Categorizing Snow Depth Trends in Vermont with Singular Value Decomposition

Brendan Whitney
Computational Story Lab, University of Vermont

December 6, 2017

Abstract

Snow depth records for 11 weather stations in Vermont were analyzed using Singular Value Decomposition to extract the seasonal modes for snowpack shape. Linear regression on the modes revealed temporal trends. Stations located in municipalities with higher population densities exhibited higher variability in their yearly snowpack. Certain stations exhibited significant changes in reliance on a particular mode. These trends were site specific and a significant trend present at one station was most likely not exhibited by the other Vermont stations.

1 Introduction/Literature Review

General warming trends in the Northeastern United States have impacted the snowfall cycles and snowpack accumulation. Vermont relies heavily upon the tourist industry to support its economy, and the activitiy most dependent on snowfall and snowpack is skiing. The skiing industry attracts roughly 4 million visitors each winter, and employs more than 14,000 Vermonters (2.5% of Vermont's population). Snowpack depth is the measurement most connected to success in the Vermont ski industry. While snowfall is required, ski resorts cannot operate without snowpack retention and consistent depths throughout the season. Ski resorts have the ability to make snow when temperatures cooperate, which allows them to build the snowpack earlier and sustain the snowpack later into the season. However, snowmaking is an expensive undertaking, and is not feasible for an entire season. A deep natural snowpack is necessary for the financial success of ski resorts. In the present thesis, we look into the current state of Vermont snowpacks across the state using Singular Value Decomposition on historical snow depth data, but begin with a literature review.

1.1 Burakowski et al [1]

Research prior to Burakowski indicated a general trend of increasing temperature in conjunction with a reduction of snow to total precipitation ratios in Northern New England. The general warming trend has been confirmed by various other hydrological and climate

measurements. Snowpacks are getting shallower and melting earlier because of this trend of warming temperatures throughout the Northeast [2] [3].

The ski industry relies heavily on deep (> 12in), cold snowpack for revenue. Research has shown that warm and slushy winters are not only detrimental to ski resort visits and sales, but to general wintertime economic activity throughout the Northeast. Fewer resort visits lead to less traffic in towns serving said resorts, and a decrease in revenue throughout the different industries supported by resort towns. Characterizing the severity of the warming trend in the Northeast is critical to predicting the economic impact that continued warming will have on the communities supported by ski tourism.

Burakowski et al created a time series from 1965 to 2005 using stations from the Northeast with less than 10% daily precipitation values missing. For the purpose of their study, the Northeast was defined as New England, New York, New Jersey, and Pennsylvania. The data were split into 12 moving windows each of size 30 years. A moving window represents the data so that the first year of a window is one year later than the previous window. For example, if a window started in 1965, the next 30-year window would start in 1966. The 30-year windows served to remove the emphasis placed on the beginning and end of the time series. Only stations with a $p < 0.10$ in warming across all 12 decadal windows were considered statistically significant. The study found that 22 of 128 stations reported statistically significant warming across all the decadal windows. However, ony one station was shown to be statistically significant for a reduction in snow covered days.

The results from Burakowski et al provide a good reason to delve into the problem of characterizing snow depth trends for the Northeast. The research shows that portions of the region have shown statistically significant warming over the 40 year period. However, they have also shown that the warming experienced by the region has not had an impact on the number of days with snow cover. The ski industry does not just rely on having snow coverage: there needs to be a significant amount of snow on the ground to have a successful season. Even though the warming weather might not affect the number of days with snow coverage, it could impact the quality of the snow. There needs to be an exploration into the impact of warming trends on the depth of the snowpack from year to year.

1.2 Dyer and Mote [2]

Snowfall record keeping for snow depth for North America was very sparse pre-1960. Before 1960, the US and Canada had a high concentration of weather stations on the coast, but not much by way of weather stations in the prairie regions of North America. After 1960, both the US and Canada prioritized a more comprehensive weather station system allowing for better data collection. Therefore, after 1960, Dyer and Mote created a grid system spanning North America using snow depth values from the 5 nearest stations to each specific 0.25°x 0.25°. The smaller grids were then spatially averaged into 1°x 1°for analysis.

Shortcomings of the data collection include persistent sparsity issues in more remote locations of North America, where data were being collected from weather stations more than 100km away from the grid. Another data collection issue is the spatial averaging does not measure snow depth of mountain slopes and locations with heavy snow deposits from wind loading.

Data analysis consisted of linear regression performed on pentad (5-day) averages. A

two-sample difference of means test was computed for each regression to determine if there have been significant changes. The snowpack has stayed relatively constant for pentads in early (October-December) and mid-winter (January-February). Significant decreases in snow depth in late March and early April point towards an earlier spring melting cycle. The earlier melting cycle in Canada could be attributed to either a less frequent spring cyclone storm generation bringing less spring snow storms to the region, or more snow melt energy in the early spring climate system. In North America, the earlier snow melt is likely attributable to shallower snowpacks and higher early spring temperatures.

Dyer and Mote combined the snow depth measurements with Snow Coverage Extent (SCE) gathered from satellite values to map the extent of specific snowpack depth across North America. The combination of SCE and snow depth led to some interesting insights. The snow pack depth with the largest decrease in SCE was the snow pack with a depth of 40cm followed closely by the snow pack of 2cm depth. The shallower snow packs reached their peak size in early-mid January, while the deeper snow packs reach their peak size in March. Combination of SCE and snow depth further confirmed the observation that the spring melt is occuring earlier in the season.

1.3 Hamburg et al [4]

Hamburg et al used the Hubbard Brook Experimental Forest (HBEF) to analyze climate trends on the local level in comparison to trends observed at the global level. HBEF is a historically undisturbed forest with weather stations that provide extensive and nearly complete ($> 99.9\%$ complete) climate records. The study focused on 18 different weather measurements, but the results for mean annual temperature, mean winter temperature, mean summer temperature, snowpack duration, and maximum snowpack were the most relevant to my analysis. The climate measurements were taken from the longest records in areas of the forest not tampered by experimental logging practices. Temperature was measured daily, and the snowpack was analyzed weekly for depth and water content. The local values were compared against the global values observed from weather stations at Mount Washington, NH, Pinkham Notch, NH, and Hanover, NH.

The non-parametric Mann-Kendall test was used to detect decadal trends in the local data over two date ranges. The first range was the entire length of data collection for the 7 different weather stations, which varied from station to station. The second range spanned from 1966-2005, which was the longest range with complete records from every recording station. Statistical significance was set at the $p = 0.10$ threshold.

The annual mean temperature was significantly increasing in 3 of the 4 temperature recording weather stations, faster than the global data. The same ratio was observed in the trend analysis for both mean summer temperature (June-August) and mean winter temperature (December-February) with 3 of 4 weather stations indicating increasing temperature trends. The mean winter temperature is experiencing a greater increase over mean summer temperature in a non significant fashion. Additionally, the mean winter temperatures are more variable than the mean summer temperature.

The snow pack analysis focused on the first and last dates of measurable snow pack. The decadal rate of change for the last date of measurable snowpack is significantly earlier by 2.5 days per decade. The rate for first date of measurable snowpack is increasing by 1.67

days per decade, but is not significant. The net decrease in measureable snowpack days (4.2 days per decade) is significant ($p = 0.06$). Therefore at the local level, the snowpack is melting earlier and on the ground for a smaller amount of time each decade. The data collection did begin in an especially cold decade (the 1960s), which does have an influence on the impact of the warming trend indicated by the trend test. A cold decade would also lead to longer lasting snowpacks, thereby making assumptions from this data set difficult to justify. Replication of the tests with a larger and longer lasting data set would be necessary to fully justify the findings of the study.

1.4 Wobus et al [5]

Wobus et al modeled the effect of climate change on the ski industry in North America. The model utilized in the study was the Utah Energy Balance (UEB). They trained their model on 30 years of climate data collected from the North American Land Data Assimilation System (NLDAS-2). NLDAS-2 is the only multi-decadal, high-spatial resolution, continental-scale dataset, which is why it was chosen for this study. Driving the UEB model with NLDAS-2 allowed for good regional approximations across the entire continental US. This allowed for UEB to be as broad and generalizeable as possible.

The researchers optimized the model to have high computational efficiency, minimal parameters to improve the applicability across North America, and acceptable performance when validated with the Snow Data Assimilation System (SNODAS). SNODAS provides daily snow water extent, a measurement of the water content in a snowpack, at high resolution. SNODAS was used as a reliable season length source against which season predictions from UEB were validated. UEB was used to model natural snow accumulation and melt for a given year. The snow accumulation modeling was done at two different elevations representing the base and the summit of each of the 247 ski resorts analyzed in the continental United States.

UEB also modeled temperature to predict snowmaking conditions, to best predict the opening day for each resort. Snowmaking hours were modeled beginning on October 1st, and are only considered when the wet bulb temperature drops below 28°F. The opening date for ski resorts was modeled as the date that the resort reached 450 cumulative hours of snowmaking. The average start date for each resort was calculated across the 30 years of collected data. Five global climate models (GCMs) and two representative concentration pathways (RCPs) were chosen for modeling. The two RCPs were chosen to represent a modeling scenario where there is no green house gas (GHG) emission reduction over the modeled years (RCP8.5), and a modeling scenario with GHG emission reduction (RCP4.5).

The modeling results indicate a stark decrease in the ski season for continental US ski resorts. Under both RCP scenarios, the climate models predict a delay in season start date. RCP4.5 predicts a 10-20 day delay in resort openings across the US, while RCP8.5 predicts a 30-70 day delay. These results are regionally variable with the Northeast changing most drastically, and the Rocky Mountains remaining the most robust to such changes. Another important metric measured in terms of ski resort success is opening before the Christmas holiday break. Both RCP levels predict that by 2090 fewer than 25% of US ski resorts will be open by December 15th. This is a significant drop from the 70% of ski resorts that currently operate by December 15th.

The Northeast is of particular note in this study because it was the most drastically affected by climate change according the UEB model. Understanding the extent of climate change disrupting the operation of Northeastern ski resorts is important to the economies that rely on ski tourism. The modeling shows there is a need for location specific research to determine the underlying shifts in snow cover in response to a warming climate. There is a strong indication that snow packs are shrinking, but the full extent and effect of shrinking on snow pack construction remain mysteries.

