Average Storm Duration.L10     0.0044       0
                                     Roots
=============================================
                    Real          Imaginary
---------------------------------------------
AR.1             -1.7083           -0.0000j
AR.2             -1.2353           -1.1329j
AR.3             -1.2353           +1.1329j
AR.4             -0.0941           -1.5982j
AR.5             -0.0941           +1.5982j
AR.6              1.2208           -0.0000j
AR.7              1.2492           -1.0776j
AR.8              1.2492           +1.0776j
AR.9              2.3003           -0.5090j
AR.10             2.3003           +0.5090j
---------------------------------------------
```

```
df['Average Storm Duration'].plot(ylabel='Average Storm Duration (Days)',
                                  title='Average Storm Duration Over Time');
```



```
plt.scatter(df['Average Storm Duration'], df['Average Storm Duration'].shift(365),
plt.ylabel('Average Storm Duration (t-365)')
plt.xlabel('Average Storm Duration (t)')
```

```
Text(0.5, 0, 'Average Storm Duration (t)')
```



Linear Regression

First, we print the results of our regressions using temperature as predictor of each variable, plot the denisty of the residuals, and visualize the correlation using a scatter plot matrix. The diagonal of this matrix shows the denisty of each variable.

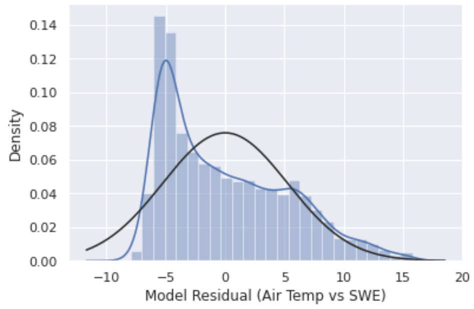
```
print(print_model)
```

```
                            OLS Regression R
=============================================
Dep. Variable:            Average SWE   R-sq
Model:                            OLS   Adj.
Method:                 Least Squares   F-st
Date:                Mon, 16 May 2022   Prob
Time:                        22:05:41   Log-
No. Observations:                5832   AIC:
Df Residuals:                    5830   BIC:
Df Model:                           1
Covariance Type:            nonrobust
=============================================
                      coef    std err
---------------------------------------------
const               9.4734      0.154      6
Average Air Temp   -0.1012      0.005     -1
=============================================
Omnibus:                      483.479   Durb
Prob(Omnibus):                  0.000   Jarq
Skew:                           0.711   Prob
Kurtosis:                       2.556   Cond
=============================================

Notes:
[1] Standard Errors assume that the covarian
```

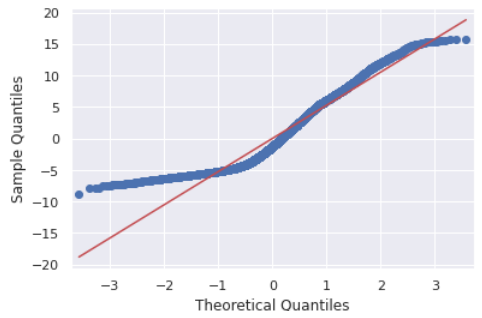
Here, we see a negative relationship between air temperature and average SWE, as expected.

```
sns.distplot(model.resid, fit=stats.norm, axlabel='Model Residual (Air Temp vs SWE)
```

```
/usr/local/lib/python3.7/dist-packages/seabo
  warnings.warn(msg, FutureWarning)
```



```
sm.qqplot(model.resid, line='s');
```



The above plots show how closely our model residuals follow a normal distribution. The closer they are to the black and red lines, repsectively, the closer they follow it. For these variables, we can see that they are not quite normally distributed.

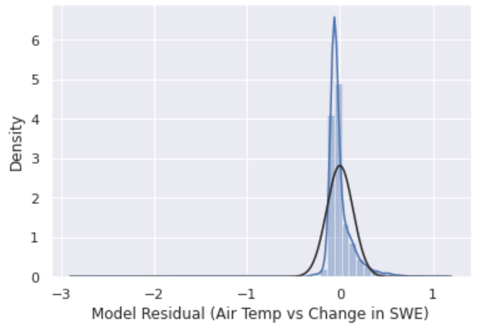
```
print(print_model2)
```

