8-2019

Automation of State Climate Office Processes & Products: Developing Efficient Approaches for Data Dissemination

Michael Shoop
East Tennessee State University

Automation of State Climate Office Processes & Products: Developing Efficient Approaches for Data Dissemination

---

A thesis

presented to

the faculty of the Department of Geosciences

East Tennessee State University

In partial fulfillment

of the requirements for the degree

Master of Science in Geosciences

---

by

Michael Shoop

August 2019

Andrew Joyner, Ph.D., Chair

Ingrid Luffman, Ph.D.,

William Tollefson.

Keywords: Python, Automation, GIS, Climate, web application, Tennessee, GDAL, Arcpy

ABSTRACT

Automation of State Climate Office Processes & Products: Developing Efficient Approaches for Data Dissemination

by

Michael Shoop

State Climate Offices (SCO's) in the United States are critical conduits for improving weather and climate data in local communities. Two states do not have a state-recognized SCO: Tennessee and Massachusetts. Efforts are underway at East Tennessee State University to develop the Tennessee Climate Office (TCO). Currently, climate services and products are severely lacking across Tennessee. This thesis provides an improved methodology for an existing TCO product and outlines the development of a new product using Python scripting. Daily storm reports within the monthly climate report are automated and a Weather Forecasts Hazard Index (WFHI) web application is developed. Both products utilize data from the National Oceanic and Atmospheric Administration (NOAA), with the automated daily storm reports providing substantial time savings and the WFHI providing a high resolution web application for emergency managers and others to interpret potentially hazardous forecasts for extreme temperatures, high winds, snowfall/ice accumulation, and tornado/hail events.

ACKNOWLEDGEMENTS

I have a lot of people to thank for completing this thesis. I should start by thanking my advisor Dr. Andrew Joyner for helping me so much in developing this thesis, and believing in what I was doing. A great deal of thanks needs to go out to Vincent Thompson who spent many hours helping me through technical difficulties. A thanks to my committee for meticulously reviewing my work. To my parents Nancy and Jeff Shoop for supporting me. To my sister Sara Graham and my cousin Leah Bailey for helping me believe I could go to graduate school and succeed. Finally to Kim Blazzard for always being there for me through thick and thin on this journey.

TABLE OF CONTENTS

Page

LIST OF TABLES

Table Page

LIST OF FIGURES

LIST OF ABBREVIATIONS

Acronym	Compoud Term	Acronym	Compoud Term
AASC	American Association of State Climatologists	NASA	National Aeronautics and Space Administration
ACIS	Applied Climate Information System	NCDC	National Climate Data Center
AGOL	ArcGIS Online	NCEI	National Centers for Environmental Information
ARSCO	AASC Recognized State Climate Office	NCEP	National Centers for Environmental Prediction
ASCII	American Standard Code for Information Interchange	NDFD	National Digital Forecast Database
AWIPS	Advanced Weather Interactive Processing System	NDGD	National Digital Guidance Database
CEMA	Center for Environmental Monitoring & Analysis	NOAA	National Oceanic and Atmospheric Administration
COOP	Cooperative Observer Program	NWS	National Weather Service
CSD	Climate Services Division	RCC	Regional Climate Center
DBOFS	Delaware Bay Operational Forecast System	SCO	State Climate Office
DOS	Disk Operating Systems	SPC	Storm Prediction Center
DWHI	Delaware Weather Hazard Index	SQL	Structured Query Language
EF	Enhanced Fujita Scale	SRCC	Southern Regional Climate Center
FTP	File Transfer Protocol	TCO	Tennessee Climate Office
GMT	Greenwich Mean Time	TDL	Techniques Development Laboratory
GPS	Global Positioning System	TIFF	Tagged Image File Format
GRIB	General Regularly-distributed Information in Binary	TVA	Tennessee Valley Authority
IFPS	Interactive Forecast Preparation System	URL	Uniform Resource Locator
LSR	Local Storm Report	USGS	United States Geological Survey
MDL	Meteorological Development Laboratory	UTC	Universal Coordinated Time
MPH	Miles Per Hour	WFHI	Weather Forecast Hazard Index
MRCC	Midwest Regional Climate Center	WFO	Weather Forecast Office
MXD	Map Exchange Document	WGS84	World Geodetic System 1984
NAD83	North American Datum 1983	WMO	World Meteorological Organization
		WRCC	Western Regional Climate Center

CHAPTER 1

INTRODUCTION

Background

Improving access to climate and weather data is essential for short- and long-term planning, emergency response, hazard mitigation, and multi-sector impact awareness (Arzberger et al. 2004, Coletti et al. 2013, Sawale and Gupta 2013). State Climate Offices (SCO's) in the United States are critical conduits for improving data access, providing context for local weather events and trends, better understanding of climate impacts on the state's economy, and data-informed product development that helps improve decision-making in local communities (AASC… 2019). Rapidly evolving hardware and software technologies greatly increase SCO-level value-added product development and dissemination. Climate and weather data can now be processed and analyzed more efficiently than ever, allowing SCO's to fulfill their National Climate Services Partnership core mission areas of extension, research, and education with greater precision and flexibility that support local communities across the state (Brooks 2013, Climate Service Partners 2018).

While a SCO utilizes both climate and weather data, it is important to distinguish the difference between the two terms and to distinguish the difference between a National Weather Service (NWS) Weather Forecast Office (WFO) and a SCO. Weather describes short-term atmospheric events or patterns, including current conditions (e.g., raining, 55°F, northeast winds at 10 mph) and near-term forecast (e.g., 7-day outlook). We most often associate 'weather' with forms of media such as our local news station broadcast meteorologists or the Weather Channel. Climate describes long-term patterns, trends, and anomalies – essentially a culmination of multiple weather events and conditions over a long period of time. NWS WFO's primarily focus

on short-term weather forecasts and weather event preparedness (e.g., for an approaching hurricane, warnings are issued with specific storm surge and wind details along with evacuation advisement and other messages). A SCO and WFO often collaborate on projects that require both weather and climate data analysis.

Within climate science, comparisons are often made in a historical context. For example, based on 30-year climate records (called 'normals'), the average high temperature in November at a given location should be 60°F and the average low temperature should be 40°F, but on a given day the high/low temperatures may be 50°/30°, indicating below average temperatures on that day. Climate data analysis allows us to evaluate how short-term conditions (weather) deviate from long-term patterns and conditions (climate) (e.g., was this year wetter than average?) (e.g., Jones et al. 2009, Karl et al. 2012). Climate also describes long-term patterns beyond those of just temperature and precipitation. SCO's and climatologists study a wide variety of phenomena including long-term tornado patterns (e.g., Dixon et al. 2011, Dixon et al. 2014), drought anomalies (e.g., Cook et al. 2011, Karl et al. 2012, Logan et al. 2010, McCabe et al. 2008), tropical cyclone landfall patterns (e.g., Keim et al. 2007, Moore et al. 2011, Nogueira et al. 2011, Nogueira et al. 2013, Needham et al. 2014), teleconnection patterns like El Niño/La Niña (e.g., Hoerling et al. 1997, Müller et al. 2008, Stoner et al. 2009) and the Atlantic Multi-decadal Oscillation (Dima et al. 2007), and local and regional rainfall return periods (e.g., Faiers et al. 1994, Powell et al. 2015). Without climate data analysis and services, engineers would not know how to design a bridge to withstand a 1% rainfall/flood event (i.e., a 100-year event) because they would not know the 1-hour and 24-hour rainfall return periods (e.g., Bonnin et al. 2004). Additionally, engineers and planners would not know when those thresholds were exceeded without in-depth climatological analyses of weather events.

Beyond climate and weather differences, it is important to understand the general hierarchy of climate data services from the federal level to the state/local level (Figure 1).

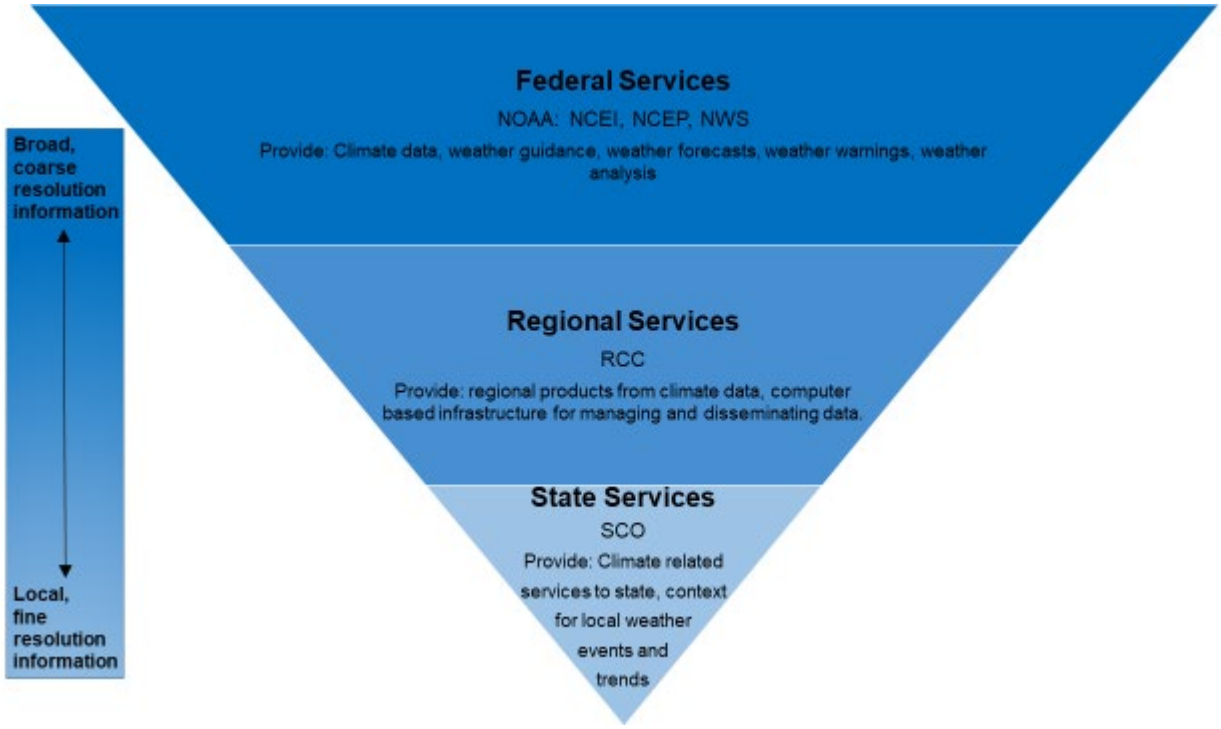


Figure 1. National weather/climate data services hierarchy

At the federal level, nearly all climate data are developed and maintained by the National Oceanic and Atmospheric Administration (NOAA). Other agencies provide some climate data services and analysis such as the National Aeronautics and Space Administration (NASA) and the United States Geological Survey (USGS), but for the purpose of this thesis, NOAA is recognized as the primary federal climate data curator and provider. Within NOAA, many divisions and departments focus on climate and weather. The National Centers for Environmental Information (NCEI, formerly the National Climate Data Center (NCDC)), National Centers for Environmental Prediction (NCEP), and the NWS provide the bulk of climate and weather data and services. NCEI, located in Asheville, North Carolina, "is

responsible for preserving, monitoring, assessing, and providing public access to the Nation's treasure of climate and historical weather data and information" (NOAA NCEI Mission Statement). NCEI maintains the world's largest climate data archive, impacting nearly every sector of the U.S. economy via services, partnerships, and product development. NCEP is composed of nine centers across the U.S. and is responsible for providing weather guidance, forecasts, warnings, and analyses to partners and users. Collectively, the centers form part of the NWS, which itself represents 122 WFO's, 13 River Forecast Centers, nine National Centers, and other support offices across the country. The NWS collects and analyzes more than 76 billion observations and releases about 1.5 million forecasts and 50,000 warnings each year. All three entities (NCEI, NCEP, NWS) share substantial resources to develop the National Digital Forecast Database (NDFD), which is a suite of products generated by the NWS using data from regional WFOs and the NCEP (National Oceanic and Atmospheric Administration 2019).