1.5 Dodds et al. [6]

Dodds et al researched the concept termed the "teletherm". Their study defined a teletherm as the average hottest or coldest day of the year for a 30 year window–commonly refered to as normals from the National Oceanic and Atmospheric Administration (NOAA). 30-year averages were used instead of the raw hottest day of each year to reduce the stochastic pattern of daily temperature distributions.

The researchers encountered very similar data issues as I did. They handled varying record lengths, missing values, and periods of non-existence for certain weather stations. Additionally,values for leap days were not used in the analysis to ensure each year was 365 days. Finally, to center the coldest teletherms, they rearranged the yearly data from July to June, which mirrors the construction of one continuous winter snowfall season. Despite the research being on a different topic, the teletherm research was an important tool for insight into handling of climate data.

The researchers plotted the teletherm data for a 30 year window only if the stations had 80% of the data collected for that window. Then they fit a smoothed curve with a Gaussian kernel to best represent the underlying shape of the points, which tend to be noisy. The Gaussian kernel works by giving weights to the surrounding points relative to the distance from the point where the estimate is desired. This process is repeated for every point in the year and the resulting smoothed line is analyzed for its maximum value. The teletherm varies little, as the number of points used for Gaussian kernel smoothing increases from 7 to 31 days.

Following their characterizations of teletherms for each weather station, the researchers split their data into two adjacent 50 year windows (1912-1961 and 1962-2011 for the winter). They compared the two teletherms from the 50 year windows, and recorded the number of days the teletherm shifted and the direction of the shift. The shifts were not random, nor centered around zero, which indicates that the teletherms are shifting in a non-trivial manner. The teletherm results have good spatial representations despite the data requiring specific results for each station. This is one aspect where the snow depth analysis falls short.

2 Data

The data for analysis in the present work comes from the Climate Database Online (CDO), made available by NOAA. CDO provides a comprehensive list of all weather stations providing data for any given period of time in the United States. The data for Vermont is included only if it had 85% data coverage and more than 50 years of data collection. The

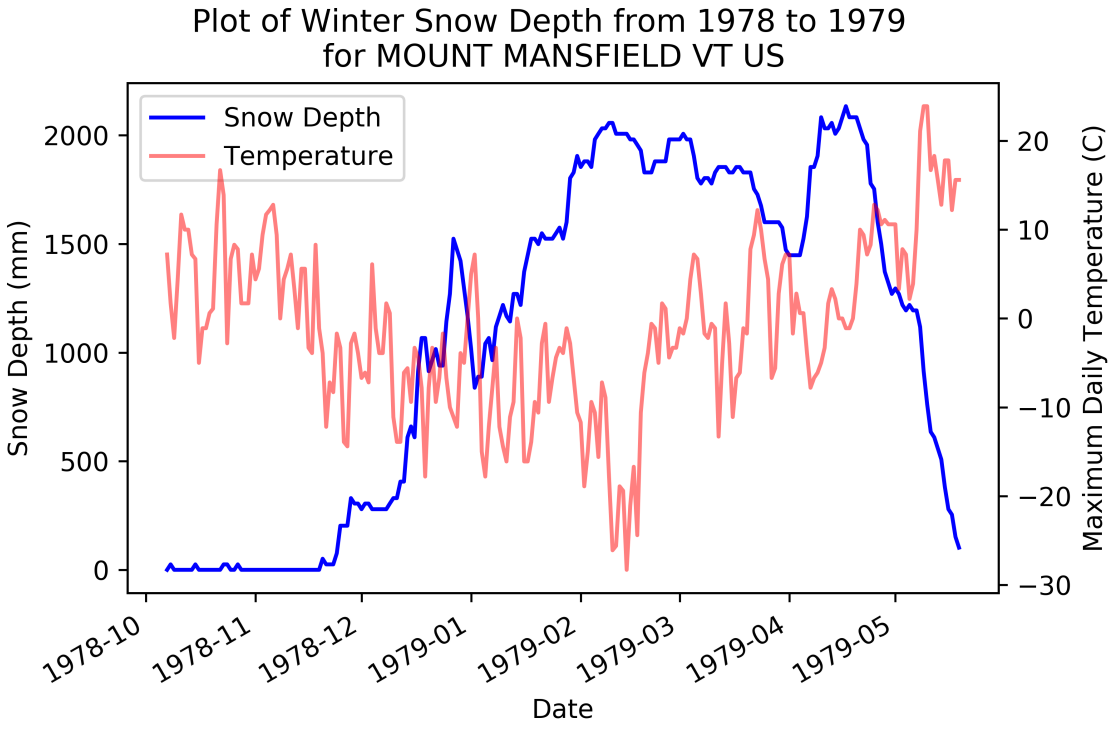


Figure 1: A sample Snow Depth plot for Mount Mansfield Weather Station, VT including daily maximum temperature values from 10/02/1978 to 05/26/1979.

coverage value, which is the percentage of missing data inputs on the daily level, on the CDO was calculated for all five datatypes satisfying the search criteria. Those datatypes were Maximum Daily Temperature (TMAX), Minimum Daily Temperature (TMIN), Daily Precipitation (PRCP), Daily Snowfall (SNOW), and Snow Depth (SNWD).

Further analysis of missing values in the individual categories, set a baseline for the stations that performed well for SNWD coverage. The analysis included graphing of the SNWD per season, with a season being defined as stretching from August to July. Figure 1 shows an example SNWD plot. Following the diagnostic analysis, visual inspection of the snow depth plots refined the usability of the weather stations gathered from the CDO. From the analysis, the weather stations from Gilman, Woodstock, and Cornwall all had missing snow depth percentages $> 45\%$. This large missing percentage in conjunction with the presence of missing values throughout the duration of record led to the determinion that the stations did not record SNWD with the completeness required for a good mathematical analysis. Other stations with large missing value percentages–Rutland (31%), Barre (28%), and Enosburg (38%)–showed long streches of uninterrupted recording, with the missing values concentrated at the beginning or end of their records (see section 2.1 for explanation of handling missing values). Those values were removed from the analysis, and the stations were included. The final 11 stations used for analysis are plotted in Figure 2.

All the data gathered from the CDO were measured in standard metric units. The temperature measurements were recorded in Celsius. The PRCP, SNOW, and SNWD values were all recorded in mm. These values are standard for weather recording in the United

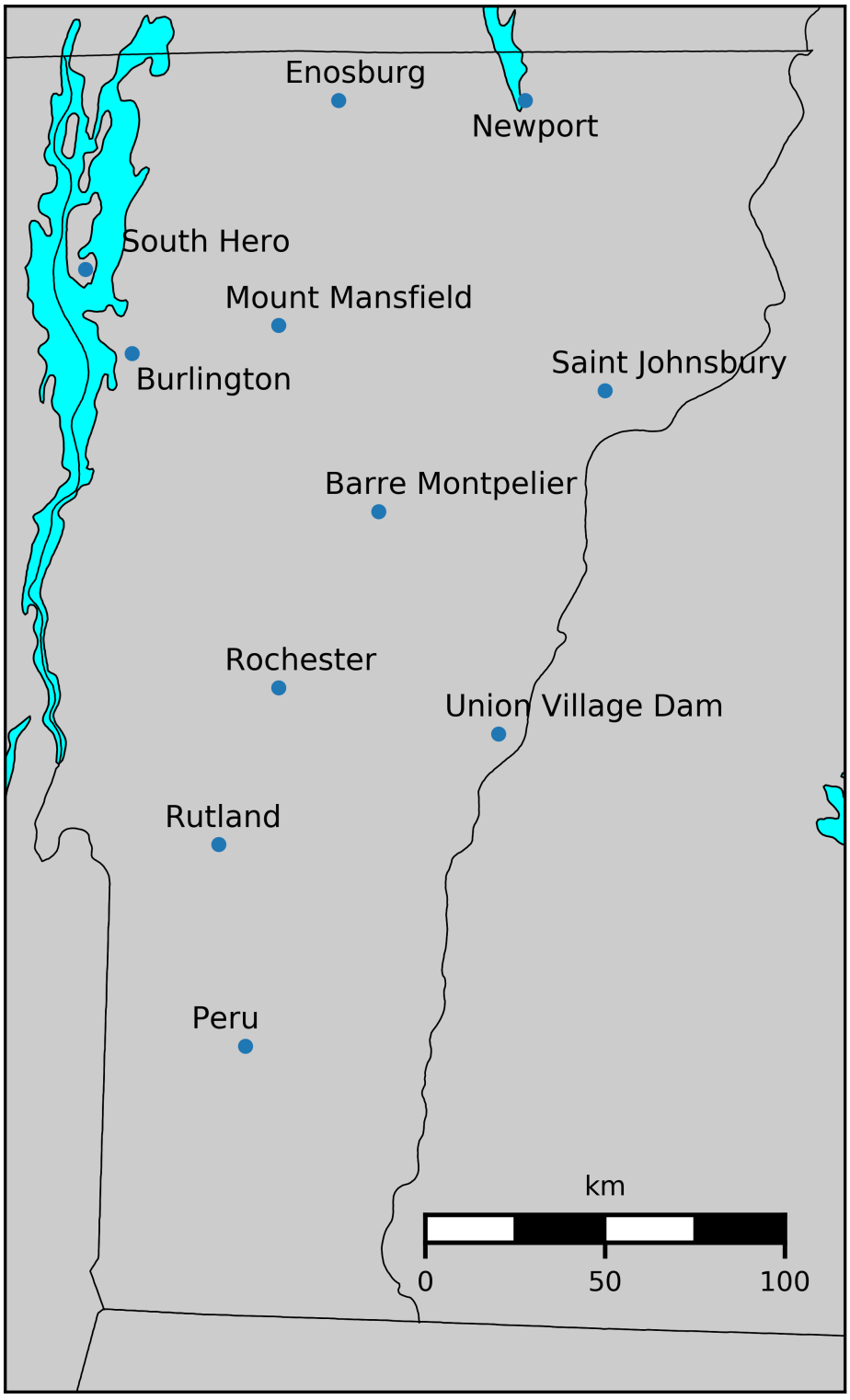


Figure 2: Locations of the 11 stations used in the present study. Map created from the CDO data.