```
                               OLS Regression
============================================
Dep. Variable:      Average SWE Difference
Model:                                 OLS
Method:                      Least Squares
Date:                     Mon, 16 May 2022
Time:                             22:05:41
No. Observations:                     5832
Df Residuals:                         5830
Df Model:                                1
Covariance Type:                 nonrobust
============================================
                        coef    std err
--------------------------------------------
const                 0.1673      0.004      4
Average Air Temp     -0.0032      0.000     -2
============================================
Omnibus:                     2443.523   Durb
Prob(Omnibus):                  0.000   Jarq
Skew:                           0.984   Prob
Kurtosis:                      38.741   Cond
============================================

Notes:
[1] Standard Errors assume that the covarian
```

This relationship is also negative, but the relationship is not as strong. This tells us that air temperature is not a great predictor of daily change in SWE

```
sns.distplot(model2.resid, fit=stats.norm, axlabel='Model Residual (Air Temp vs Cha
```

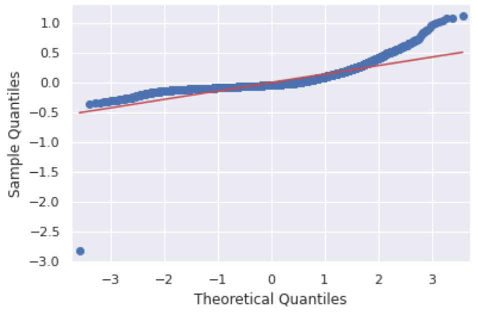
```
/usr/local/lib/python3.7/dist-packages/seabo
  warnings.warn(msg, FutureWarning)
```



Aimé Fournier
May 19, 2022
Resolve

It's surprising the anticorrelation with T is so weak.

```
sm.qqplot(model2.resid, line='s');
```



```
print(print_model3)
```

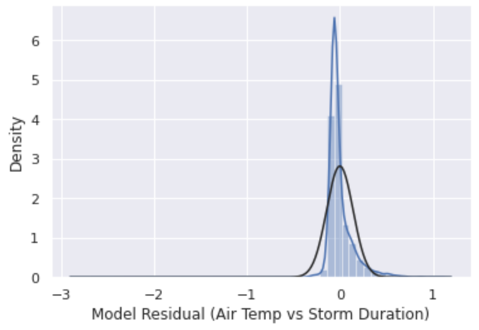
```
                               OLS Regression
============================================
Dep. Variable:      Average Storm Duration
Model:                                 OLS
Method:                      Least Squares
Date:                     Mon, 16 May 2022
Time:                             22:05:41
No. Observations:                     5832
```

```
Df Residuals:                     5830
Df Model:                            1
Covariance Type:             nonrobust
============================================
                      coef    std err

--------------------------------------------
const               2.3187      0.037      6
Average Air Temp   -0.0377      0.001     -2
============================================
Omnibus:                  1669.045   Durb
Prob(Omnibus):               0.000   Jarq
Skew:                        1.551   Prob
Kurtosis:                    5.866   Cond
============================================

Notes:
[1] Standard Errors assume that the covarian
```

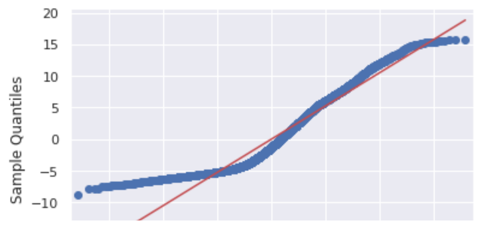
Here, we get a higher R-squared, but a small coefficient. This relationship is more interesting when visualized.