Regional Climate Centers (RCC) (NCEI partners) are managed by NCEI. The RCCs are partnered with NWS's Climate Services Division (CSD) and work with organizations such as the American Association of State Climatologists (AASC). The RCC program was initiated in 1978 under the National Climate Program Act. The first formally designated RCC was the Western Regional Climate Center (WRCC) in Reno, Nevada, established in 1986. By 1990 all six RCCs were established. Of the six regional climate centers, five evolved out of State Climate Programs (Degaetano 2010). State-level RCC alignment is shown in Figure 2.

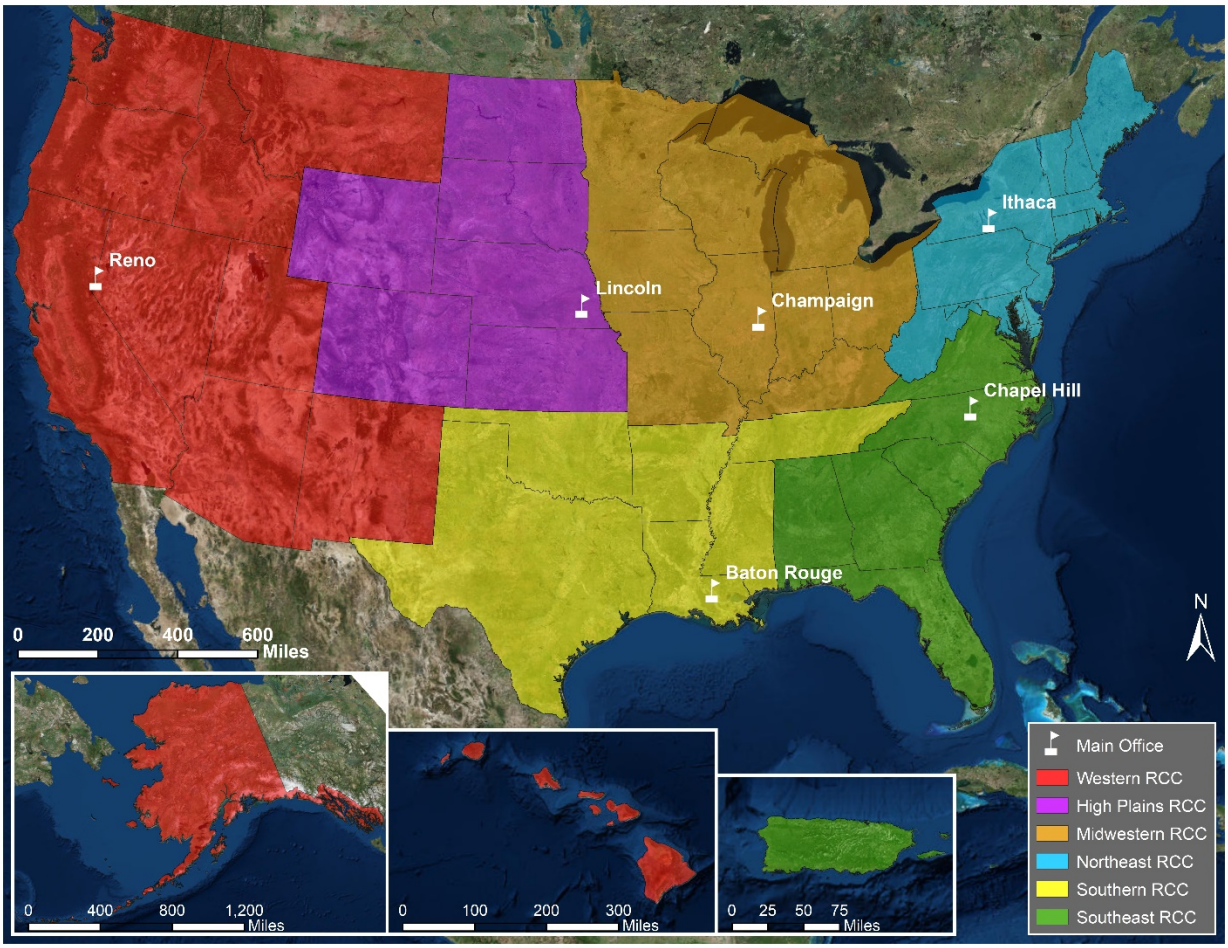


Figure 2. RCC regions and the location of RCC offices

RCCs accomplish two important National Climate Services roles. The first role is the provision or development of customer-specific products from collected data. The West Nile Virus Risk Model developed by the Midwestern Regional Climate Center (MRCC) is one example of this type of product (Degaetano 2010). Another is the Tropical Desk web application developed by the Southern Regional Climate Center (SRCC). Which shows historical and current Atlantic basin tropical cyclone activity (Interactive… 2007). A final example is the Climate Perspective (CLIMPER) tool developed by the Southeast Regional Climate Center

(SERCC). This tool shows current or forecast precipitation and recent temperature in relation to historical data for that same time of year (Climate…2019).

The second important role is maintenance of a computer-based infrastructure for managing and disseminating data. An example of this is the Applied Climate Information System (ACIS) maintained by the RCCs. This system is designed to manage and disseminate collected climate data from federal, regional, state, and local networks. These data represent a combination of historical and near-real time climate data. ACIS uses multiple data sources to produce products for end users (ACIS 2017). RCCs also develop and operate the primary data management software for the National Weather Service Cooperative Observer Program (COOP). The daily climate observations collected by the COOP are a primary source of weather data in the United States (ACIS 2017).

Originally federally funded and a program of the NWS, the SCOs were disbanded in 1973 when each state government was asked to support a state climatologist at the local level to provide the services eliminated by the NWS. Most SCOs re-established themselves immediately or within a few years, often at public universities and sometimes at state agencies. Most commonly, formal agreements or memorandums were signed between the university/agency, state government (governor and/or legislature), and NOAA's NCDC/NCEI. Currently, there are 49 SCOs in the United States, including one in Puerto Rico (Figure 3).

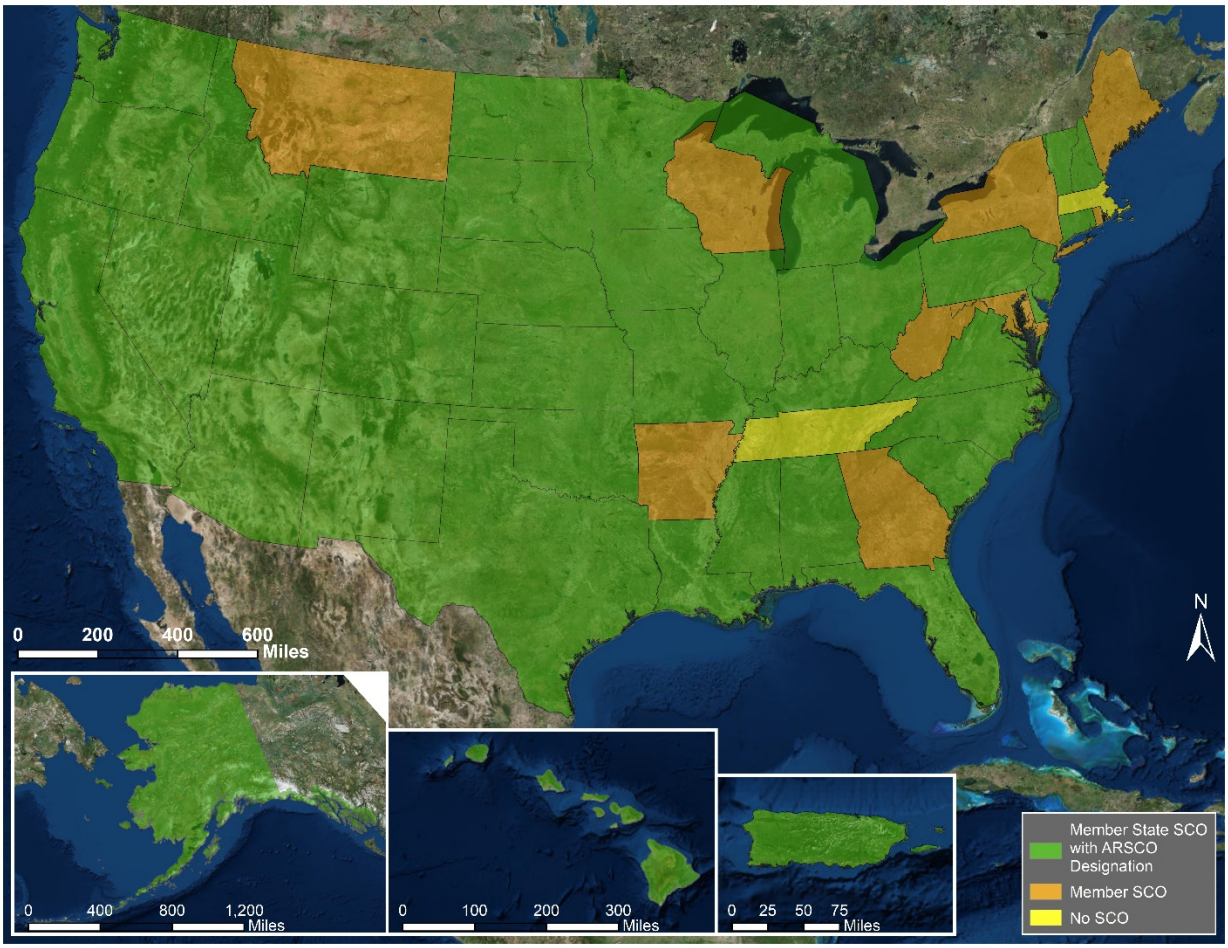


Figure 3. SCO status across the United States

Two states do not have a state-recognized SCO: Tennessee and Massachusetts. Amongst the 49 SCOs, there are excellent examples of state and local weather and climate data products and services that drastically improve the state's access to weather and climate data. This enables these states to use refined, local data for planning, emergency response, economic assessment, agricultural impacts, and more. Some states have established extensive mesonet systems – a collection of well-distributed, very high accuracy automated weather stations. This allows states to provide several valuable data services. Mesonet systems are able to capture "meso" scale climate differences in localized areas that are not recorded from coarser climate data. This leads

to improved weather forecasting for these smaller areas and improved climate data for long term analysis (Mesonet Essentials… 2019).

A good example of a product produced by an SCO is the Severe Storm Reports tool developed by the North Carolina SCO. This tool shows the climate history of severe weather in North Carolina using the validated severe storm reports originally from the National Weather Service offices (see Figure 4) (Severe Storm Reports Tool… 2019).

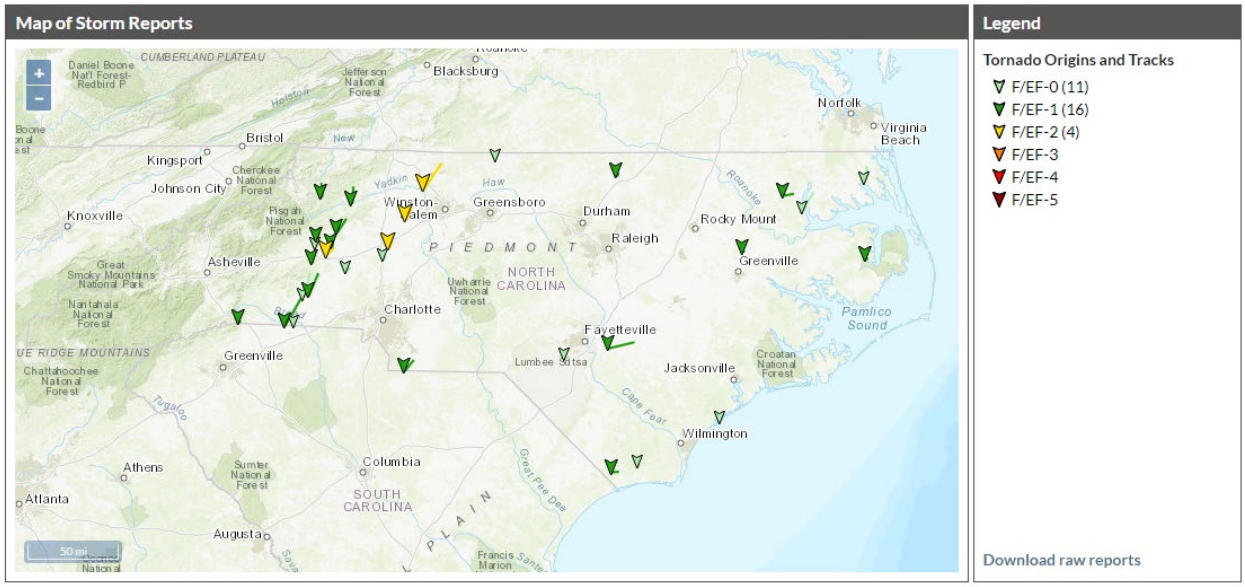


Figure 4. Screenshot of the Severe Storm Reports Tool taken on 03/17/2019

The Kentucky mesonet agriculture application is an example of a mesonet capturing meso-scale weather and climate differences in localized areas. This application shows current, past, and forecast weather in the area surrounding the weather stations (Kentucky Mesonet Agriculture Tool… 2019).