Table 1: Length of Analyzed Record

Station	Seasons Analyzed
Barre Montpelier	1948-1994
Burlington	1948-2015
Enosburg	1948-2009
Mount Mansfiled	1955-2015
Newport	1949-2013
Peru	1948-1999
Rochester	1948-1992
Rutland	1948-2015
Saint Johnsbury	1926-2015
South Hero	1970-2015
Union Village Dam	1950-2015

States. The measurement values were not altered for the purpose of this analysis.

2.1 Handling of Missing Values

Missing values were prevalent in two different aspects of the CDO data. There were both missing recordings of daily SNWD and missing days of record from stations.

The missing daily values for SNWD were handled with simple linear interpolation if the missing value was isolated, i.e. it had a recorded value on the day before and the day after. This assumption is safe with SNWD data, because it is highly unlikely that in one given day there would be a significant snow accumulation instance, followed by a significant snowmelt to bring the total back down to the total recorded the following day. Additionally, a single day spike in the snowpack does not have a wide ranging impact on the shape of the snowpack over the course of a winter. A single day spike is noise, and does not contribute to the underlying seasonal structure of the snowpack. However, the handling of missing values for multiple days in a row proved to be more difficult. This is because multiple missing days could contain a significant accumulation and melt phase. Without sufficient radiation data, snowmelt probabilities could not be calculated [7]. Therefore, no action was taken with multiple days of missing SNWD values, and they were left in the analysis as missing values.

If the weather station had missing days of all recordings from their season, the days were filled with linear separation when graphed for SNWD. However, for the singular value decomposition (SVD), those seasons with missing records were not included. The canonical SVD requires complete years in order to work properly (See section 3).

Stations were analayzed individually to determine completeness of time series in the SVD manipulation. The time series for Barre-Montpelier was stopped after 1995 because the station no longer recorded snow depth data after the 1995 season. The time series for Rochester was shortened to only consider the seasons starting in 1948 and ending in 1992 because every season in between those dates was complete for snow depth measurements. A complete list of season lengths for the analyzed stations can be found in Table 1.

3 Methods

3.1 Singular Value Decomposition

3.1.1 Linear Algebra

For an arbitrary $m \times n$ matrix A, singular value decomposition (SVD) extracts the orthonormal left and right singular vectors and corresponding singular values associated with A. Both sets of singular vectors are unit vectors at right angles with each other. The left singular vectors form the columns of a $m \times m$ matrix U. The singular values are arranged in descending order on the diagonal of Σ, which is a $m \times n$ diagonal matrix. The right singular values sit in the columns of V, a $n \times n$ matrix. Both U and V are unitary matrices, which means that $U^{-1} = U^T$.

The construction of right singular vectors maximizes the projections of rows of A onto a unit vector v. The rows of A are considered m points in a n-dimensional space. The projection of a particular row in A, a_i onto v is $|a_i \cdot v|$. To construct the right singular vectors we want to maximize the square of the sum of all the row projections onto v, written as $|Av|^2$ [8]. The first singular vector is the vector v_1 that maximizes $|Av|^2$. The construction of the next vector chooses the vector v_2 such that $v_1 \perp v_2$, and v_2 is the orthogonal vector that maximizes $|Av|^2$. Construction of the remaining right singular vectors follows in a similar fashion, with the current vector, v_n satisfying $v_n \perp v_1, v_2, \ldots, v_{n-1}$ and maximizing $|Av|^2$

The singular values follow from the construction of the right singular vectors. Namely, the singular value for the right singular vector, σ_1, is the sum of row projections from A onto v_1, i.e $|Av_1|$. Since the construction of singular vectors are ordered by maximizing the projections, it follows that the singular values are in desceding order. Because $|Av_1| > |Av_2| > \cdots > |Av_n|$, $\sigma_1 > \sigma_2 > \cdots > \sigma_n \geq 0$.

The left singular vectors, u_i, of A are constructed by the following formula:

$$u_i = \frac{1}{\sigma_i} A v_i \text{ where } \sigma_i = 0 \text{ is handled by choosing } u_i \perp u_1, u_2, \ldots, u_{n-1}$$

The construction of left singular vectors, singular values, and right singular vectors allow for the construction of A using the following simple equation

$$A = \sum_{i=1}^{n} \sigma_i u_i v_i^T$$

3.1.2 Application to Snowpack

SVD was used to extract the season snow depth modes from each of the useful stations. In order to utilize SVD, a matrix was constructed out of the years for each weather station. The matrix has 365 rows, one for each day of recorded snow depth. The number of columns was dependent on the number of seasons with complete yearly recordings for each weather station. To ensure that each year was 365 days, all years with a leap day had the recording for February 29th removed. Seasons with missing recordings that could not be resolved using the interpolation method described in the Section 2.1 were not included in the matrix.

The constructed matrix (A) was then decomposed into three different matrices using SVD. The left matrix (U) is called the left singular matrix. U is a unitary matrix with 365 rows and 365 columns. Each column represents the left singular vector component of our constructed matrix. The singular vectors in U are ranked in order of importance to the construction of A. The singular vectors contained within U are the unweighted representations of the modes for snow depth for the station.

The middle matrix (Σ) is a 365 by number of years matrix with the singular values on the diagonal and 0s elsewhere. The singular values give the appropriate weight to the importance of the left and right singular vectors. The larger the singular value, the more the corresponding singular vectors contribute the construction of A. The singular values are ranked along the diagonal in descending order.

The right matrix (V^T) is the right singular matrix. V^T is a unitary matrix with rows and columns equal to the number of complete years recorded by the particular weather station it is representing. The rows of V^T contain the right singular vectors responsible for the construction of A.

For mode analysis, $Z = \Sigma V^T$ was calculated as a representation of the coefficients of the mode's importance to each year constructing the A matrix. The weights in Z transform the left singular vector to the corresponding mode acting on that particular year. The weights construct A as follows:

$$Z[i, j] \cdot U[:, i] = \text{Construct the ith mode for the jth year of } A$$

Adding each weighted left singular vector, i.e each mode, for a particular season constructed the complete snow depth profile for that season. For a station matrix A with n years, the construction of complete season for a particular year j, denoted A_j is

$$A_j = \sum_{i=1}^{n} Z[i, j] \cdot U[:, i]$$

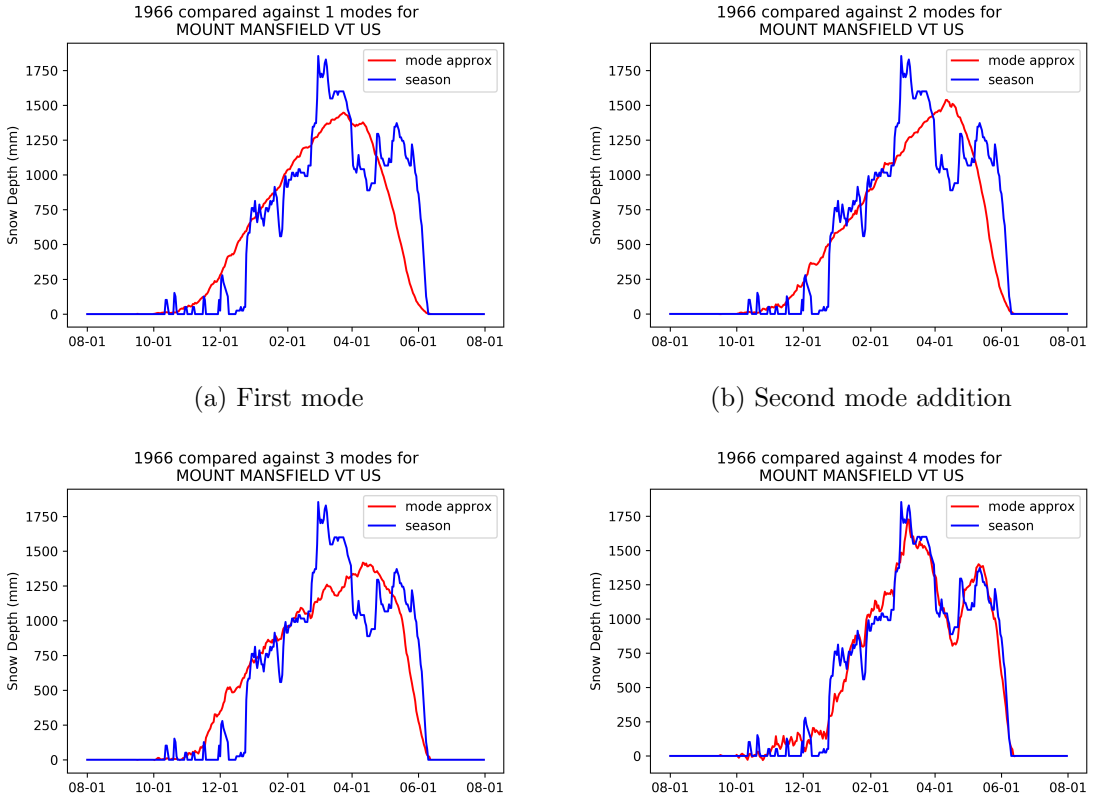
The above method was adapted from a paper that used mode reconstruction to explain common story arcs [9]. See Figure 3 for an example of mode construction for a given season.

3.1.3 Explained Variance

In order to determine the variance explained by each seasonal mode, the singular value for each mode was divided by the sum of the singular values for that particular station.

$$\text{Explained Variance} = \frac{\sum_{i=1}^{p} \sigma_i}{\sum_{i=1}^{n} \sigma_i}$$

A useful measure of mode importance is the calculation of the cumulative variance explained by the modes. For example, we are interested in the explained variance of the first five modes added together, then we add the first five singular values together and divide by the sum of all the singular values. See Figure 4 for an example of the explained variance plots. For modal analysis, a threshold of 90% explained variance was chosen.



(a) First mode

(b) Second mode addition

(c) Third mode addition

(d) Fourth mode addition

Figure 3: Demonstration of mode addition for Mount Mansfield, VT during the 1966-1967 season. The blue curve represents the actual recorded snow depth for the season. The red curve represents the mode approximations for the season.