```
sns.distplot(model2.resid, fit=stats.norm, axlabel='Model Residual (Air Temp vs Sto
```

```
/usr/local/lib/python3.7/dist-packages/seabo
  warnings.warn(msg, FutureWarning)
```



```
sm.qqplot(model.resid, line='s');
```



The following cells visualize these relationships using a correlation matrix, heat map, and scatter plot matrix with density plots along the diagonal.

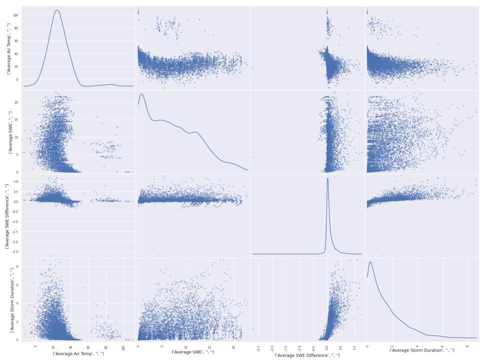
```
dfAverages.corr()
```

Station Name	Average Air Temp	Average SWE	Avera SWE Diffe
Station Name			
Average Air Temp	1.000000	-0.241135	-0.

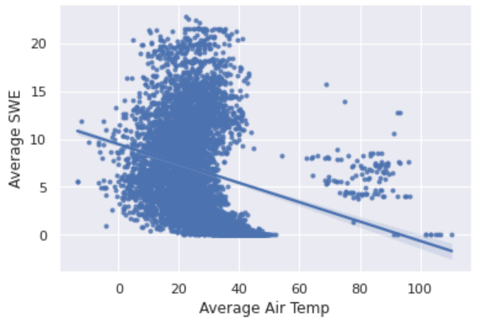
```
sns.heatmap(dfAverages.corr());
```

ion Name
Average Air Temp--
Average SWE--
1.0
0.8
0.6

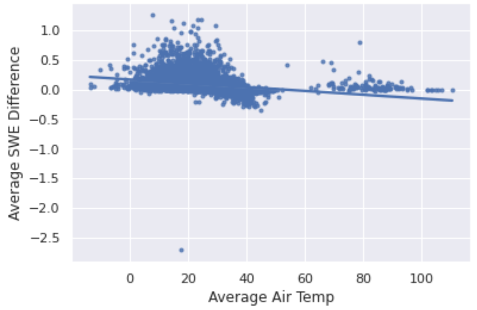
```
scatter_matrix(dfAverages, alpha = 0.5, figsize = (24, 18), diagonal = 'kde', grid=
```



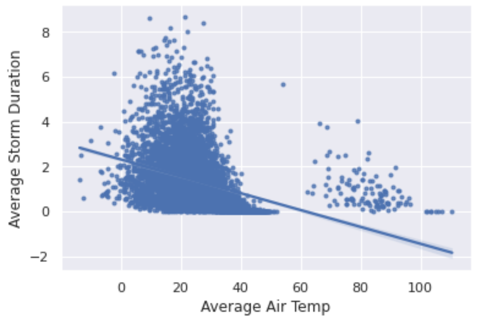
```
sns.regplot(x='Average Air Temp', y='Average SWE', data=df, marker='.');
```



```
sns.regplot(x='Average Air Temp', y='Average SWE Difference', data=df, marker='.');
```



```
sns.regplot(x='Average Air Temp', y='Average Storm Duration', data=df, marker='.');
```



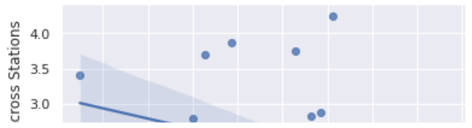
Next are the results of our prediction of frequency and duration. The bands around the regression lines indicate 95% confidence intervals

```
print(print_model4)
```

```
                                    OLS Regres
==============================================
Dep. Variable:      Average Storms Across Sta
Model:
Method:                              Least Sq
Date:                            Mon, 16 May
Time:                                     22:
No. Observations:
Df Residuals:
Df Model:
Covariance Type:                          nonr
==============================================
                   coef    std err          t
----------------------------------------------
const            7.4442      2.479      3.002
x1              -0.1454      0.070     -2.079
==============================================
Omnibus:                          3.293   Durb
Prob(Omnibus):                    0.193   Jarq
Skew:                             0.634   Prob
Kurtosis:                         3.244   Cond
==============================================