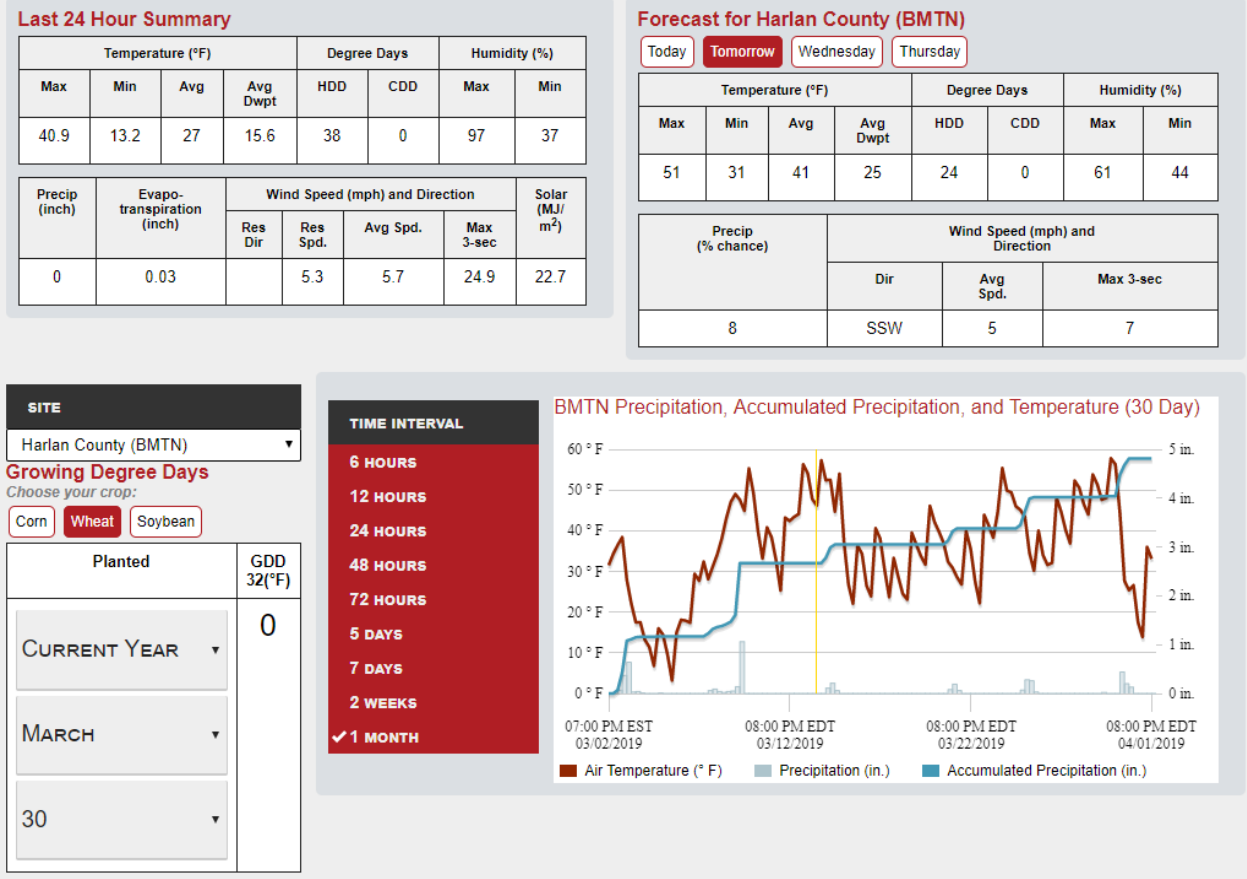


Figure 5. Screenshot of Kentucky mesonet agriculture application taken on 04/01/2019

Another example is the Oklahoma SCO Mesonet Agriculture Essentials tools. This is a suite of tools showing climate conditions such as drought as well as weather forecasting such as the Cattle Comfort Advisor (see Figure 6) (Agriculture Essentials… 2019).

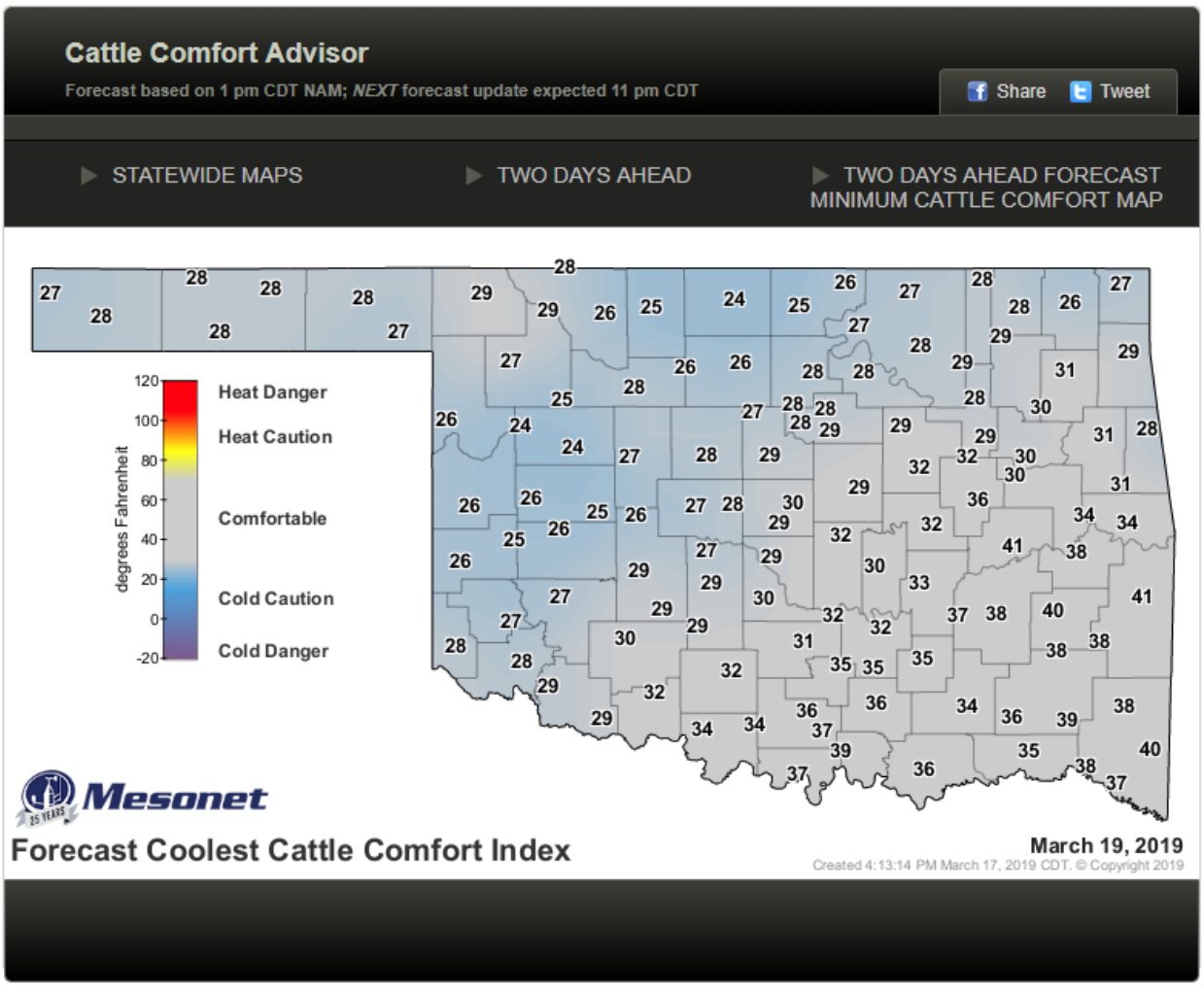


Figure 6. Screenshot from the Oklahoma SCO agriculture essentials tool

A final example is the Delaware Weather Hazard Index (DWHI), developed by the Center for Environmental Monitoring & Analysis (CEMA) which houses the Office of the Delaware State Climatologist. This web application was the inspiration for the second product of this thesis, which is covered in more detail in section 3.2 (Callahan 2014).

As mentioned previously, Tennessee is one of two states lacking a state-recognized SCO. However, efforts are underway at East Tennessee State University to develop an official SCO. Currently there is an 'acting' SCO and the two primary products that this thesis will describe are

project initiatives of the unofficial Tennessee Climate Office (TCO). Tennessee's climate varies greatly from west to east and has wide-ranging impacts on many parts of the state's economy. The SCO for Tennessee was originally established under the direction of the Tennessee Valley Authority (TVA), however it ceased operations in 2006. Over ten years later in 2017, researchers at East Tennessee State University began to re-establish the TCO. The mission of the TCO is to provide climate-related services (e.g., weekly drought monitor advisement, monthly climate summaries, historical climate data for hazard mitigation planning, etc.) to state, local and federal agencies, businesses, and the citizens of Tennessee. The TCO partners with the four NWS WFOs serving Tennessee, multiple state agencies, and regional climate data representatives at the University of Tennessee Institute of Agriculture, the University of Tennessee-Martin, and Vanderbilt University. For additional information about the TCO's mission, see Appendix A: Mission Statement.

Without a long SCO history, climate services and products are severely lacking across Tennessee. Only recently (October 2017), statewide monthly climate reports started production and in November 2017, members of the TCO began to participate in the weekly U.S. Drought Monitor discussion, providing input on the expansion and reduction of drought in areas across the state (https://droughtmonitor.unl.edu/CurrentMap/StateDroughtMonitor.aspx?TN). This was identified as a critical gap for Tennessee by meteorologists at the NWS-Morristown WFO (personal communication). The monthly reports constitute the first core product of the TCO. Within the report are several analysis summaries of climate data for the month. The first summary describes the monthly temperature summary compared to normal temperature for the month based on historical climate data. There are similar summaries for monthly precipitation, soil moisture, and stream flow. A summary of the drought monitor is also included as well as the

top ten warmest, coldest, wettest, and driest monitoring stations in Tennessee, and a miscellaneous section that covers topics such as crop and potential fire conditions. Lastly, the storm reports for the month are combined in a map that incorporates the locations of reported extreme storm conditions for each day of the month. An example Tennessee monthly climate report can be found in Appendix B. The monthly climate report is sent to the Southern Regional Climate Center (SRCC) no later than the 5th of the following month, where it is then synthesized and sent to NCEI to become a part of the national monthly climate report (https://www.ncdc.noaa.gov/climate-information/analyses/monthly-us-climate-reports). The goal of monthly climate reports (at the state, regional, and national levels) is to place climate conditions and significant events into historical perspective.

For the TCO, short- and long-term goals center around two questions: 1) How can the monthly climate report be improved upon and produced more efficiently? and 2) What value-added, data-driven products should the TCO develop based on local and statewide needs? To help accomplish the TCO goals based on those two questions, a Python script was written to automate the production of daily storm report maps that are generated by the TCO. A Weather Forecast Hazard Index (WFHI) web application product was also created using the Python scripting language for the TCO.

Automation and processing of 'big data' are key elements in addressing TCO goals. One important way of harnessing the power of computer technology to improve existing products and develop new products for the TCO is through the use of programming languages. The Python scripting language was developed in the early 1990's as a successor of the ABC language (Rossum & Drake 2003). It is an interpreted, object-oriented, high-level programming language. Additional details will be provided in Chapter 2. Today it is widely used by professional and

amateur programmers and in the development of standalone programs and scripting applications for a host of different domains (e.g., Boeing 2017, Brown 2014, Brown et al. 2017, Roberts et al. 2010, Granger et al. 2011). Using Python to automate processes is one of its key benefits. It greatly increases productivity, reduces the chance of error, and provides the ability to execute large, complex processes in a greatly reduced amount of time. Use of the Python programming language is applied in this thesis to provide partial or complete automation of several potential TCO services.