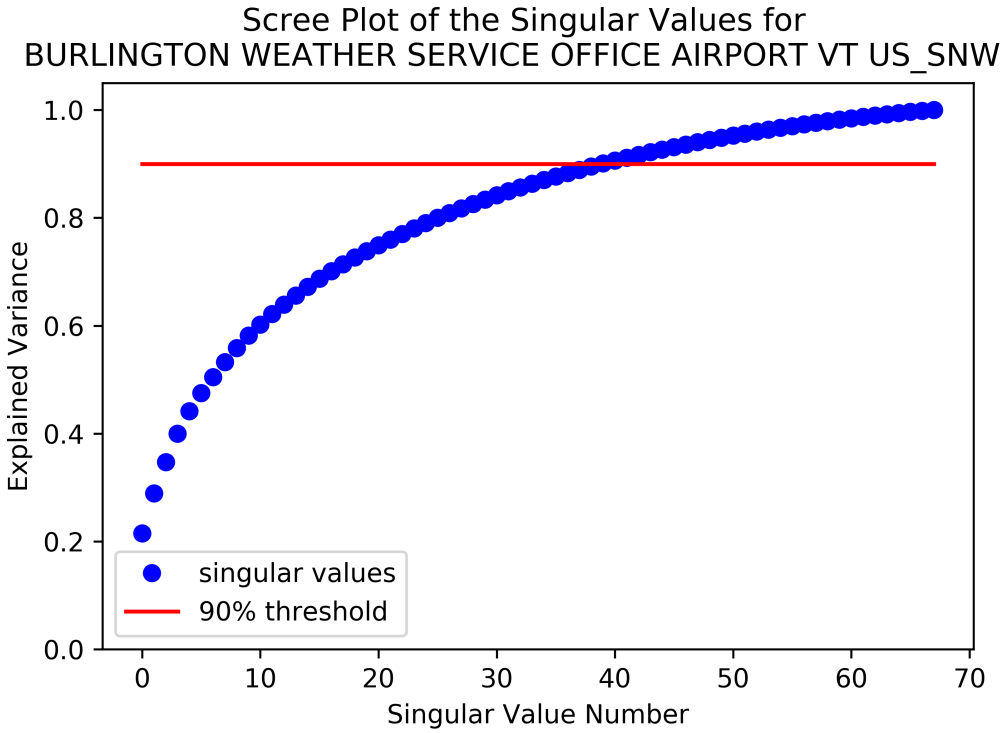


Figure 4: Explained Variance plot for the Burlington, VT weather station

3.2 Linear Regression

The entries of the Z matrix described in the project specifications section provide insight into the importance of a particular mode to a given year. In order to quantify the trend of modal importance, the absolute values of the entries of the Z matrix were used for linear regression. The entries in Z can take on both positive and negative values depending on the influence of a left singular value. A negative value simply indicates the year requiring construction with a negative mode instead of a positive one. The influence of a particular mode changes with a negative value, but a larger number indicates a more substantial influence on seasonal construction regardless of the coefficient taking on a positive or negative value. Therefore, despite the difference in interpretation between a positive and negative entry in the Z matrix, the absolute values of the entries were utilized to assess modal importance.

Statistical significance of yearly trends in modal importance were reported at a $p = 0.05$ level. A different regression model was created for each mode, and for each station independently of one another. This was due to the orthogonality between the modes inherent in the use of singular value decomposition. Therefore, it would not be reasonable, statistically, to combine similar modes from different stations, i.e. the transformation coefficients of mode 1 for Mansfield were not combined with the transformation coefficients of mode 1 for Burlington.

4 Results

The modal constructions, see Figure 3 for a modal construction example, of each season were starkly different for individual stations across Vermont. Analysis of one station did not directly provide insight to analysis of a different station, even if that station was nearby. This station to station independence is likely due to the number of weather phenomena snow pack construction relies upon, and the spatial variability of these phenomena in Vermont. The creation of stable snowpacks requires long continuous periods of freezing temperatures, and weather events generating significant snowfall events during that cold strecth. Once a snowpack has become stable, it can withstand short warming fronts with little snowmelt or reduction. Ideal snowpack conditions generally require less energy than the requirements for an ideal snowfall event. Additional requirements for snow pack retention beyond temperature include limited thermal and solar radiation, wind interaction, and precipitation added to snowpack (rain) [7].

Vermont experiences a lot of variation of these measurements across the state due to its variations in elevation, terrain types, and proximity to bodies of water. This results in varying snow packs for locations in Vermont that are very close to each other. The Green Mountains heavily influence the variation observed between stations. In conjunction with the prevailing Westerly winds, the mountain spine influences the amount of precipitation experienced on either side of the range and at the summits. Another huge factor for spatial variability is Lake Champlain, which typically moderates the weather experienced along the western edge of Vermont. From the east, large energetic storms move in and deposit large amounts of snow throughout the winter. These storms push their way up to the Green Mountains were they dissipate, which results in larger snowfall events to the east of the Green Mountains. In the case of easterly storms, the Green Mountains act as a wall stopping the storms from heavily impacting the western parts of the state.

4.1 Winter Variability

The number of modes required to reach the 90% threshold of explained variance provide insight into the seasonal variability of each station. The more modes required to reach the threshold, the more seasonal variability that is observed at that weather station. Results are shown in Table 2.

4.2 Mode Trends

Analysis of mode trends, indicate relatively few significant trends in mode influence values. Linear trend regression was performed on the first 5 modes for the 11 VT weather stations in the analysis. Table 3 on page 17 contains the slope coefficients for regression and the corresponding p-value. Significant trends are highlighted in gray.

Understanding the meaning of significant changes in influence values over time depends on the sign of the coefficients, and how the sign alters the influence of that particular mode on season construction. Figure 5 shows an example of the trend analysis for Saint Johnsbury. The example trend indicates a non-significant increasing trend in mode 2. Mode 2 is more

Table 2: Modes Required to explain 90% variance per station

Station	Modes
Barre Montpelier	27
Burlington	40
Enosburg	24
Mount Mansfiled	24
Newport	31
Peru	19
Rochester	23
Rutland	35
Saint Johnsbury	43
South Hero	26
Union Village Dam	26

important in construction of snow pack of more recent years than the earlier years of record for Saint Johnsbury, but with little statistical support.

Trend analysis indicates a significant decreasing trend for mode 2 for Mount Mansfield (see Figure 6). More recent snow packs on Mansfield depend less on the shape of mode 2 for snowpack construction. Figures 7 and 8 on pages 19 and 20 respectively show the unweighted mode shape for the first five modes for Mansfield. Using the shape of unweighted modes, a decreasing trend for mode 2 indicates one of two possible interpretations. For years with a positive coefficient value, smaller influence values indicates a decrease in the number of late season snow accumulation (i.e. a mid-April) snow accumulation event. For years with a negative coefficient value, smaller influence values indicate a reduction in the amount of faster early season accumulation and early April snowmelt. Analysis of the trend graphs for mode 2 and mode 3 indicate that the influence values for mode 3 are larger in more recent years than the influence values for mode 2. Therefore, mode 3 is explaining more variance for recent snow packs than mode 2.

4.3 Limitations

It is important to note that other modes can compensate for the decreasing importance for a particular mode. For instance, while mode 2 indicates a smaller chance of a late season snow event, or earlier spring melting, mode 3 for Mansfield has a similar shape as mode 2 at the end of the season. Therefore, depending on the influence value for mode 3, the influence of mode 2 could be overridden or canceled out.

Another shortcoming of seasonal mode analysis falls to the construction method of adding modes together to create each season. Therefore, for instance, the influence of adding mode 4 to a particular year not only depends on the influence value, but the prior construction of the season. Different additions of modes 1 through 3 affects the impact that mode 4 has to a particular season. Look at modal construction of Mount Mansfield using the unweighted mode shapes as seen in Figures 7 and 8 on pages 19 and 20. Depending on the sign of a transformation coefficient, mode 2 either indicate a late increase in snowpack, or an early melt. Then after adding either mode 2 shape to the first mode, the addition of mode 3

could indicate another late season snow pack increase, or even earlier snow melt. The two could possibly cancel each other out. There are many different possibilities for just 3 modes, nevermind 53. Considering the combination of modes could change from season to season, analysis is difficult to generalize to each season for a particular station.

5 Conclusion

The results from linear trend analysis indicate that while modes are changing for some stations in Vermont, they aren't changing in a manner consistent across stations. The trends for modal importance seem to behave stochastically, and are heavily dependent on the station for which the modes are calculated. Interpretation of the modes proved difficult and only applicable to the station for which the mode was calculated. Calculating the number of modes required to explain the variance of a particular gave insight into the variability of the snowpack at a given station.

Future research into mode construction could categorize the seasons that are most adherent to each mode, i.e. which seasons are most similar to the first mode, second mode, etc. Perhaps there will be a more spatially quantifiable results for the years that adhere most to certain modes. For instance, it could be the case that across Vermont, years 1980-1990 drive the creation of a certain mode for each weather station. Another topic for future research could average snow pack construction over a few years for a particular station and determine the modal construction of these averages. This approach could certainly reduce the variability of year to year construction and potentially extract a more meaningful trend analysis of modal importance. From this approach, it might be possible to determine which mode drives the characteristic snow pack for a period of time in Vermont.

Climate models are predicting less snowfall in conjunction with higher temperatures for the Northeast. An example of this trend was shown in the paper written by Wobus et al. Understanding the snowpack change in light of the predicted climate change for the Northeast could result in very interesting results predicting the change in snow pack shape for winter seasons. The problem of changing snow pack is vital to the success of the ski industry, and requires a combination of mode construction methods, and climate model predictions.

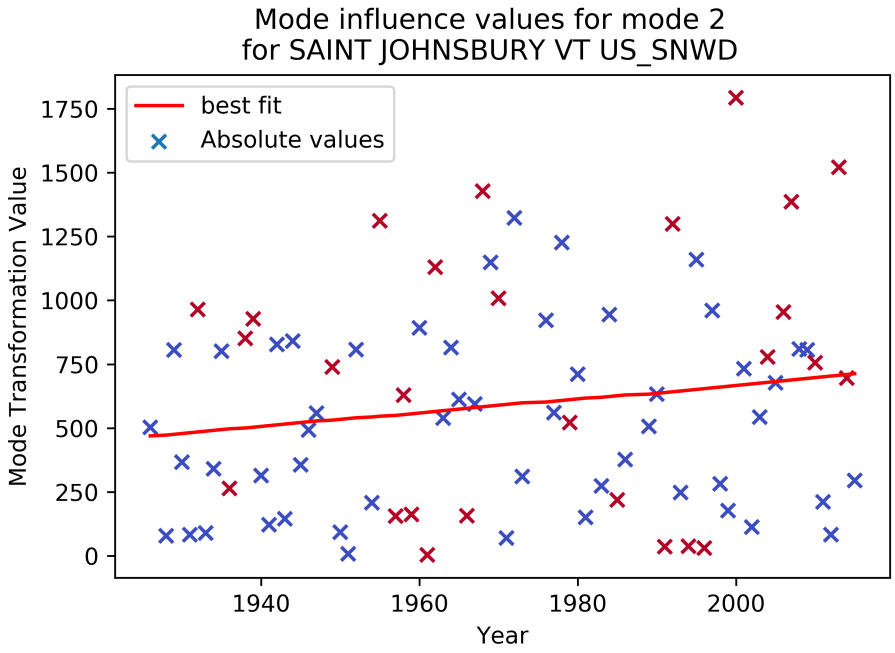


Figure 5: Linear trend of influence values for mode 2. Blue x indicates a negative coefficient, and red x indicates a positive coefficient. This mode is not experiencing a significant change in time.