Notes:
[1] Standard Errors assume that the covarian
```

The coefficient and R-squared value here indicates that average air temperature somewhat negatively predicts frequency of storms. In future analysis, other variables may be included.

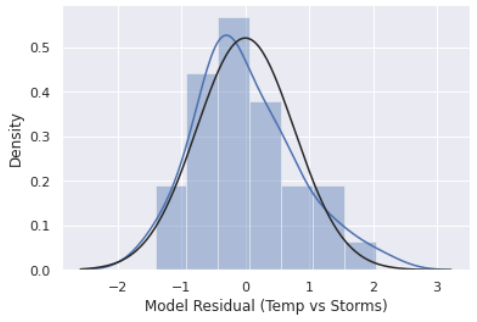
```
sns.regplot(x=dfTemp['Air Temp Avg'],
            y=dfStorms['Average Storms Across Stations']);
```



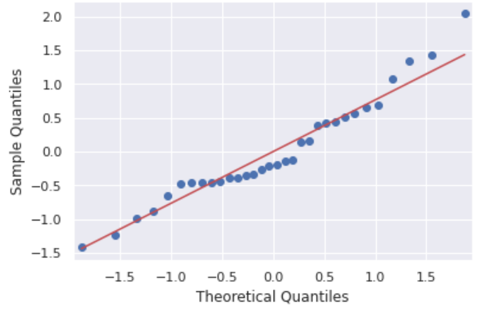
Here, we see a negative correlation between air temperature and frequency of storms.

```
sns.distplot(model4.resid, fit=stats.norm, axlabel='Model Residual (Temp vs Storms)
```

```
/usr/local/lib/python3.7/dist-packages/seabo
  warnings.warn(msg, FutureWarning)
```



```
sm.qqplot(model4.resid, line='s');
```



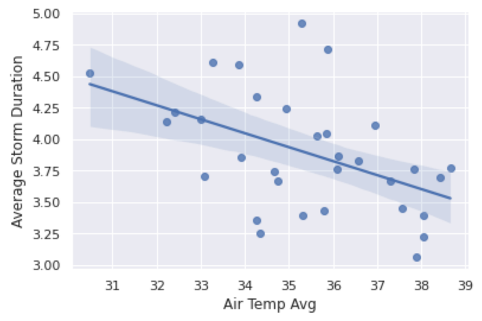
```
print(print_model5)
```

```
                                    OLS Regression
```

```
=============================================
Dep. Variable:     Average Storm Duration
Model:                                OLS
Method:                     Least Squares
Date:                    Mon, 16 May 2022
Time:                            22:05:42
No. Observations:                      32
Df Residuals:                          30
Df Model:                               1
Covariance Type:                nonrobust
=============================================
                 coef    std err          t
---------------------------------------------
const          7.8246      1.305      5.997
x1            -0.1111      0.037     -3.019
=============================================
Omnibus:                        1.562   Durb
Prob(Omnibus):                  0.458   Jarq
Skew:                           0.406   Prob
Kurtosis:                       3.112   Cond
=============================================

Notes:
[1] Standard Errors assume that the covarian
```

Air temperature appears to be a better predictor of average storm duration and frequency.

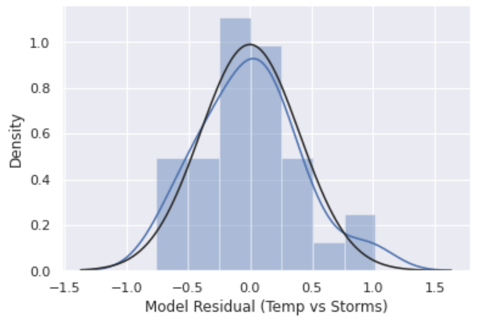
```
sns.regplot(x=dfTemp['Air Temp Avg'],
            y=dfIntensity['Average Storm Duration']);
```



We also see a negative relationship between average yearly storm duration and air temperature.