Motivation

The process of producing daily storm report maps was chosen as an automation project for this thesis since it has the greatest impact on the production time of the monthly climate reports outside of writing. This thesis will explain in detail which processes were automated and how they were automated using Python scripting. Issues encountered during the building of the automation script will also be discussed along with plans for additional automation scripting for the monthly climate reports in the future.

While building a relevant data-driven product for statewide needs, several ideas were proposed. These ideas are based on existing SCO products in other states or products in development. The WFHI web application is based on the Delaware Weather Hazard Index (DWHI),

Development of a similar ‘weather hazard index’ product was chosen for various reasons. The immediate impact of a hazard index for forecasted weather for Tennessee provides very high value by improving decision-making in local communities, specifically for emergency managers. It provides long-term value in being a scalable service that can be built for counties and many of the processes within the primary script can be reused for other weather-related products.

Additionally, the technical nature of scripting a web application that automatically updates every 6-12 hours is appealing and provides an avenue for developing advanced Python scripting abilities.

This thesis will explain in detail what the hazard index web application is, how it functions, and how it was built and automated using Python scripting. Issues that were encountered during the building of the hazard index web application will also be outlined along with future plans for additions to the web application.

Organization and Contribution by Chapter

This thesis has four chapters in total. These chapters together should clearly outline how the objectives for this thesis were fulfilled and what additional work could be accomplished.

Chapter 1: Introduction

This chapter introduces the reader to the importance of climate and weather data. It explains the general hierarchy of climate data services from the federal level to the state/local level. It then describes efforts to develop a Tennessee SCO, the current goals in this effort, and how the objectives of this thesis can help achieve those goals.

Chapter 2: Python

This chapter introduces the reader to the Python scripting language so they will understand later chapters. The chapter discusses the most basic of Python scripting concepts such as variables, classes, and modules. It then progresses to more complicated ideas such as lists, dictionaries, tuples, looping, and decision structures. The chapter ends with a brief discussion of writing batch files for automation.

Chapter 3: Methodology and Products

This chapter describes the process of building the monthly storm reports and how part of that process is automated. It then details the data used in the automation process, and the automation process methodology. It goes on to describe and explain the hazard index web application. The methodology for building and automating the hazard index web application with Python is then explained.

Chapter 4: Results, Discussion and Conclusion

This chapter presents the challenges faced while building the automated daily storm report maps and hazard index and implementing full automation using batch files and task scheduler. Limitations of the scripts and products are then discussed, followed by an outline of future research priorities and goals.

CHAPTER 2

PYTHON

Introduction

To explain the methodology of automation in this thesis it is important to have a basic understanding of the language used to automate processes. That language is Python. This is a brief overview of the language to help in understanding the next two chapters, which will describe two products developed via Python scripts and tools. There are several versions of Python. Python version 3 is primarily referenced. When Python version 2 is used it will be stated. Python is a high level language as opposed to low-level languages such as machine languages that have instructions written in binary (Python… 2019). Python is also a scripting language. This means that it does not compile before running the code that has been written. Compiling means that what has been written in the language is turned into machine language for use on the computer before executing what was written. Python interprets what has been written. Interpreting means that the program goes line by line through what has been written, translating it into machine language and executing it (Python… 2019). The programs that are written in Python are referred to as scripts. Python is an object oriented language. When we define a variable as an object type, such as the number 3 being an integer, it has attributes with which it is associated. Attributes for an object may include a unique ID or location in the computer's memory, properties describing the object, and methods or tasks that the object can complete (Quinn & Dutton 2019).

Variables, Types, and Classes

Variables mentioned previously represent objects and are stored for use within a given script. There are many different object types. Two object types that will be used frequently in

this thesis are strings and integers. Strings are an array of characters such as a word (Germain 2019). Integers are numerical variables such as the number 3. Below are some examples of string and integer object variables written in Python.

```
grandmaJane = 3
grandpaGeorge = 'I really need to work on my spelling.'
russia = 'Shoop_interstate.shp'
nephew = r'E:\Grad_School\Archive_2\IDSCloudServerGIS'
doggo = nephew + '\\' + russia
```

A special group of object types are known as a class object. The attributes that these objects have are defined by its class. It is a map of the properties that describe it, and the methods or things the object can execute. One way to introduce additional object classes to scripts, is to import them through modules using the import statement (Quinn & Dutton 2019).

```
import os
import time
```

Modules

There are a number of modules that will be imported for these projects. Table 1 will briefly describe what each module is and an example of what it does.

Table 1. Most modules used in product development are listed and described in this table

Import Name	Full Name	Description	Function Example
os	Miscellaneous operating system interfaces	Allows the use of operating system dependant functionality	creating a file folder
sys	System specific parameters and functions	Allows access to variables of interpreter and functions that interact with it.	list path that script originated from
time	Time access and conversions	Allows access to time related functions	returns a string of local time
datetime	Basic date and time types	provides classes for manipulating dates and times	return string of date minus 1 day
dateutil	dateutils	provides additional functionality for datetime module	return string of date minus 1 month
shutil	High-level file operations	provides funtions for copying and removal	copy the contents of file to target
arcpy	arcpy	provides functions for geographic data analysis, data conversion, data management, and map automation	convert raster to ascii file
arcgis	arcgis	provides an information model for GIS hosted within ArcGIS Online or ArcGIS Enterprise	update data on ArcGIS Online
csv	CSV File Reading and Writing	implements classes to read and write tabular data in CSV format	read a csv file
gdal	Geospatial Data Abstraction Library	two libraries for manipulating raster and vector data.	counting the number of rasters
urllib.request	extensible library for opening URLs	provides functions and classes which help in opening URLs (Python version 3)	Open URL network object for reading

Modules are a script or collection of scripts that allow access to not only additional classes but also a collection of tools to manipulate objects. These tools are called functions. There are many functions used in this paper from modules and they provide a host of services. An example of a function is the os module function, used to create a directory (file folder). The code for this function is shown below. This is the os module function with variable ‘newfolder’ designating the location to create the new folder and the name of it:

```
os.mkdir(newfolder)
```

Another example of this would be the extract subdataset function from the arcpy module used to extract data from multidimensional array files in this thesis. The extract subdataset function has two variables 'ndfd_data' and 'suboutpath' included in the example representing paths for input and output data and the variable 'numa' represents a number.

```
arcpy.ExtractSubDataset_management(ndfd_data, suboutpath, numa)
```

Lists, Dictionaries, and Tuples

Lists, dictionaries, and tuples are data types used for storing multiple strings and integers. Each has advantages and disadvantages depending on what script is being written.

Lists are used throughout this project and are designated by using brackets. See below for an example:

```
shapefilelist = [myinterstates, mycities, myTNBorder]
```

Each of the variables shown within the brackets is indexed, meaning there is a number associated with them. Indexing starts at 0. This would mean in the example above that 'myinterstates' has an index of 0, while 'myTNBorder' has an index of 2.

Tuples are, in a sense, lists that cannot be changed. This is known as being immutable. Tuples are immutable and lists are mutable meaning they can be changed. Tuples are designated by using parentheses and are indexed just like lists. An example of a tuple is shown below:

```
shapefilelist = (myinterstates, mycities, myTNBorder)
```

Dictionaries are another way of storing multiple pieces of data. Unlike lists, dictionaries have no index. Instead they have a key value they use to reference items. This is known as a key value pair. A key word is used to pull values from the dictionary much like an index can be used

to pull data in a list. Dictionaries are designated by using curly braces. An example of this can be seen below (Quinn & Dutton 2019):

```
shapefilelist = {myinterstates: 'US-81' , mycities: 'Tampa', myTNBorder: 'Eastside'}
```

The major advantage of using a dictionary is the key value pair. Data can be associated to other data through the use of a dictionary. In the script above, 'myinterstates' is paired with string 'US-81.'

These three data types can be used within each other as well as themselves. The first examples of this below would be lists within a list. The second example is a dictionary within a dictionary. The third example below shows lists within a tuple.

```
shapefilelist = [[myinterstates, mycities, myTNBorder],
                 [herinterstates, hercities, herTNBorder]]
```

```
weatherlist = {'ha_0_ticlipppolyclip': {hail: 'high'}, 'ic__TNinchpolyclip':
               {lowiceaccum: 'low', midiceaccum: 'medium', highiceaccum: 'high'},
               'sn__TNinchpolyclip': {lowsnowfall: 'low', midsnowfall: 'medium',
                highsnowfall: 'high'},'temp_max_TNfehrpolyclip': {lowhitemp: 'low',
                midhitemp: 'medium', highhitemp: 'high'}, 'temp_min_TNfehrpolyclip':
               {lowlowtemp: 'low', midlowtemp: 'medium', highlowtemp: 'high'},
               'to_0_ticlipppolyclip': {tornado: 'high'}, 'wg__TNmphpolyclip':
               {lowwgust: 'low', midwgust: 'medium', highwgust: 'high'}}
```

```
shapefilelist = ([myinterstates, mycities, myTNBorder],
                 [herinterstates, hercities, herTNBorder])
```

Looping

A loop is a piece of script that repeats a set of actions (Quinn & Dutton 2019). There are two types of loops. There is the 'for loop' statement and the 'while loop' statement. 'For loops' are often used to perform the same functions and statements over and over on each individual variable in a list, dictionary, or tuple. A 'for loop' statement appears in the example below.

```
for shape in shapefilelist:
    arcpy.analysis.CreateThiessenPolygons()
```

In this ‘for loop’ statement, each individual variable that is represented by the variable shape in the list, tuple, or dictionary, which is presented by the variable shapefilelist, will have the ‘create thiessen polygons’ function from the module arcpy performed on it.

‘While loops’ are slightly different than ‘for loops’ and execute whatever script has been built for them until a condition is met (Quinn & Dutton 2019). ‘While loops’ usually involve the use of a counter to determine when a condition has been met. A ‘while loop’ statement appears in the example below.

```
numa = 0
while numa < 5:
    arcpy.analysis.CreateThiessenPolygons()
    numa += 1
```

In the ‘while loop’ statement we see the variable ‘numa’ is set to 0. Every time that ‘create thiessen polygons’ is executed the value of the numa variable is increased by 1. Once the numa variable reaches 5 the ‘while loop’ stops.

Loops can be nested. This means that you can have a loop inside of a loop. An example of this is the example below.

```
for shape in shapefilelist:
    for ape in shape:
        arcpy.analysis.CreateThiessenPolygons()
```

In the nested loops shown in the example above, each individual variable that is represented by the variable ‘shape’ in the list, tuple, or dictionary, which is presented by the variable shapefilelist, is iterated through. In this case there are variables within shape. As the first ‘for loop’ goes through each individual variable it will perform the second ‘for loop’ which will iterate through the variables within the first ‘for loops’ variables.

Decision Structures

Decision structures are used to provide conditional logic to scripts. 'If and else' statements are used often to set up a Boolean decision structure. The example below shows what a typical 'if and else' statement looks like in Python.

```
for shape in shapefilelist:
    if shape == myinterstates:
        arcpy.analysis.CreateThiessenPolygons()
    else:
        arcpy.management.CalculateField()
```

In the script above, the 'for loop' goes through the variable shapefilelist testing each variable shape to see if it is named myinterstates. If the variable is named myinterstates then the function 'create thiessen polygons' is executed. If the variable is not named myinterstates then the function CalculateField is executed.

Not all decision structures have to be Boolean. Instead of using an 'else' with an 'if' statement, an 'elif' statement can be used. An 'elif' statement allows for multiple decisions to be made. An example of this can be found below.

```
for shape in shapefilelist:
    if shape == myinterstates:
            forcast = 'wg_'
    elif shape == mycities:
            forcast = 'tp_'
    elif shape == myTNBorder:
            forcast = 'sn_'
```

In the script above, the 'for loop' goes through the variable shapefilelist testing each variable shape to see if it is named myinterstates. If the variable is named myinterstates then the variable forecast is 'wg_'. Else if the variable is named mycities then the variable forecast is 'tp_'. Else if the variable is named myTNBorder then the variable forecast is 'sn_'.