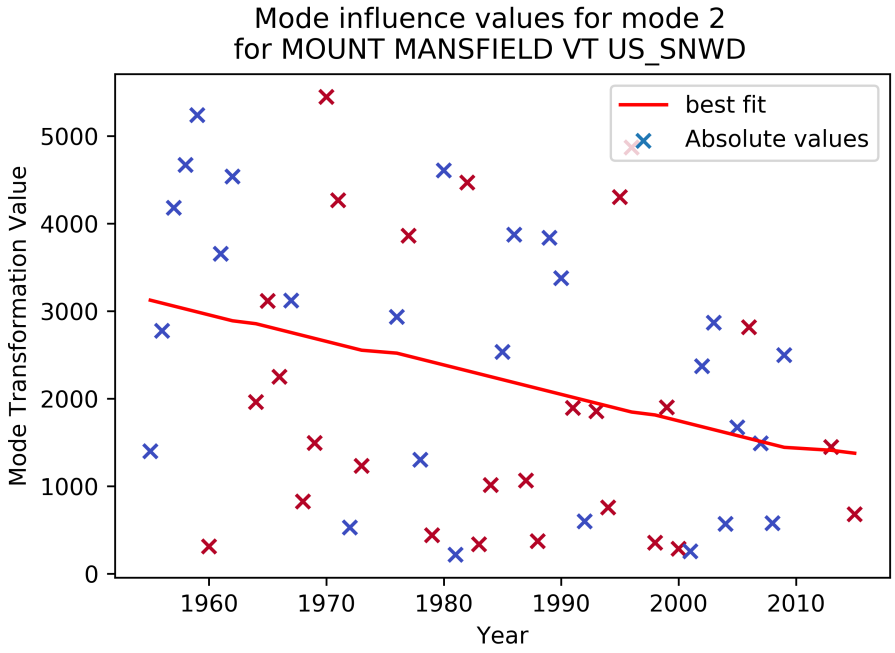


Figure 6: Linear trend of influence values for mode 2. Blue x indicates a negative coefficient, and red x indicates a positive coefficient. This mode is reducing in importance over time.

Table 3: Modular Trend Analysis

Station	Mode	Slope	p-value
Barre Montpelier	1	-27.76	0.267
	2	-6.14	0.348
	3	-7.19	0.160
	4	-10.06	0.037
	5	-0.94	0.820
Burlington	1	0.84	0.887
	2	2.09	0.406
	3	-0.68	0.717
	4	4.26	0.013
	5	-0.32	0.816
Enosburg Falls	1	-9.39	0.450
	2	-0.74	0.888
	3	-6.20	0.137
	4	4.34	0.167
	5	-3.48	0.199
Mount Mansfield	1	65.59	0.184
	2	-33.62	0.016
	3	3.51	0.675
	4	-2.83	0.712
	5	0.30	0.955
Newport	1	-15.09	0.304
	2	8.97	0.017
	3	4.33	0.129
	4	2.32	0.237
	5	-0.59	0.774
Peru	1	-9.21	0.811
	2	0.23	0.989
	3	9.55	0.277
	4	-2.26	0.773
	5	-0.96	0.891
Rochester	1	18.93	0.496
	2	4.93	0.587
	3	-0.61	0.909
	4	7.58	0.072
	5	1.10	0.773
Rutland	1	9.73	0.198
	2	5.88	0.025
	3	5.13	0.019
	4	1.02	0.513
	5	2.11	0.111

Station	Mode	Slope	p-value
Saint Johnsbury	1	-3.03	0.667
	2	3.08	0.135
	3	0.27	0.881
	4	-0.27	0.801
	5	0.57	0.605
South Hero	1	-19.14	0.107
	2	-10.22	0.043
	3	-2.56	0.551
	4	-7.37	0.103
	5	-3.24	0.302
Union Village Dam	1	-2.32	0.898
	2	-0.80	0.872
	3	-6.96	0.060
	4	-5.45	0.035
	5	1.14	0.668

Trend calculations for the first 5 modes of each weather station analyzed.