```
sns.distplot(model5.resid, fit=stats.norm, axlabel='Model Residual (Temp vs Storms)
```

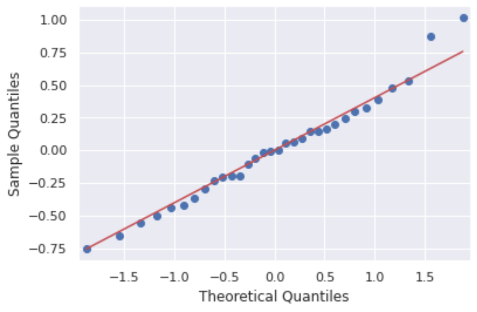
```
/usr/local/lib/python3.7/dist-packages/seabo
  warnings.warn(msg, FutureWarning)
```



A Aimé Fournier
May 19, 2022
Resolve

The time axis should have run from Oct. through Mar., not had a spurious jump in the middle

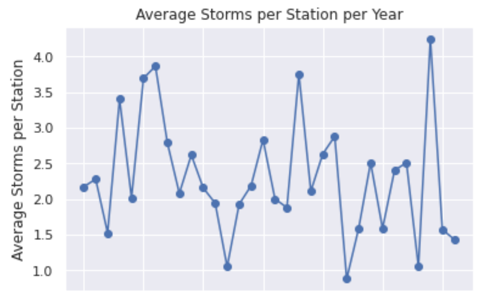
```
sm.qqplot(model5.resid, line='s');
```



Additional Visualizations

```
dfStorms['Average Storms Across Stations'].plot(ylabel='Average Storms per Station'
                                                title='Average Storms per Station p
                                                marker='o')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x
```

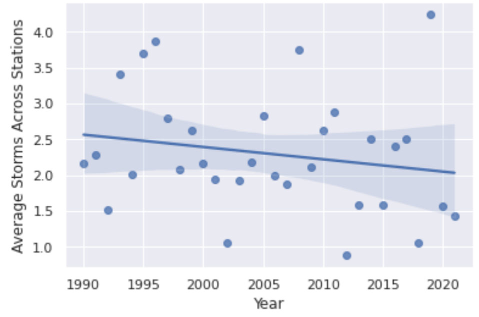


Professor Fournier recommended that we do not use year as a predictor variable, but the following plots show the trend of both frequency and duration versus year.

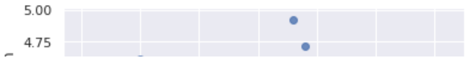
Aimé Fournier
May 19, 2022
Resolve

What about cleaning the T data of bogus values?

```
sns.regplot(x='Year', y='Average Storms Across S
```



```
sns.regplot(x='Year', y='Average Storm Duration', data=dfIntensity);
```



```
dfAverages = dfAverages.reset_index()
```

```
dfAverages.groupby([pd.Grouper(key='Date', freq='Y')])['Average SWE']
```

```
<pandas.core.groupby.generic.SeriesGroupBy o
```

```
index = df.index
dfSwe = pd.DataFrame({'date': index, 'Average_SWE': dfAverages['Average SWE']})
dfSweDiff = pd.DataFrame({'date': index, 'Average_SWE_Difference': dfAverages['Aver
dfAirTemp = pd.DataFrame({'date': index, 'Average_Air_Temp': dfAverages['Average Ai
dfStormDur = pd.DataFrame({'date': index, 'Average_Storm_Duration': dfAverages['Ave
```

```
dfSwe['Year'] = dfSwe.date.dt.year
dfSwe['Date'] = dfSwe.date.dt.strftime('%m-%d')
unstacked = dfSwe.set_index(['Year', 'Date']).A

dfSweDiff['Year'] = dfSweDiff.date.dt.year
dfSweDiff['Date'] = dfSweDiff.date.dt.strftime(
unstacked2 = dfSweDiff.set_index(['Year', 'Date                                     (-

dfAirTemp['Year'] = dfAirTemp.date.dt.year
dfAirTemp['Date'] = dfAirTemp.date.dt.strftime('%m-%d')
unstacked3 = dfAirTemp.set_index(['Year', 'Date']).Average_Air_Temp.unstack(-2)