'Try and except' are similar to the decision structures already discussed but they serve as an alternative to script failing. A 'try and except' used often in this thesis is shown below.

```
try:
    os.mkdir(bombsite)
    print('folder created')
except:
    print('file already made')
```

In the script above, the ‘try’ statement is used for the function to make a new file folder. If for some reason the file folder cannot be made in that location, instead of cancelling the entire script the script moves on to the ‘except.’ In this case it executes the function ‘print.’

Batch Files and Task Scheduler

To fully automate Python scripts, Task Scheduler is used. Task Scheduler is a tool that allows actions to be executed when a trigger condition is met. Trigger conditions can be many different things. For this thesis only a trigger linked to a schedule is used (see Figure 7).

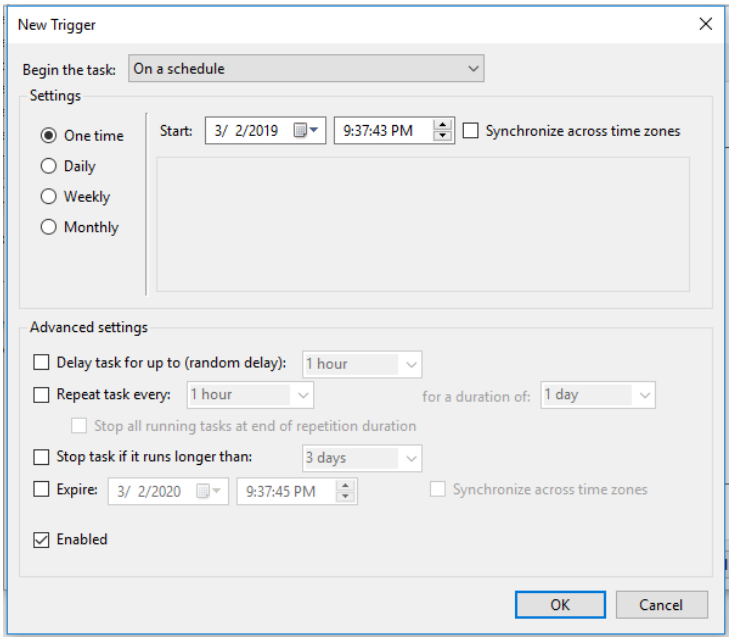


Figure 7. New trigger dialog within Task Scheduler

The trigger is used to execute an action. There are several different types of actions but this thesis uses the type ‘start a program.’ With this type of action a script can be set to execute. For this thesis the action settings are set up as follows. The Program/script, designates the

version of Python to be used. An argument is added that is a path to a batch file (see figure 8). Once this basic set up is complete with minor additional changes, the batch file should run at the scheduled time set up as the trigger to execute the action.

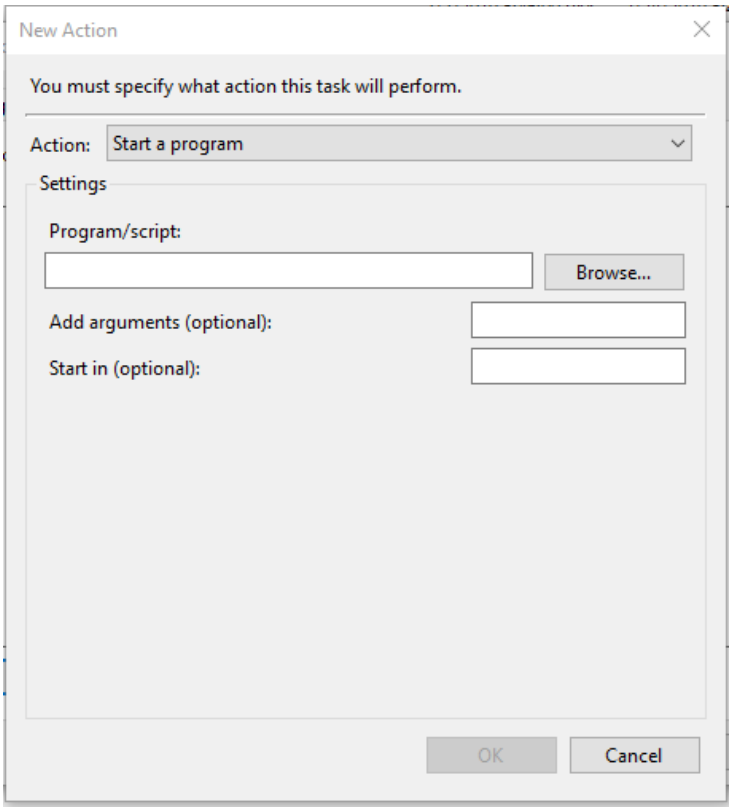


Figure 8. New Action dialog within Task Scheduler

Batch file is the term used for a disk operating systems (DOS) script file that is written to execute a list of commands in DOS. By listing the path to individual scripts in the batch file in executable order, all the scripts for a process can be tied to one Task Scheduler action. This thesis uses a sophisticated approach by listing the preferred Python executable in front of the path to the Python script (see Figure 9).

```
"E:\Grad_School\Archive_2\Thesis\State_Climate_Office\Final scripts\NDFD_Conus_only.py"

cd "\Program Files\ArcGIS\Pro\bin\Python\envs\arcgispro-py3"

"C:\Program Files\ArcGIS\Pro\bin\Python\envs\arcgispro-py3\python.exe" "E:\Grad_School\Archive_2\Thesis\State_Climate_Office\Final scripts\GRIB_to_VectorTile4test.py"

"C:\Program Files\ArcGIS\Pro\bin\Python\envs\arcgispro-py3\python.exe" "E:\Grad_School\Archive_2\Thesis\State_Climate_Office\Final scripts\HazardIndex_VectorTile2update.py"

"C:\Program Files\ArcGIS\Pro\bin\Python\envs\arcgispro-py3\python.exe" "E:\Grad_School\Archive_2\Thesis\State_Climate_Office\Final scripts\Autopost_jupytertyped.py"

PAUSE
```

Figure 9. Sample Batch file used in argument in Task Scheduler

CHAPTER 3:

METHODOLOGY

SPC Storm Reports Description

Creation of the Tennessee monthly climate report requires completion of three broad steps each month. Monthly data must be analyzed, a summary of findings made, and maps generated. Automation of daily storm reports reduces, and in some cases eliminates, the manual process of generating maps for daily storm events for inclusion in the monthly report, resulting in a streamlined and efficient process.

To produce daily storm report maps for the monthly climate reports, the following process is executed weekly. Daily storm report data are downloaded from the NOAA Storm Prediction Center (SPC) website. Data are used to generate point shapefiles representing recent extreme storm events. If any storm report locations are within the state of Tennessee, a daily storm report is created for those days. Data within the state of Tennessee are extracted and placed within a map exchange document (.mxd). Within the .mxd, the number of weather occurrences are changed in the legend to match the daily data. On the top center of each map the date is changed to reflect the day of the occurrences. A portable network graphic (PNG) are then created from the .mxd.

The production efficiency from generating one daily storm report map versus using the automated process is small. However, used every day over months and years the value of this efficiency becomes much more significant.

SPC Storm Reports

The NOAA Storm Prediction Center (SPC), which is a part of the National Weather Service (NWS) and the National Centers for Environmental Prediction (NCEP) (SPC…2019),

produces storm reports in real time (updated every 10 minutes) and maintains archives of major events as far back as the 1950s. The SPC provides forecasts for extreme weather such as severe thunderstorms, tornadoes, and hazardous winter weather. It also provides daily (summary) storm reports of weather events. These are collected from local storm reports (LSRs) that are produced by local NWS offices. If there is no LSR for weather or if it takes more than 10 days for an LSR to be created, then that weather event will not appear on the SPC storm report (Storm Prediction Center… 2019).

Daily summary storm reports cover the previous day and are split into three separate reports: tornado reports, hail reports, and wind reports. They can be downloaded in a .csv format for viewing and analysis. There are several fields contained within the storm reports file. Tornado storm reports include Time, EF Scale, Location, County, State, Latitude, Longitude, and Comments. Time is shown in Universal Coordinated Time (UTC), also called Greenwich Mean Time (GMT) (Storm Prediction Center… 2019). EF Scale is the rank that the tornado was assigned on the Enhanced Fujita scale. This scale measures the speed of a tornado based on damage caused to structures and ranges from EF0 to EF5 (Fujita… 2019). Location is the general address of the occurrence. County and State are self-explanatory. Latitude and Longitude describe the GPS coordinates of the event. The datum they represent is North American Datum 1983 (NAD83). Comments is for general comments and may contain a three letter ID that represents the local NWS office that created the LSR. The hail report has similar fields except the EF Scale field is replaced by a Size field. Size records hail size in 1/100 of an inch increments. The wind report is similar to the other two reports with the exception that Speed is included instead of EF Scale or Size. Speed refers to the speed of wind gusts and is measured in miles per hour (mph) (Storm Prediction Center… 2019).

Automation of SPC Storm Reports

One script (Figure 10) automates 'pulling' the SPC storm report data, building storm report feature classes, and exporting a map to a .png format. A 'for loop' goes through a list of urls for accessing the storm report data. At the beginning of the 'for loop,' the module urllib.request requests and opens the storm report data urls. Immediately afterwards, the csv module is used to read the .csv at the SPC storm report data url. The .csv file is then written to the local computers working directory. The module os is used to designate this working directory which is also an archive directory.

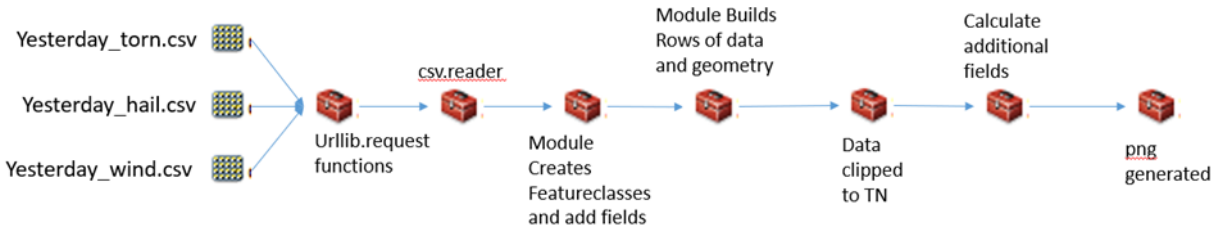


Figure 10. Model of Automation of SPC Storm Reports

An 'if and elif statement' differentiates between the tornado, hail, and wind storm reports. The headers for each field within the .csv file are then identified and two modules built within the script are called with the field's headers as variables.

The first module creates a feature class and then adds fields named according to the headers that it receives in the 'if, elif' statements. An additional field 'Date' is added to the shapefile. The second module then uses the arcpy module 'Insert Cursor' function to add data to feature class fields from the .csv file.

The feature class is then clipped to a feature class of the Tennessee state boundary. The Date field for the original feature class file, the clipped feature class, and the state boundary

feature class are then calculated. The calculation fills in a time as the full month name, the day and the year with century. This is done so the field can be inserted into the title of the storm report map using dynamic text within an ArcGIS Pro project. The original feature class and the clipped feature class are copied to a working folder for the storm report map. The time stamp is removed allowing these feature classes to replace the feature classes with the same names from the previous day. These feature classes are referenced in the storm report map ArcGIS Pro project used to publish maps of daily storm reports. The Arcpy module function 'ArcGISProject' references the ArcGIS Pro project. The arcpy module 'mapping export to png' exports the map created in the ArcGIS Pro project to a png.

Designing the Weather Forecast Hazard Index Webapp

The Weather Forecast Hazard Index (WFHI) web application is based on the Delaware Weather Hazard Index (DWHI), developed by the Center for Environmental Monitoring & Analysis (CEMA), which houses the Office of the Delaware State Climatologist. The goal of the DWHI is to complement and enhance the warnings from the local NWS WFO. The DWHI provides a multi-faceted approach to potential hazards and incorporates local geographic influences when predicting potential hazards (Callahan 2014).

Potential hazards include extreme temperatures (heat index, wind chill), maximum wind gust, precipitation, snow, ice, convection outlook, and coastal water level. Forecast data are downloaded from the National Digital Forecast Database (NDFD) with the exception of the coastal water level which is acquired from the Delaware Bay Operational Forecast System (DBOFS). The forecasting length is approximately two days (48 hours) and the index is updated after the release of new forecast data every 12 hours. The hazard risk level is represented by a 5

km^2 raster cell. With color coding representing the level of hazard (see figure 11) (Callahan 2014).