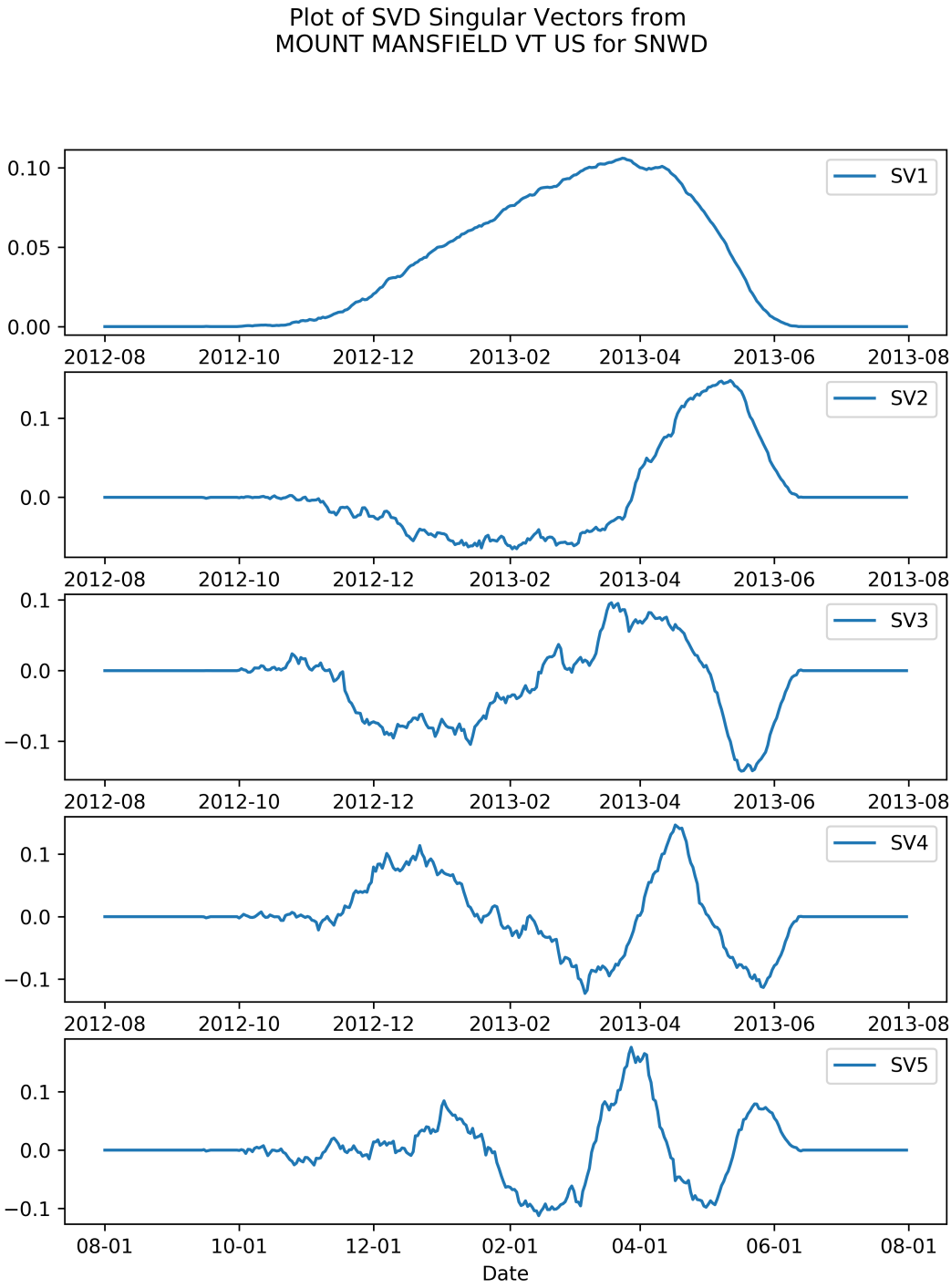


Figure 7: Positive coefficient mode vectors for Mount Mansfield

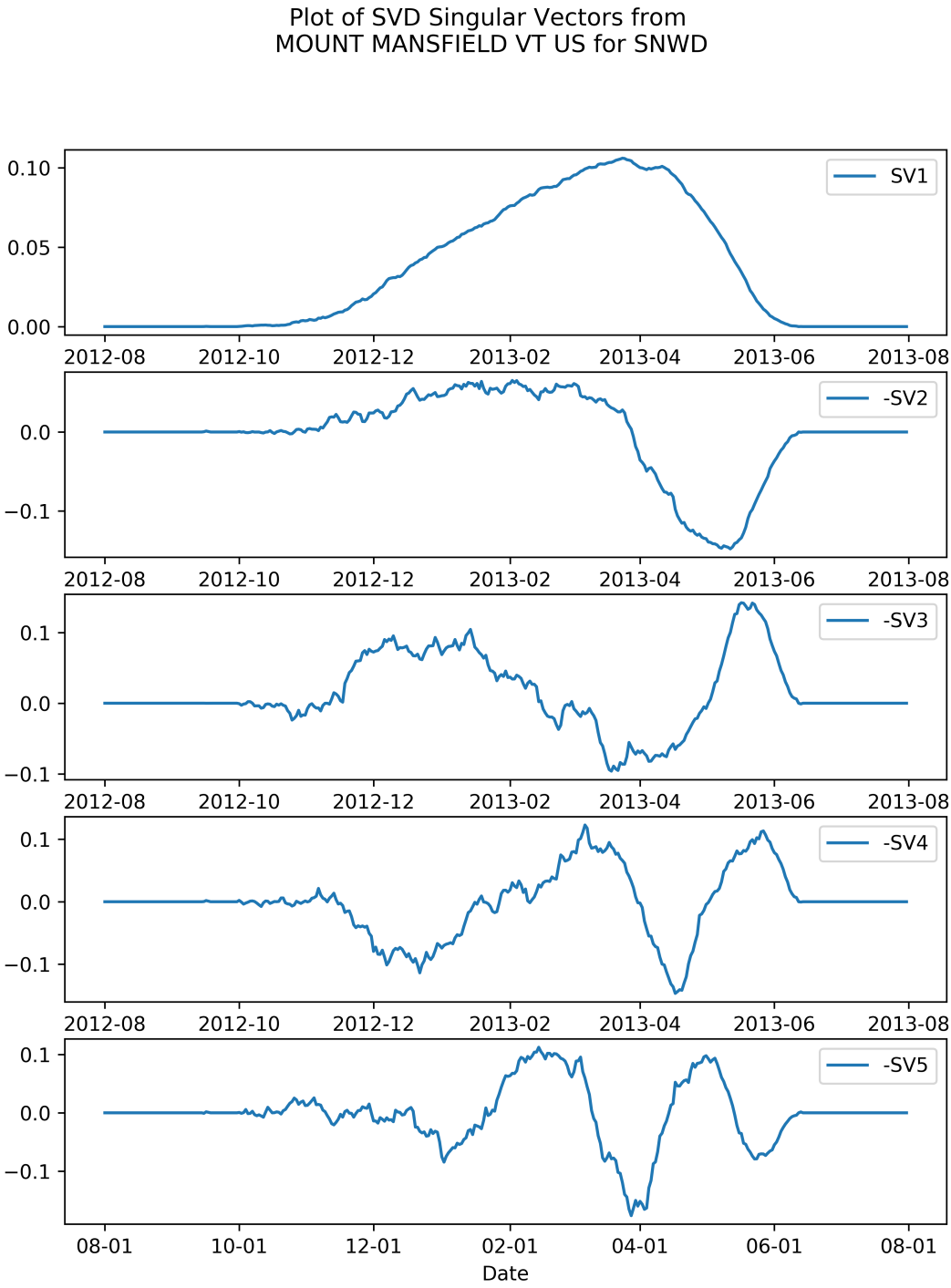


Figure 8: Negative coefficient mode vectors for Mount Mansfield

References

[1] E. A. Burakowski, C. P. Wake, B. Braswell, and D. P. Brown, "Trends in wintertime climate in the northeastern united states: 1965–2005," *Journal of Geophysical Research: Atmospheres*, vol. 113, no. D20, 2008. D20114.

[2] J. L. Dyer and T. L. Mote, "Spatial variability and trends in observed snow depth over north america," *Geophysical Research Letters*, vol. 33, no. 16, 2006. L16503.

[3] R. D. Brown, "Northern hemisphere snow cover variability and change, 1915–97," *Journal of Climate*, vol. 13, no. 13, pp. 2339–2355, 2000.

[4] S. P. Hamburg, M. A. Vadeboncoeur, A. D. Richardson, and A. S. Bailey, "Climate change at the ecosystem scale: a 50-year record in new hampshire," *Climatic Change*, vol. 116, pp. 457–477, Feb 2013.

[5] C. Wobus, E. E. Small, H. Hosterman, D. Mills, J. Stein, M. Rissing, R. Jones, M. Duckworth, R. Hall, M. Kolian, J. Creason, and J. Martinich, "Projected climate change impacts on skiing and snowmobiling: A case study of the united states," *Global Environmental Change*, vol. 45, no. Supplement C, pp. 1 – 14, 2017.

[6] P. S. Dodds, L. Mitchell, A. J. Reagan, and C. M. Danforth, "Tracking climate change through the spatiotemporal dynamics of the teletherms, the statistically hottest and coldest days of the year," *PLOS ONE*, vol. 11, pp. 1–20, 05 2016.

[7] U. S. S. C. Service, *National Engineering Handbook. Part 630, Hydrology*. 1985.

[8] J. Hopcroft and R. Kannan, "Computer science theory for the information age," 2012.

[9] A. J. Reagan, L. Mitchell, D. Kiley, C. M. Danforth, and P. S. Dodds, "The emotional arcs of stories are dominated by six basic shapes," *EPJ Data Science*, vol. 5, p. 31, Nov 2016.