dfStormDur['Year'] = dfStormDur.date.dt.year
dfStormDur['Date'] = dfStormDur.date.dt.strftime('%m-%d')
unstacked4 = dfStormDur.set_index(['Year', 'Date']).Average_Storm_Duration.unstack(
```

Aimé Fournier
May 19, 2022
Resolve

All the R^2 results are very small, correct? So really none of the models is very good.

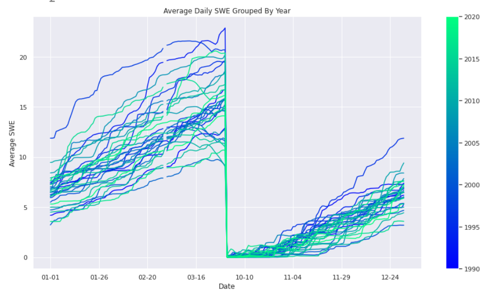
The following cells visualize each of our variables with the day of year on the x-axis and each line representing a different year. These show trends over time more clearly visually. It appears that storm duration and average daily SWE have decreased over the years, but average temperature has increased (although data still needs cleaning).

```
plt.rcParams["figure.figsize"] = [15, 8]
```

```
nValues = np.arange(1990,2021)

```

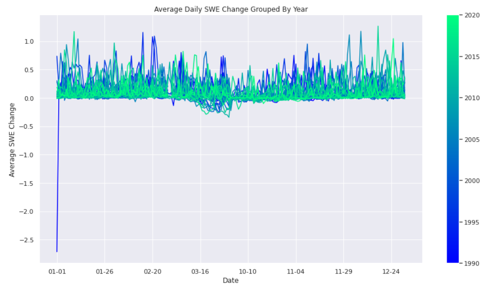
```
normalize = mcolors.Normalize(vmin=nValues.min(), vmax=nValues.max())
colormap = cm.jet

unstacked.plot(title='Average Daily SWE Grouped By Year',
               ylabel='Average SWE', cmap='winter', legend=None)

scalarmappable = cm.ScalarMappable(norm=normalize, cmap='winter')
scalarmappable.set_array(nValues)
plt.colorbar(scalarmappable)
```

```
<matplotlib.colorbar.Colorbar at 0x7f90940ef
```



```
unstacked2.plot(title='Average Daily SWE Change Grouped By Year',
               ylabel='Average SWE Change', cmap='winter', legend=None)
plt.colorbar(scalarmappable)
```

```
<matplotlib.colorbar.Colorbar at 0x7f90936ed
```

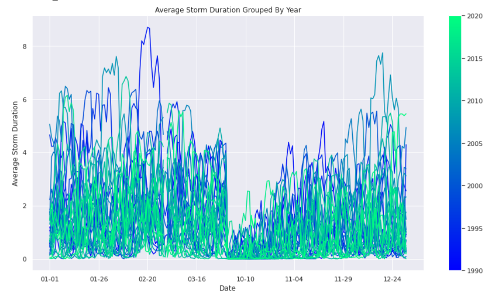


```
unstacked3.plot(title='Average Daily Air Temperature Grouped By Year',
               ylabel='Average Air Temp', cmap='winter', legend=None)
plt.colorbar(scalarmappable)
```

```
<matplotlib.colorbar.Colorbar at 0x7f90935a9
```

```
unstacked4.plot(title='Average Storm Duration Grouped By Year',
                ylabel='Average Storm Duration', cmap='winter', legend=None)
plt.colorbar(scalarmappaple)
```

```
<matplotlib.colorbar.Colorbar at 0x7f9092c57
```



Conclusion

In this study, we analyzed SWE and temperature data from weather stations across the state of Colorado to discover trends in frequency and intensity of winter storms.

Our most meaningful takeaway thus far is that there is a statistically significant negative relationship between average air temperature, storm frequency, and storm duration, as well as between air temperature and daily SWE change. By statistically significant, we mean that we can say with confidence that these trends exist. We have also discovered slight trends from our autoregressions, but they require further analysis before we draw any meaningful conclusions. We plan to formulate these conclusions by the we submit our final notebook.

While we can assess the internal validity of our results, we cannot determine whether they imply anything about climate trends outside of Colorado. We encourage our sponsors from the CAIC to further explore the relationship between temperature and winter storms, as this topic is highly relevant within the realm of climate change.