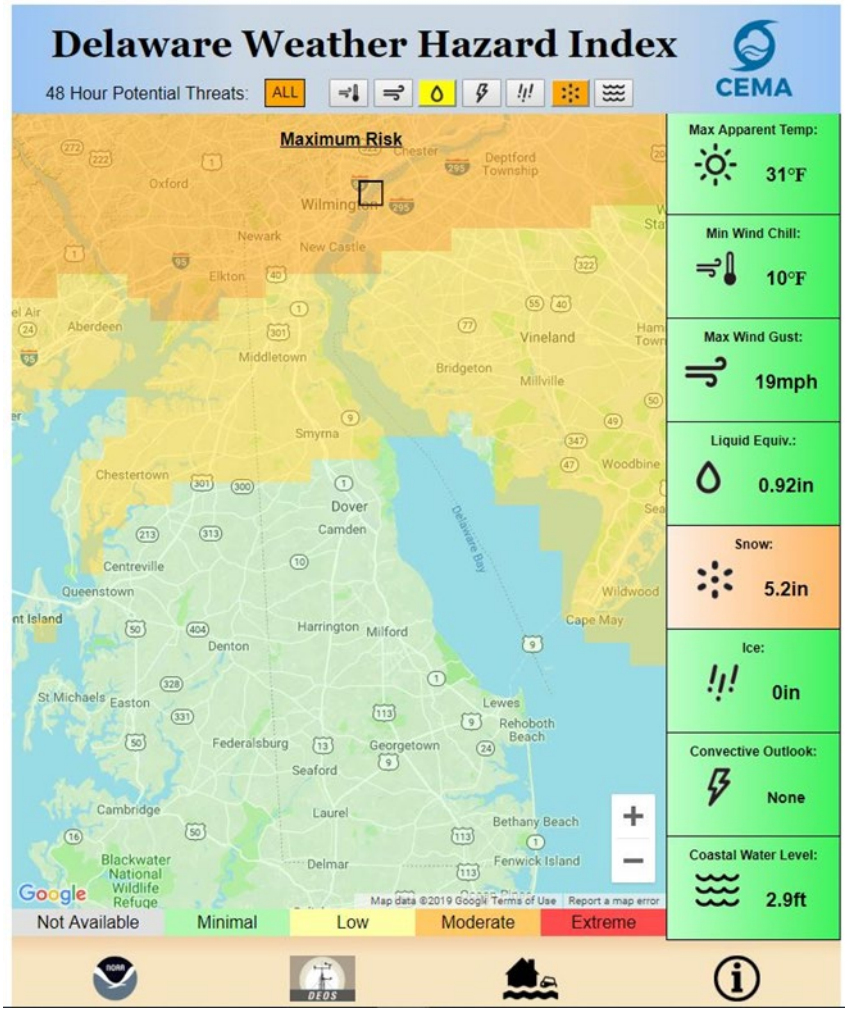


Figure 11. Image of DWHI showing low to moderate risk of snow in northern Delaware on 03/03/2019

The Weather Forecast Hazard Index (WFHI) webapp is a dashboard web application developed using ArcGIS Online (AGOL). This web application is updated every 12 hours using the latest forecast weather data. It shows the most extreme weather forecast for the next 72 hours in discreet polygons that cover the state of Tennessee. These squares are typically 5 x 5 km polygons from a scale of 1:290,000 and outward, and typically 2.5 x 2.5 km polygons from a

scale of 1:290,000 and inward. Each polygon is color-coded to show a forecast for singular or multiple weather conditions that are classified as hazardous in the polygon's area (figure 12).

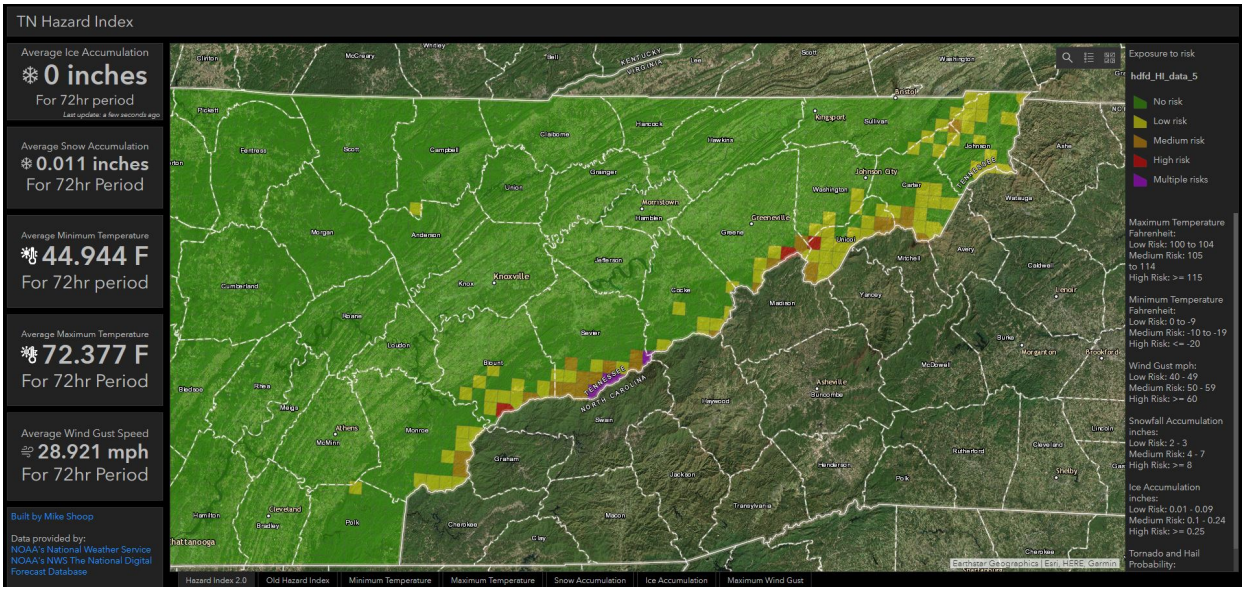


Figure 12. Screenshot of WFHI with low to high risk levels for wind gusts on April 19, 2019

The primary weather elements examined for hazardous conditions include minimum apparent temperature, maximum apparent temperature, wind gust, ice accumulation, snow accumulation, maximum quantitative precipitation over 6 hours, and categorical convection. Hazardous classifications for each weather type are shown at the bottom right of the dashboard.

Clicking on an individual polygon activates a popup showing the extreme forecast weather condition or conditions that are classified as a hazard. It also shows a list of measurements for all monitored weather extremes over the 72-hour period. Widget indicators have been added to the Dashboard to show the average in extreme weather at the scale and location of the viewers' choosing. These are shown on the top left of the Dashboard. At the top right of the map, an address locator and basemap switcher can be found.

In addition to the main WFHI map, the Dashboard web application also contains individual maps of each of the extreme weather forecast datasets being monitored, allowing for individual, granular analysis of forecast weather.

Data

The primary source of national weather forecast data is the National Digital Forecast Database (NDFD). The NDFD was created by the Meteorological Development Laboratory (MDL), formally known as the Techniques Development Laboratory (TDL). The MDL is under the Office of Science and Technology within the National Weather Service (NWS) (US Department of Commerce 2018). Data provided by the NDFD includes weather forecasts from WFOs with the addition of data from the National Digital Guidance Database (NDGD). The WFO's use the Interactive Forecast Preparation System (IFPS) software on an Advanced Weather Interactive Processing System (AWIPS) to produce geospatially referenced digital forecast data (Dion et al. 2019). The NDGD is used to validate NDFD data (MDL… 2018).

Raster data collected from the NDFD comes in a format called General Regularly-distributed Information in Binary (GRIB) form. Specifically, it comes in the GRIB2 format. GRIB is the data format chosen by the World Meteorological Organization (WMO) for exchanging and storing gridded data. The three main reasons this format is used are its smaller size relative to normal binary files, self-describing fields that result in a small database of metadata, and general format as an open files source that meets international standards (GRIB… 2019).

The GRIB2 files collected from NDFD represent a multidimensional array that shows measurements of an element such as temperature or snow over different time periods. NDFD stores GRIB2 files showing forecast data for 1-3 days, 4-7 days and 8-450 days. Within those

forecast data time periods, data are split into the multidimensional array of rasters representing a period of hours within the forecast data time periods. An example of this would be the 1-3 day temperature GRIB2 file. Within the GRIB2 file are hourly temperature forecasts for the first 36 hours. After that, the remaining 36 hours will have a temperature reading every 3 hours. For files with forecasts past 72 hours there is a temperature reading for every 6 hours up to 168 hours (Dion et al. 2019). This will differ for some files. Snow and ice accumulation have forecasts for new accumulation every 6 hours. The 1-3 day data will be utilized in this thesis for product development.

There are raster files covering the entire continental US and subsections of the US. For this thesis the data file for the entire continental US was chosen. Tennessee is located in two subsections, the Mid Atlantic and Central Mississippi Valley subsections. Clipping data to the extent required was the simplest solution to obtain data for our case study. The size of each cell in the GRIB2 file is 2.5 square kilometers. The horizontal datum used is the World Geodetic System 1984 (WGS84). The vertical datum used is WGS84 geoid. The projection type used is Lambert Conformal Conic (Spatial Reference System… 2019).

Weather elements ‘pulled’ from the NDFD were the most common weather phenomenon. Below in table 2 all elements pulled from the NDFD database are listed.

Table 2. Weather elements pulled from the NDFD for product development

Element	Native units	GRIB units	GRIB2 File Name	Description
Apparent Temperature	deg Fahrenheit	deg Kelvin	ds.apt.bin	perceived temperature either wind chill or heat index
Snow Accumulation	inches	meters	ds.snow.bin	total accumulated new snow over a 6 hour period.
Ice Accumulation	inches	kgm-2	ds.iceaccum.bin	total accumulated new ice over a 6 hour period.
Wind Gust	knots	ms-1	ds.wgust.bin	3 second wind speed forecast to occur within a 2 min interval
Wind Speed	knots	ms-1	ds.wspd.bin	3 second wind speed forecast to occur within a 2 min interval
Quantitative Precipitation	inches	kgm-2	ds.qpf.bin	total accumulated liquid precipitation over a 6 hour period.
Convective Hazard Outlook	categorical forecast	categorical forecast	ds.conhazo.bin	categorical forecast of the potential for severe thunderstorms

Downloading Data from the NDFD Website

The first script built for this process downloads GRIB2 data to a local folder. To automate a download of the GRIB2 data from NDFD to a local file folder, a web module was chosen from several options. NDFD has a file transfer protocol (FTP) site enabling data to be loaded from that site using FTPlib. For the http site, urllib.request could be used to download the data. The module httplib was considered but more coding would be involved. Ultimately due to previous exposure to urllib.request's predecessors urllib and urllib2 in Python version 2, urllib.request was chosen to download the GRIB2 data. Modules used in the script were os, time, and urllib.request. The following is a general explanation of how the script is built.

A timestamped file folder is created using the time and os module. A 'try and except' statement was written to build the timestamped file folder. This ensures that if the file folder is already created the script will not fail. Instead, the print function will execute, stating that the directory has already been created.

A 'for loop' was constructed to go through a list of variables, each representing an individual NDFD GRIB2 file name. Each file name is then merged with the general url path to

create a full path to each GRIB2 file. The full path is then used with urllib.request functions 'Request' and 'urlopen' to access the data. The data are then written into the working directory folder.

The os module is used to set the working directory for the script. This causes the data to be written directly to the timestamped file folder created at the beginning of the script. A generalized diagram of this process can be seen below in figure 13. It is important to note that the use of lists and for loops built into this script will allow easy additions of GRIB2 data requests from different parts of the NDFD database with minimal script changes.