A Source Code

```
# -*- coding: utf-8 -*-
"""
Created on Wed Aug 30 11:22:27 2017

@author: brendan

This file will run diagnostics to test the viability of data
"""
import pandas as pd
import datetime as dt
# Make a function that keeps track of number of data points and missing data
def data_check_year(data):
    #create an empty dataframe to store the diagnostic values in
    diag_yr = pd.DataFrame()
    # Generate a list of stations to iterate through
    stations = list(data['STATION_NAME'].unique())
    for station in stations:
        # subset to just a dataframe of the given station
        dfcheck= data[data['STATION_NAME'] == station]
        #create a column of years determined from the datetime feature
        dfcheck['YEAR'] = pd.DatetimeIndex(dfcheck['DATE']).year
        # generate a list of years to iterate through
        years = list(dfcheck['YEAR'].unique())
        for year in years:
            #subset further to a dataframe for just the year
            itdf = dfcheck[dfcheck['YEAR']==year]
            #calculate days, missing values, and percentage of missing values
            days = len(itdf['DATE'])
            missing = len(itdf['DATE'])-itdf.count()
            mperc = itdf.isnull().sum()/days*100
            #assign the missing values and percentages to indiviual variables
            m_prcp, m_snwd, m_snow, m_tmax, m_tmin = missing[2:-1]
            p_prcp, p_snwd, p_snow, p_tmax, p_tmin = mperc[2:-1]
            #add all the variables as a dataframe line and append them to the
            #empty data frame
            ins = pd.DataFrame({'STATION':station,'YEAR':year,'NUM_DAYS':days,
                                'MISSING_PRCP':m_prcp,'MISSING_SNWD':m_snwd,
                                'MISSING_SNOW':m_snow,'MISSING_TMAX':m_tmax,
                                'MISSING_TMIN':m_tmin,'PERCENT_PRCP':p_prcp,
                                'PERCENT_SNWD':p_snwd,'PERCENT_SNOW':p_snow,
                                'PERCENT_TMAX':p_tmax,'PERCENT_TMIN':p_tmin},
                                 index=[0])
            diag_yr = diag_yr.append(ins, ignore_index=True)

    #rearrange the columns to make them look pretty
    cols=diag_yr.columns.tolist()
    cols = cols[-2:]+cols[5:6]+cols[:5]+cols[6:-2]
    diag_yr=diag_yr[cols]
    print(daig_yr)
    diag_yr.to_csv('data_frames/yearly_diagnostics.csv',index=False)
```

```
def data_check(data):
    #create an empty dataframe to store the diagnostic values in
    diag=pd.DataFrame()
    # Generate a list of stations to iterate through
    stations = list(data['STATION_NAME'].unique())
    for station in stations:
        # subset to just a dataframe of the given station
        dfcheck= data[data['STATION_NAME'] == station]
        days = len(dfcheck['DATE'])
        missing = len(dfcheck['DATE'])-dfcheck.count()
        mperc = dfcheck.isnull().sum()/days*100
        #assign the missing values and percentages to indiviual variables
        m_prcp, m_snwd, m_snow, m_tmax, m_tmin = missing[2:]
        p_prcp, p_snwd, p_snow, p_tmax, p_tmin = mperc[2:]
        #add all the variables as a dataframe line and append them to the
        #empty data frame
        ins = pd.DataFrame({'STATION':station,'NUM_DAYS':days,
                            'MISSING_PRCP':m_prcp,'MISSING_SNWD':m_snwd,
                            'MISSING_SNOW':m_snow,'MISSING_TMAX':m_tmax,
                            'MISSING_TMIN':m_tmin,'PERCENT_PRCP':p_prcp,
                            'PERCENT_SNWD':p_snwd,'PERCENT_SNOW':p_snow,
                            'PERCENT_TMAX':p_tmax,'PERCENT_TMIN':p_tmin},
                             index=[0])
        diag = diag.append(ins, ignore_index=True)
    #rearrange the columns to make them look pretty
    cols=diag.columns.tolist()
    cols = cols[-1:]+cols[5:6]+cols[:5]+cols[6:-1]
    diag=diag[cols]
    print(diag)
    diag.to_csv('data_frames/diagnostics.csv',index=False)


data = pd.read_csv("data_frames/final_vermont.csv", na_values= -9999)
data.drop(['STATION','ELEVATION','LATITUDE','LONGITUDE'],axis=1,inplace=True)

data_check_year(data)
data_check(data)
```

```
# -*- coding: utf-8 -*-
"""
Created on Thu Mar 30 21:31:40 2017

@author: brendan

This code organizes the raw data into the form that is later used to plot
the snow depth and tmax for each year for each station.
"""
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import os
import pathlib as pl
import re


mypath = "data_frames/snwd_temp_comparison/"
destination_path = 'plots/snwd_temp_data/'
datatypes = ['STATION_NAME','DATE','SNWD','TMAX']

#save the data to a csv and create a new directory if it doesn't already exist
def file_save(filepath,file, data):
    pl.Path(filepath).mkdir(parents=True, exist_ok=True)
    data.to_csv(filepath+file, index=False)

def assess_outliers(data):
    snow = np.squeeze(np.asarray(data['SNWD']))
    #We can ignore the two endpoints for assessment because they were zeros
    #added in by the code prior to assessing for outliers
    for i in range(len(snow)-1):
        if i==0:
            continue
        x = snow[i]-snow[i-1]
        y = snow[i+1]-snow[i]
        if x>0:
            #if the snowfall increases by 1000 and drops by 1000 the next day
            #classify the point as an outlier
            if (np.abs(x)>1000 and np.abs(y)>1000):
                snow[i] = np.NaN
            else:
                continue
        if x<0:
            #if the snowdepth goes down to zero and back up the next day
            #classify it as an outlier
            if (np.abs(x/snow[i-1])==1 and np.abs(y/snow[i+1])==1):
                snow[i] = np.NaN
            #else if the snowdepth drops 500 and rises 500 the next day,
            #classify it as an outlier
            elif (np.abs(x)>500 and np.abs(y)>500):
                snow[i] = np.NaN
    data['OUT'] = snow
    #interpolate the NaN values linearly since each day is represented by the
    #data
```

```
    data['SNWD'] = data['OUT'].interpolate()
    data['TMAX'] = data['TMAX'].interpolate()
    return data[datatypes]

def depthplot(data, org_file):
    #get the indexes of the first and last rows with snow on the ground
    snow = data['SNWD']
    first_snow = min(min((snow>0).nonzero()))
    last_snow = max(min((snow>0).nonzero()))
    #subset the winter by these values and add one on either side to better
    #complete the graph
    winter = data.iloc[first_snow-1:last_snow+1,:]
    winter = assess_outliers(winter)

    file_save(destination_path, org_file, winter)

#turn the text file path into something the os can read
directory = os.fsencode(mypath)

for file in os.listdir(directory):
   filename = os.fsdecode(file)
   data = pd.read_csv(mypath+filename)
   if 'MANSFIELD' in filename:
       try:
           depthplot(data, filename)
       #if a value error is raised just print the filename instead of saving it
       #in the file. This just means all the snow values equal 0 or are
       #missing values
       except ValueError:
           error_path = "plots/snwd_temp_data/value_errors/"
           file_save(error_path,filename, data)
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Sep 27 21:57:42 2017

@author: brendan

Visualize the snow depth and max temp data for each season using the snow depth
data generated by classify_snowd.py
"""
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import os
import pathlib as pl

mypath = 'plots/snwd_temp_data/'
destination_path = 'plots/snwd_temp_plots/'

def depthplot(data):
    #get the station name and year for the title of the plot
    station = list(data['STATION_NAME'].unique())
    year = min(pd.DatetimeIndex(data['DATE']).year)

    fig, ax1 = plt.subplots()
    #initialize the first axis and plot the snow depth on this axis in blue
    ln1 = ax1.plot_date(data['DATE'],data['SNWD'],'b-',label='Snow Depth')
    ax1.set_xlabel('Date')
    ax1.set_ylabel('Snow Depth (mm)')
    ax1.tick_params('y')
    ax1.legend()

    #initialize a second y-axis on the same x-axis to plot the maximum temp
    ax2 = ax1.twinx()
    ln2 = ax2.plot_date(data['DATE'], data['TMAX'], 'r-',
                  label='Temperature', alpha=0.5)
    ax2.set_ylabel('Maximum Daily Temperature (C)')
    ax2.tick_params('y')
    #Combine the two lines on different axes into one legend
    lns = ln1+ln2
    labs = [l.get_label() for l in lns]
    ax1.legend(lns, labs, loc='upper left')
    #make the x axis look nice
    plt.gcf().autofmt_xdate()
    plt.suptitle("Plot of Winter Snow Depth from "+str(year)+" to "+str(year+1)
    +"\n for "+station[0])

    filename = (station[0]+'_'+str(year)+'-'+str(year+1)+'.png')
    #create directory if it doesn't exist and save file as .png
    pl.Path(destination_path).mkdir(parents=True, exist_ok=True)
    plt.savefig(destination_path+filename, dpi=600)
    plt.close(fig)

#turn the text file path into something the os can read
```

```
directory = os.fsencode(mypath)

for file in os.listdir(directory):
    filename = os.fsdecode(file)
    try:
        data = pd.read_csv(mypath+filename, engine = 'python')
    except IsADirectoryError:
        continue
    try:
        depthplot(data)
    except ValueError:
        error_path = 'plots/snwd_temp_plots/error/'
        file_save(error_path,filename, data)
```

```
# -*- coding: utf-8 -*-
"""
Created on Fri Oct 13 07:49:50 2017

@author: brendan

Perform singular value decomposition after handling outliers in the data
and organizing the original data into seasons years instead of calendar years.
Then save the resulting matrices into appropriate folders for later use.
"""

import pandas as pd
import numpy as np
import pathlib

final_path = 'data_frames/svd_data/'
datatypes = ['STATION_NAME','DATE','SNWD','TMAX']

#save the data to a csv and create a new directory if it doesn't already exist
def file_save(filepath,file, data):
    pathlib.Path(filepath).mkdir(parents=True, exist_ok=True)
    data.to_csv(filepath+file,index=False)

def process_svd(svd_data,station, datatype):
    #interpolate over the missing values, and perform svd over the matrix of
    #the values from the dataframe
    evals = svd_data.interpolate()
    if (evals.isnull().values.any()):
        evals = evals.dropna(axis=1, how='any')
    try:
        #perform SVD and save the data to the appropriate files
        U,s,V = np.linalg.svd(np.matrix(evals.values))
        left = pd.DataFrame(U)
        eigenvals = pd.DataFrame(s)
        right = pd.DataFrame(V)
        file = station+'_'+datatype
        file_save(final_path+'originals/',file, evals)
        file_save(final_path+'left_singular/', file, left)
        file_save(final_path+'right_singular/', file, right)
        file_save(final_path+'evalues/', file, eigenvals)
    except:
        print(station, datatype)

def assess_outliers(data):
    snow = np.squeeze(np.asarray(data['SNWD']))
    #We can ignore the two endpoints for assessment because they were zeros
    #added in by the code prior to assessing for outliers
    for i in range(len(snow)-1):
        if i==0:
            continue
        x = snow[i]-snow[i-1]
        y = snow[i+1]-snow[i]
        if x>0:
            #if the snowfall increases by 1000 and drops by 1000 the next day
```

```
            #classify the point as an outlier
            if (np.abs(x)>1000 and np.abs(y)>1000):
                snow[i] = np.NaN
            else:
                continue
        if x<0:
            #if the snowdepth goes down to zero and back up the next day
            #classify it as an outlier
            if (np.abs(x/snow[i-1])==1 and np.abs(y/snow[i+1])==1):
                snow[i] = np.NaN
            #else if the snowdepth drops 500 and rises 500 the next day,
            #classify it as an outlier
            elif (np.abs(x)>500 and np.abs(y)>500):
                snow[i] = np.NaN
    data['OUT'] = snow
    #interpolate the NaN values linearly since each day is represented by the
    #data
    data['SNWD'] = data['OUT'].interpolate()
    data['TMAX'] = data['TMAX'].interpolate()
    return data[datatypes]

def separate_years(data,station):
    #Set the indexes to run from August 1st to July 31st
    dates = pd.date_range('2012-08-01', periods =365, freq='D').to_series()
    indexes = dates.dt.strftime('%m-%d')
    svd_snwd = pd.DataFrame(index=indexes)
    svd_tmax = pd.DataFrame(index=indexes)
    years = list(pd.DatetimeIndex(data['DATE']).year.unique())
    for year in years:
        #These logical statements give a year that goes from July to August
        #wrapping through the winter. Very useful for winter long graphs
        data_winter = data[
               ((pd.DatetimeIndex(data['DATE']).year == year-1) &
                (pd.DatetimeIndex(data['DATE']).month>=8)) |
                ((pd.DatetimeIndex(data['DATE']).year == year) &
                 (pd.DatetimeIndex(data['DATE']).month <= 7))]
        #remove leap year days to get uniform 365 day years
        data_winter = data_winter[
               ~((pd.DatetimeIndex(data_winter['DATE']).month == 2) &
                 (pd.DatetimeIndex(data_winter['DATE']).day == 29))]
        #make sure the dataframe to be added to svd has exactly 365 days
        if (len(data_winter.index)==365):
            data_winter = assess_outliers(data_winter)
            col_years = list(
                    pd.DatetimeIndex(data_winter['DATE']).year.unique())
            svd_snwd[str(col_years[0])] = data_winter['SNWD'].values
            svd_tmax[str(col_years[0])] = data_winter['TMAX'].values

    #handle case specific data shortages for both Barre and Rochester where
    #the values are all NaN or 0 outside of the specified ranges
    if 'BARRE' in station:
        svd_snwd = svd_snwd.loc[:,'1948':'1995']
    if 'ROCHESTER' in station:
        svd_snwd = svd_snwd.loc[:,'1948':'1992']
```

```
    process_svd(svd_snwd, station, 'SNWD')

sep = pd.read_csv('data_frames/final_vermont.csv')
stations = list(sep['STATION_NAME'].unique())
for station in stations:
    sep_1 = sep[sep['STATION_NAME']== station]
    data = sep_1[['STATION_NAME', 'DATE', 'SNWD', 'TMAX']]
    separate_years(data, station)
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Oct 17 19:05:17 2017

@author: brendan

Generate the mode transformations in comparison to the particular year.  Plot
each iteration of mode addition to visualize the progress of adding each mode
to the previous modes.  This where we construct the season mode by mode to show
how the modes are related to each season for each station.
Additionally, keep track of the transformation variables for the linear model.
"""
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import pathlib as pl

evalues_path = 'data_frames/svd_data/evalues/'
uvectors_path = 'data_frames/svd_data/left_singular/'
vvectors_path = 'data_frames/svd_data/right_singular/'
original_path = 'data_frames/svd_data/originals/'
plot_destination = 'plots/mode_plots/'
file_destination = 'data_frames/svd_data/plot_data/'

#pairs up the eigenvectors with the appropriate eigenvalues by matching
#the two filenames
def get_evalues(filename):
    evalues_directory = os.fsencode(evalues_path)
    for file in os.listdir(evalues_directory):
        evalues_filename = os.fsdecode(file)
        if filename == evalues_filename:
            try:
                evalues = pd.read_csv(evalues_path+evalues_filename,
                                      engine = 'python')
                return evalues
            except IsADirectoryError:
                continue

#returns the original data that generated the eigenvectors by the same method
#used by get_evalues
def get_original_data(filename):
    orig_directory = os.fsencode(original_path)
    for file in os.listdir(orig_directory):
        orig_filename = os.fsdecode(file)
        if filename == orig_filename:
            try:
                original = pd.read_csv(original_path+orig_filename,
                                       engine = 'python')
                return original
            except IsADirectoryError:
                continue
```

```
#returns the right singular vectors by matching the two filenames
def get_v(filename):
    v_directory = os.fsencode(vvectors_path)
    for file in os.listdir(v_directory):
        v_filename = os.fsdecode(file)
        if filename == v_filename:
            try:
                v = pd.read_csv(vvectors_path+v_filename,
                                engine = 'python')
                return v
            except IsADirectoryError:
                continue

#create a plot of the modes and each season, keeping track of how many
#modes there are
def make_mode_plot(mode, uvector, original, station, datatype):
    uvec = np.squeeze(np.asarray(uvector))
    dates = pd.date_range('2012-08-01', periods =365, freq='D').to_series()
    x = np.squeeze(np.asarray(dates))
    fig, ax = plt.subplots()

    ax.plot(x,uvec,'r-', label='mode approx')
    ax.plot(x,original, 'b-', label = 'season')
    ax.set_ylabel('Snow Depth (mm)')
    ax.legend()

    plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%m-%d'))
    plt.suptitle(original.name+' compared against %s modes for \n'%(mode+1)+station)

    filename = (datatype+'_'+original.name+'_%s'%(mode+1)+'.png')
    destination_path = (plot_destination+station+'/')
    #create directory if it doesn't exist and save file as .png
    pl.Path(destination_path).mkdir(parents=True, exist_ok=True)
    plt.savefig(destination_path+filename, dpi=600)
    plt.close(fig)

#save the data to a csv and create a new directory if it doesn't already exist
def file_save(filepath,file, data):
    pl.Path(filepath).mkdir(parents=True, exist_ok=True)
    data.to_csv(filepath+file,index=False)

def manipulate(original,evalues,uvectors, vvectors, station, datatype):
    uvectors = np.matrix(uvectors.values)
    tuvectors  = uvectors.T
    vvectors = np.matrix(vvectors.values)
    evals = np.squeeze(np.asarray(evalues.values))
    #put the singular values in a diagonal matrix
    sig = np.diag(evals)
    #make the matrix that has the coefficients to build each year out of the
    #modes, which are contained in the left singular matrix (u)
    mode_mult = sig*vvectors
    modes = 5
    years = len(original.columns)
```

```
    vals_mat = np.matrix(np.empty((modes,years)))
    columns = list(original)

    for i in range(years):
        #initialize the plot values for the mode in question
        plot_u = np.zeros((1,365))
        for j in range(modes):
            #add the transformed variables to plot vector and then plot it
            #for each mode. In this constructions the modes add onto each other
            plot_u += mode_mult[j,i]*tuvectors[j,:]
            season = original.iloc[:,i]
            make_mode_plot(j,plot_u,season, station, datatype)
            #keep track of the transformation variables
            vals_mat[j,i] = mode_mult[j,i]
    #make a dataframe of the transformation variables, and save them for
    #later visualization.
    vals_df = pd.DataFrame(vals_mat, columns = columns)
    file_save(file_destination+'Manipulation_Values/', filename, vals_df)

directory = os.fsencode(uvectors_path)

for file in os.listdir(directory):
   filename = os.fsdecode(file)
   components = filename.split('_')
   station = components[0]
   datatype = components[1]
   try:
       uvectors = pd.read_csv(uvectors_path+filename, engine='python')
       evalues = get_evalues(filename)
       original = get_original_data(filename)
       vvectors = get_v(filename)
       manipulate(original,evalues,uvectors,vvectors, station, datatype)
   except IsADirectoryError:
       continue
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Oct 13 09:35:22 2017

@author: brendan

Plot both the original unweighted and the negative unweighted modes for the
5 most important modes of each station.
"""

import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import pandas as pd
import os
import pathlib as pl

mypath = 'data_frames/svd_data/left_singular/'
destination_path = 'plots/svd_plots/'

def depthplot(data,filename):
    name_data=filename.split('_')
    station=name_data[0]
    datatype=name_data[1]

    dates = pd.date_range('2012-08-01', periods =365, freq='D').to_series()

    raw_data = np.matrix(data.values)
    raw_data = raw_data.T
    fig1,ax = plt.subplots(5, figsize  = (8,10))

    for i in range(5):
        if (i==0):
            #flip the first mode regardless because all of the transformation
            #values are negative for every station. Flipping it makes it more
            #intuitive, doesn't change analysis
            plot_data = np.squeeze(np.asarray(-raw_data[i]))
        else:
            plot_data = np.squeeze(np.asarray(raw_data[i]))
        indexes = np.squeeze(np.asarray(dates.values))
        #initialize the first axis and plot the snow depth on this axis in blue
        ax[i].plot(indexes,plot_data,label='SV'+str(i+1))
        ax[i].legend()
    #put the x label below all 5 plots
    ax[4].set_xlabel('Date')
    #make the dates look better
    plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%m-%d'))
    plt.suptitle("Plot of SVD Singular Vectors from \n"+station+' for '+datatype)

    filename = (station+'_'+datatype+'original.png')
    #create directory if it doesn't exist and save file as .png
    pl.Path(destination_path).mkdir(parents=True, exist_ok=True)
    plt.savefig(destination_path+filename, dpi=600)
```

```
    plt.close(fig1)

    #make a second file that will have the negative
    fig2, ax = plt.subplots(5, figsize=(8,10))
    for i in range(5):
        plot_data = np.squeeze(np.asarray(-raw_data[i]))
        indexes = np.squeeze(np.asarray(dates.values))
        #initialize the first axis and plot the snow depth on this axis in blue
        if (i==0):
            ax[i].plot(indexes,plot_data,label='SV'+str(i+1))
        else:
            ax[i].plot(indexes,plot_data,label='-SV'+str(i+1))
        ax[i].legend()

    ax[4].set_xlabel('Date')

    plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%m-%d'))
    plt.suptitle("Plot of SVD Singular Vectors from \n"+station+' for '+datatype)

    filename = (station+'_'+datatype+'negative.png')
    #create directory if it doesn't exist and save file as .png
    pl.Path(destination_path).mkdir(parents=True, exist_ok=True)
    plt.savefig(destination_path+filename, dpi=600)
    plt.close(fig2)


#turn the text file path into something the os can read
directory = os.fsencode(mypath)

for file in os.listdir(directory):
   filename = os.fsdecode(file)
   try:
       data = pd.read_csv(mypath+filename, engine='python')
   except IsADirectoryError:
       continue
   depthplot(data,filename)
```

```
# -*- coding: utf-8 -*-
"""
Created on Thu Nov  2 22:01:33 2017

@author: brendan

Generate the data and then visualize the explained variance plots. In addition
print out the number of modes that explain 90% of the variance.
"""

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import pathlib as pl
import os

directory_path = 'data_frames/svd_data/evalues/'
destination_path = 'plots/scree_plots/'
#create a scree plot by transforming the singular values to eigenvalues for
#each weather station matrix
def scree_plot(data,filename):
    #square all the data points to get the eigenvalues of the covariance
    #matrix, which will explain the variance by each number
    sums = np.squeeze(np.asarray(np.cumsum(np.power(data,1))))
    var= []
    for i in range(len(sums)):
        var.append(sums[i]/sums[-1])
        if (sums[i]/sums[-1]>0.9):
            print(filename, i+1)
    line = 0.9*np.ones(len(data))
    fig, ax = plt.subplots()
    ax.plot(np.arange(len(data)),var, 'bo',label='singular values')
    ax.plot(np.arange(len(data)),line, 'r-', label='90% threshold')
    ax.set_xlabel('Singular Value Number')
    ax.set_ylabel('Explained Variance')
    ax.set_ylim(0,1.05)
    ax.legend()
    plt.suptitle('Scree Plot of the Singular Values for\n'+filename)
    #create directory if it doesn't exist and save file as .png
    pl.Path(destination_path).mkdir(parents=True, exist_ok=True)
    plt.savefig(destination_path+filename, dpi=600)
    plt.close(fig)


directory = os.fsencode(directory_path)

for file in os.listdir(directory):
    filename = os.fsdecode(file)
    try:
        data = pd.read_csv(directory_path+filename, engine='python')
        scree_plot(data, filename)
    except IsADirectoryError:
        continue
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Thu Nov  2 22:42:45 2017

@author: brendan

Generate the linear regression data, and plot it for the best five modes
saved by the svd_manipulation source code.
"""

import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import pandas as pd
import os
import pathlib as pl
import statsmodels.api as sm
import matplotlib

data_path = 'data_frames/svd_data/plot_data/Manipulation_Values/'
destination_path = 'plots/linear_trend_plots/'

#save the data to a csv and create a new directory if it doesn't already exist
def file_save_model(file, data):
    pl.Path(destination_path).mkdir(parents=True, exist_ok=True)
    data.to_csv(destination_path+file,index=False)

def linear_regression(data,filename):
    #take the absolute value of all the data
    abslns = [np.abs(data.iloc[i,:]) for i in range(len(data.index))]
    #create a map to track which values are positive and which values are
    #negative. Negative values are assigned a 0, positive assigned a 1
    cmap = data.copy()
    cmap[cmap<0]=0
    cmap[cmap>0]=1
    indexes = data.columns
    #for each mode (rows of data), create an array of constants and unknown
    #coefficients for regression
    for i in range(len(data.index)):
        X_org = np.arange(len(abslns[i]))
        X = sm.add_constant(X_org)
        y = abslns[i]
        #fit the linear model, and then use the fit to predict y values
        model = sm.OLS(y,X).fit()
        predictions = model.predict(X)
        #plot the data with a map to show which values were positive and which
        #values were originally negative
        fig, ax = plt.subplots()
        ax.scatter(indexes, abslns[i],c= cmap.iloc[i,:],marker = 'x',
                   cmap = plt.cm.coolwarm, label = 'Absolute values')
        #plot the linear best fit line
        ax.plot(indexes[X_org], predictions, 'r-', label = 'best fit')
        ax.legend()
```

```
        ax.set_xlabel('Year')
        ax.set_ylabel('Mode Transformation Value')
        plt.suptitle('Mode influence values for mode %s\n'%(i+1)+'for '+filename)
        pl.Path(destination_path).mkdir(parents=True, exist_ok=True)
        plt.savefig(destination_path+filename+'_mode%s'%(i+1), dpi=600)
        plt.close(fig)
        #print the regression summary to the console for manual recording into
        #a table in the final product
        print(filename+"%s:\n"%(i+1), model.summary())

#turn the text file path into something the os can read
directory = os.fsencode(data_path)

for file in os.listdir(directory):
    filename = os.fsdecode(file)
    try:
        data = pd.read_csv(data_path+filename, engine='python')
    except IsADirectoryError:
        continue
    for station in stations:
        if station in filename:
            linear_regression(data,filename)
```