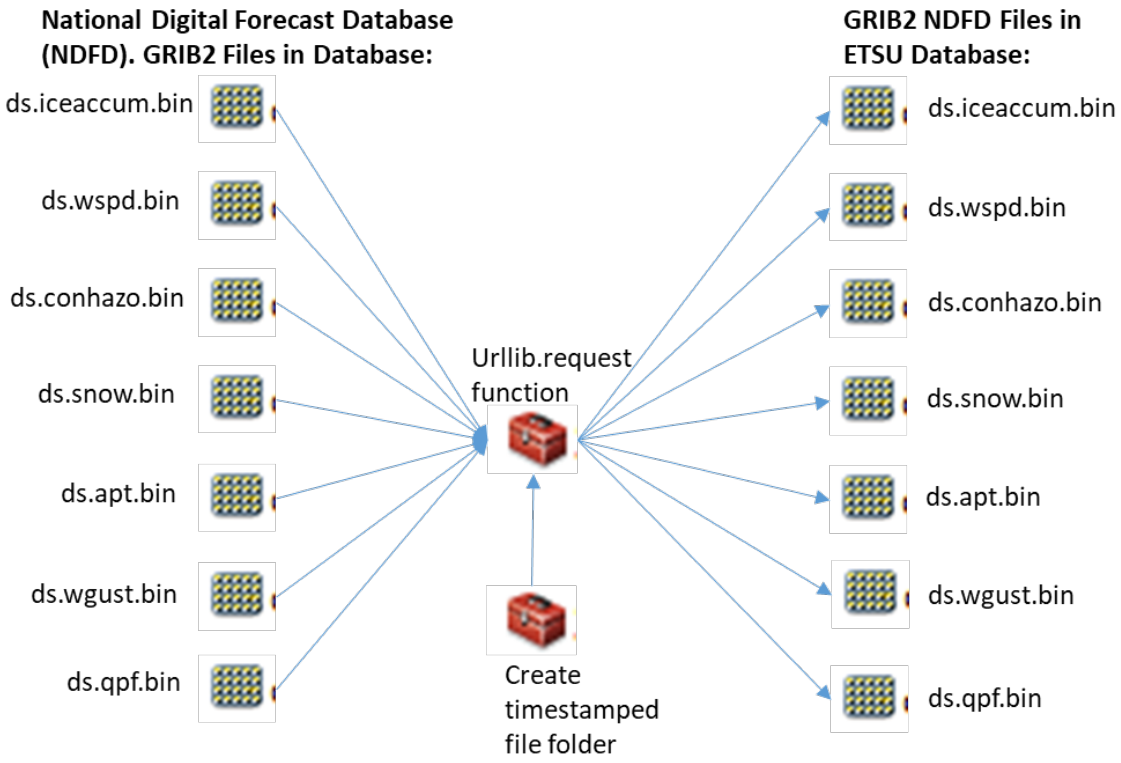


Figure 13. Model of NDFD data transfer

GRIB2 data converted to vector polygons

After the GRIB2 data are obtained, they are converted to vector tiles. There are two reasons they are not kept as a binary raster. The first is related to the complexity of setting up an enterprise server. This is pivotal in posting rasters to ArcGIS Online (AGOL). Without it you can only post a tiled map service. This format has very restricted functionality. The second relates to the difficulty in finding a way to show a grid with each grid containing a list of attributes. Below in figure 14 is a flowchart of the general process used to convert GRIB2 files to vector tiles.

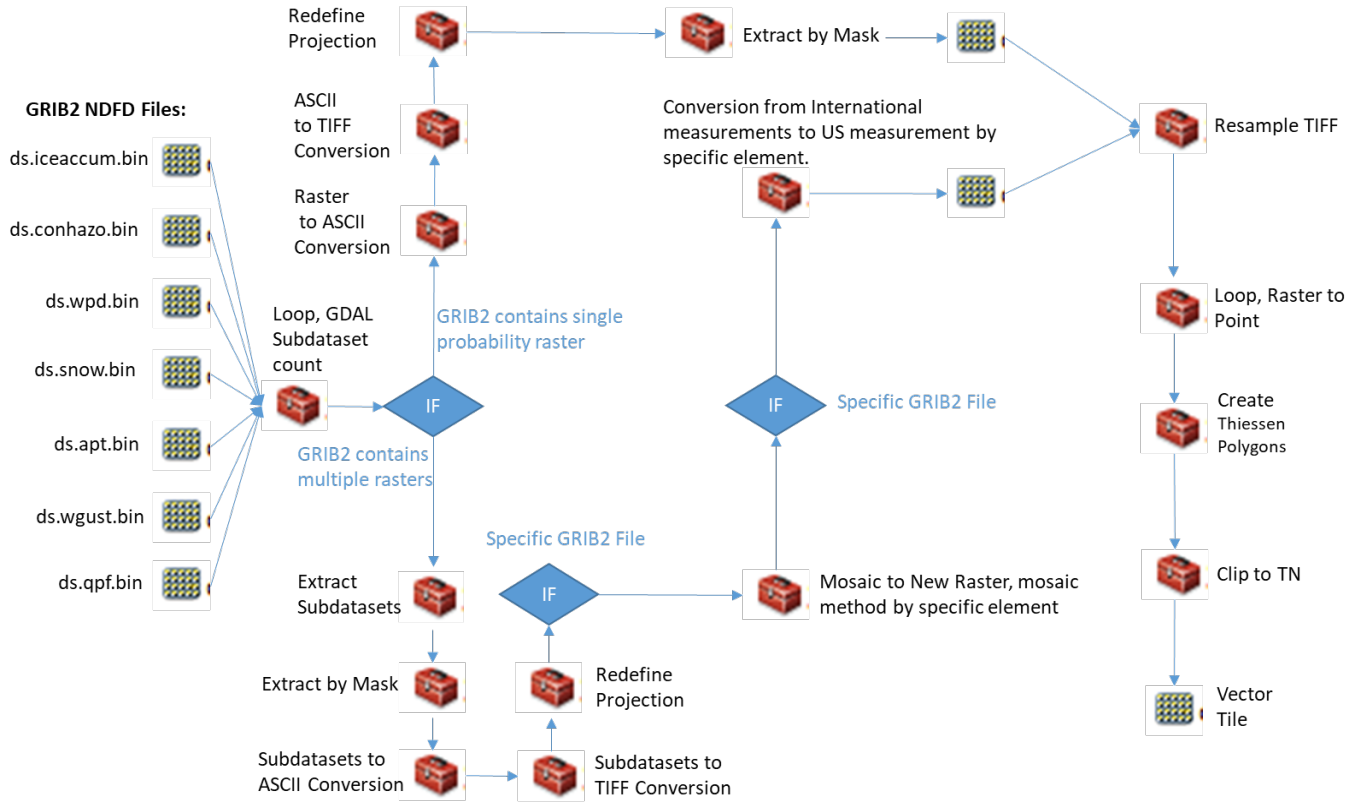


Figure 14. Model of WFHI data conversion to vector polygons

A timestamped file folder is created using the time and os module. A ‘try and except’ statement was created around the building of the timestamped file folder. This folder is designed to contain converted raster files. A ‘for loop’ was created to process GRIB2 files copied to the

timestamped file folder created in the previous script. File names were saved in a list and each file was then merged with the general url path to create a full path to each GRIB2 file. Within the 'for loop,' the module gdal used the functions open and rastercount to count how many rasters are in each GRIB2 array since the number of rasters within the GRIB2 file fluctuate. The module gdal is a translator library for raster and vector geospatial formats (osgeo… 2018). Using 'if' and 'else' statements, individual GRIB2 files are sorted between files with only one raster and files with more than one. If only one raster is contained within the GRIB2 file, conversion will begin on the file. 'Else' multiple rasters are contained within the GRIB2 file and the arcpy module function Extract Subdataset management is used to remove each raster so it can be converted individually. This is done using a raster count provided by the gdal function 'rastercount.' A 'for loop' is built into the 'else' statement which runs the loop for numbers up to the number provided by the 'rastercount' function. A variable representing the number '0' was created outside of the 'for loop' and it increases by one every time a loop is completed. This is used to tell the function 'extract subdataset management' which raster to pull from the GRIB2 file and be converted.

The area of interest for each of these rasters is extracted from the data using the arcpy 'extract by mask' tool. This provides three benefits. The first is that the reduced number of raster cells reduces the overall space needed to store our final dataset. The second is that it reduces processing time after this has been completed. The third is that using 'extract by mask', all rasters from the GRIB 2 files are resampled to have the identical cell coordinates and alignment as the other rasters from NDFD.

Raster conversion consists of using the arcpy module function 'raster to ascii conversion' to convert the binary GRIB2 rasters into an American Standard Code for Information

Interchange (ASCII) (Pattis 2019) file. The arcpy module function 'ascii to raster conversion' is then used to convert to a Tagged Image File Format (TIFF).

During the use of the function 'raster to ascii conversion,' the original projection of Lambert Conformal Conic and horizontal datum of WGS84 for the raster is lost. After the conversion to a TIFF file the arcpy function 'redefine projection' is used to define the projection and datum for the raster. This definition is recorded from the original GRIB2 raster file before the conversion to ascii and saved for use in the 'redefine projection' function.

After all rasters from one of the GRIB2 multidimensional arrays are converted to a TIFF and the projection is defined, the rasters for that GRIB2 file can be merged together into one single raster file. To do this, a list of rasters made when the ASCII file is converted to a TIFF file is used in the arcpy module function 'mosaic to new raster management'. The if, elif, and else statements are used here to sort by name so specific mosaic parameter methods depending on the weather element are used to produce specific mosaic rasters of weather elements. If the weather element is temperature, two separate mosaics are performed. This is the maximum and minimum temperature for the cells in the merged rasters. With this, the forecast maximum and minimum apparent temperature for a raster cell over the next 72 hours can be shown. The wind gust, quantitative precipitation over 6 hours, and categorical convection element rasters are mosaicked together to show each of their maximum forecasts for the next 72 hours. For snow accumulation and ice accumulation the sum is given when mosaicked. This provides a forecasted estimate of accumulation of each weather element over a 72 hour period. This completes the original for loop with all GRIB2 weather files pulled from NDFD being processed to TIFF files.

A list of all the mosaicked TIFF files is created right after the 'MosaicToNewRaster' function. A new 'for loop' is then created for the mosaicked TIFF files and the TIFF files are

then converted to US measurement standards. One if, several elif, and one else statement are used to sort TIFF files with the proper conversion. Up to this point, temperature is shown in Kelvin (K). Kelvin is the international standard for scientific temperature measurement (Kelvin… 2018). The formula applied to the mosaicked raster is as follows.

*Fahrenheit unit = (Kelvin unit -273.15) * (9/5) + 32*

Wind gust is represented as meters per second (m s^{-1}). This is an international system of units (Meters per second… 2018). The formula below converts the meters per second measurements to miles per hour within the mosaicked raster.

*Miles per hour = m/s unit * 2.236936271*

Ice accumulation and quantitative precipitation is shown in kilogram per square meter (kg/m^2). This is the pressure exerted by one kilogram of force applied to an area of one square meter. This was converted to inches of water table which is the pressure exerted by a 1 inch high column of water at a given temperature (Kilogram force… 2018). The formula for this conversion is shown below.

*In H_2O = kg/m^2 *0.039370*

Snow accumulation is shown in meters (m). This is converted to inches. Below is the formula used for this conversion (Meters to Inches… 2018).

*Inches unit = metric unit * 39.370*

This ends the 'for loop' converting the mosaicked TIFF files to US measurement standards. Any single raster TIFF files from NDFD data are extracted to an area of interest. This is beneficial for the same three reasons stated during the extraction to an area of interest for the multiple raster GRIB2 files. Testing showed that single raster GRIB2 files such as the tornado and hail percent probability rasters have slightly different raster cell locations from the weather

element data currently used in this thesis. During the extract by mask process, TIFF rasters from the single raster GRIB2 files are resampled to have the identical cell coordinates and alignment as the other rasters from NDFD. A 'for loop' processes both TIFF files which have a list built during their conversion from ASCII to TIFF files.

A new 'for loop' is now written to convert from raster data to vector data using a new list created after the conversion to US measurement standards for the mosaicked TIFF files and after the 'extract by mask' function for the single raster TIFF files. To begin with, the center of each cell in a raster is converted to a vector point. To do this, the arcpy module function 'raster to point' is used. The arcpy module function 'create Thiessen polygons' is then used to generate vector polygons from the vector point files created. The Thiessen polygons will always be closer to its origin point than to any other point (Thiessen… 2019). Due to points being generated from the center of raster cells, vector polygons were created that match the original cell size. Points that are on the edge of the study area represent an exception. Here they are much larger polygons. To address this issue, the Thiessen polygons are clipped to the study area boundary using the arcpy module function 'clip.' This will cause many of the Thiessen polygons on the boundary to be smaller than the standard size of 6.25 km^2 but insure that few polygons will be exceed that.

Creating a Hazard Index

With the data in a vector polygon format, a hazard index can be created. This process of creating a hazard index can be seen in the flowchart in figure 15.

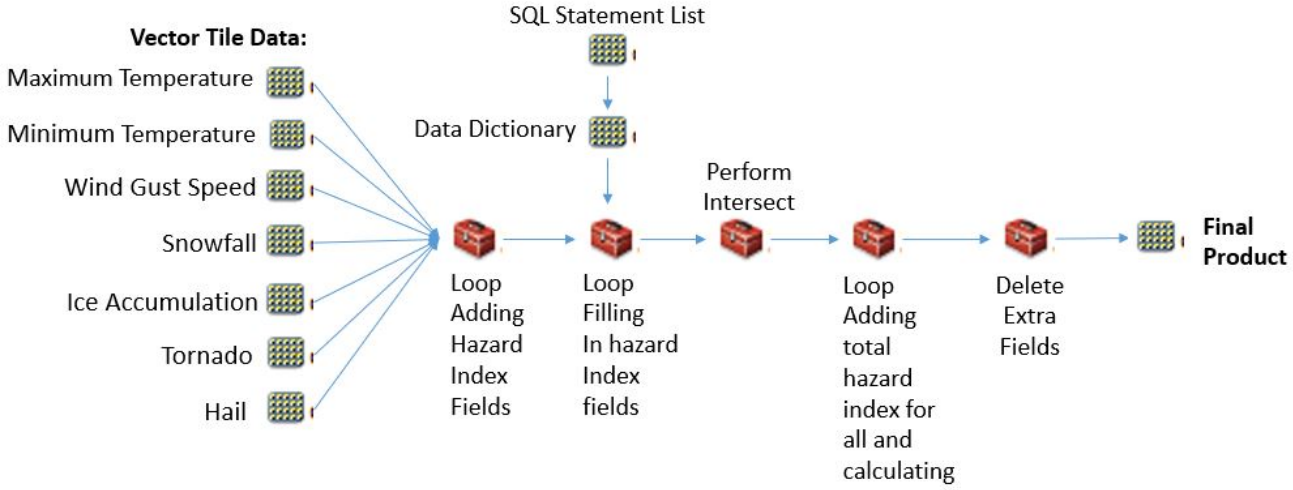


Figure 15. Model of WFHI vector polygons being used to create a hazard index

Variables are identified containing strings of Structured Query Language (SQL) statements. These statements are used to identify whether the weather attributes of each individual vector polygon should be defined as a low, medium, or high hazard index. A data dictionary is then defined. The key for this dictionary is the name of each weather vector polygon created. Another dictionary is the value of this key. Within this dictionary the key is the variable for the SQL statements mentioned above. The value corresponding to this key is a text string of low, medium, or high.

A ‘for loop’ is used to go through the weather vector polygon keys and values. If and else statements are used to create two lists of weather vector polygons at 5 x 5 km and 2.5 x 2.5 km resolutions. If and elif statements are then used to assign variables for names based on the weather vector polygon key. An example of this would be if the weather vector polygon key

starts with temp_max. If this is the case then one variable will be named maximum temperature while another will be named temp_max.

Several fields are then added to each weather vector polygon. These will be our hazard index fields. One is for weather values in each vector polygon which is partially named by the variables in the previous if and elif statement. This field will have a name that reflects the weather vector polygon where it is created. For discussion, this field will be referred to as its variable name (gridname). This field is important in providing field clarity after all the weather vector polygons are merged together. The second field added appears as a text string to show the hazard level of a weather element. This field will be referred to as its variable name (hazname). The third field shows the hazard index as a number of 0 for no hazard, 3 for low, 4 for medium, or 5 for high. This will be referred to by its variable name (haznum). The arcpy module function 'calculate field' is used to calculate the values for the gridname field based on the weather values field within the weather vector polygon.

A 'nested for loop' is used within the 'for loop' used to go through the weather vector polygon keys and values. This 'for loop' cycles through the keys that are the variables for the SQL statements. As it goes through each variable, the arcpy module function 'select layer by attribute management' is used. This function selects only attributes within the weather vector polygon that match the SQL statement represented by the variable. The arcpy module 'calculate field' is then used to insert a string text value associated with the variable for the SQL statement key into the hazname field along with the name of the weather element. This field is used in AGOL to identify the weather elements that are at a low, medium, or high hazard index value.

If, elif, and else statements are used to define the level within the hazard index based on the numbers for the haznum field. The module arcpy function 'calculate field' is then used to

calculate values for the haznum field. This completes the ‘for loop’ cycling through the keys that are the variables for the SQL statements. The module arcpy function ‘select layer by attribute management’ is used to select vector polygons with a hazname field value of null. The module arcpy function ‘calculate field’ is then used to generate a number of 0 for the field haznum. This is important later when the haznum fields of all the weather vector polygons are added together. Null values can not be added to integer values so the number 0 has to be generated for vector polygons with no hazard. This ends the ‘for loop’ used to go through the weather vector polygon keys and values.

A list is then created containing two sub lists. The first sub list contains a variable representing the list of weather vector polygons that are 2.5 x 2.5 km polygons and a variable representing the destination and name of a feature class we want to create. The second sub list is a mirror of the first sublist but contains the list of weather vector polygons that were created to be 5 x 5 km polygons and a slightly different name for the feature class.

A ‘for loop’ then cycles through each sub list within the list. The arcpy module function ‘Intersect’ takes the list of weather vector polygons and merge the polygons into one polygon. All fields are kept during this process. The new merged feature class is then added as the field ‘haz_index.’ The module arcpy function ‘select layer by attribute management’ is used to select all records and the function ‘calculate field’ adds all haznum fields together. This field will now symbolize hazard levels. As stated before, a number of 0 represents no hazard, 3 represents low, 4 represents medium, and 5 represents high. Anything higher than 5 represents multiple hazards in an area.

To help with functionality, extra fields in the merged feature classes are deleted. This is done by using a ‘for loop’ to go through the fields in the merged feature classes. If and elif

statements are used to select fields to be included in a new list. At the conclusion of the 'for loop' the arcpy module function 'delete fields' is used to delete all fields included in the new list. This completes all for loops and the script generating a hazard index in 2.5 x 2.5 km polygons and 5 x 5 km polygons.

Posting Data to ArcGIS Online

The data are now posted to ArcGIS Online. There are several steps to doing this. The first is a one time nonscripting process. The hazard index polygons are added into an ArcGIS Pro project. The symbology is built and the data projection is adjusted to match the base map coordinate system. A feature template was also created for both polygons. The hazard index polygons are then shared to AGOL using the 'share as web map' function. This creates a hosted feature layer made up of two layers of the data.

The second step in this process uses a script found on the ESRI ArcGIS Blog to automate updates from the ArcGIS Pro project to ArcGIS Online (Hibma 2017). Only minor changes were made to the script. The script starts by using the arcpy module function 'ArcGIS Project.' This allows access to the ArcGIS Pro project. Two more arcpy module functions are then used. These functions are create web layer SDDraft and stage service server. The first 'create web layer SDDraft' creates a service definition draft (SDDraft) of the hazard index polygons. The second 'stage service server' converts the SDDraft of the hazard index polygons into a service definition. The arcgis.gis module 'object GIS' then sets up an entry connection with AGOL. The url connecting to AGOL is provided, as well as the login credentials and password. The arcgis.gis module 'content.search' accesses the content manager and search for the web feature layer created, our user name, and the item type, which is service definition. The function 'update' from the arcgis.gis module is used to update the service definition. The function

arcgis.gis module function ‘publish’ sets an environmental overwrite for the hazard index polygons located in AGOL. Lastly, the arcgis.gis module function ‘share’ is used to set up sharing within AGOL.

Setting up a Dashboard Webapp in ArcGIS Online

In AGOL, a webmap of the feature layer created in ArcGIS Pro was built. This is required to build the Dashboard web application. To adjust the popup information for when individual vector polygons are selected, the popup configuration is used. With this the popup attributes were edited to only show certain fields and the field name was rewritten to be easier to read. Popup configuration was also used to list hazard indexes above low risk for specific weather. This was done using the hazname field generated during the building of the hazard index.

Visibility range and transparency were also set in the webmap. Visibility was set to show the 5 x 5 km polygons from a visibility range of 1:290,000 to a world view. The 2.5 x 2.5 km polygons were set to have a visibility of 1:290,000 down to room view. Transparency for both polygon layers was set at 50%.

With these webmap settings, several additional webmaps were created each representing a weather element. A new legend for each weather element was created. From the original webmap ‘share’ was used to create a dashboard operations webapp.

In the Dashboard operations webapp, the additional webmaps representing weather elements were added. Indicators of average weather were also added and tied into the hazard index webmap. This results in the indicators showing the average weather where the user is zoomed into. An address locator, map embedded legends, links websites, and the ability to change base maps were also added to the Dashboard operations webapp.

Task Scheduler and Batch Files

To fully automate the process of updating the hazard index webapp, a batch file was built (see Appendix C). This batch file references the path to the four scripts that update the hazard index web application. Each of the four scripts referenced have a path listed back to the execution file for Python 3.

Task scheduler is then used to activate the batch file at a designated time. The trigger was set at 10a.m. and 10p.m. daily. The action was program/script referencing the Python version 3 executable and the argument being the batch file. Conditions were set to wake the computer to run this task. The task was also set to run with the highest privileges and run whether the user is logged on or not.

CHAPTER 4

RESULTS, DISCUSSION & CONCLUSION

SPC Storm Report Results

The storm report script generates a dated storm report map showing the location and count of wind, tornados and hail events in and around the state of Tennessee for the day previous to when the script is run. The current output of the storm report map is in a png format.

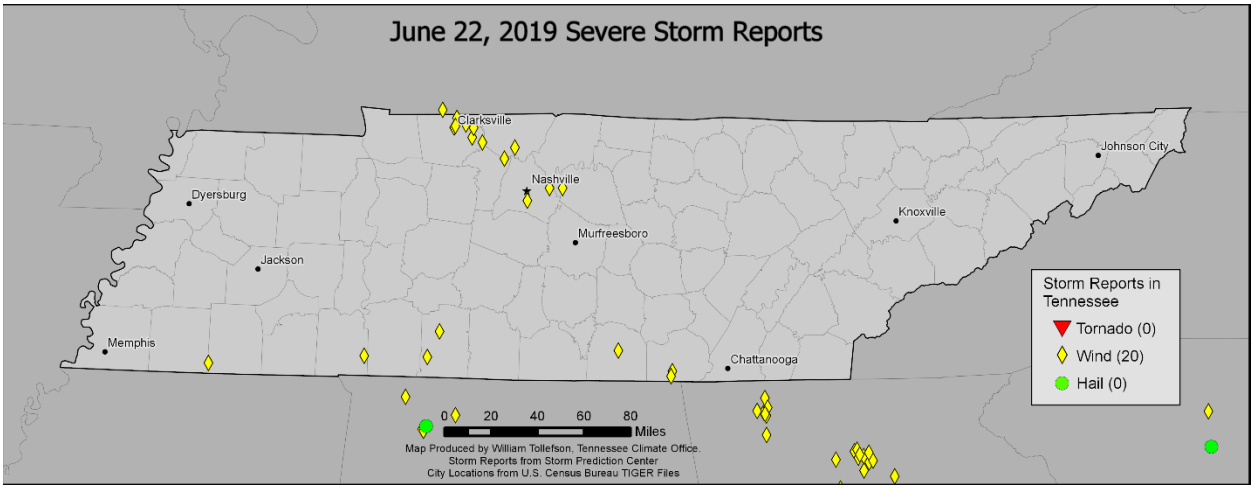


Figure 16. Storm report for June 05, 2019 showing one instance of wind in Tennessee

Inspection of the data generated from the SPC storm reports shows the script to be functioning properly. Point shapefiles produced are identical to the Longitude and Latitude provided in the CSV file. Additional information provided in the SPC storm reports CSV file appear in the shapefile attribute table. The png maps generated by the script show the correct storm report data for Tennessee, the correct date and count of events. The objective of the script to produce the daily storm report maps with increased efficiency was successful.

The storm report script and associated batch file will continue to produce storm reports until triggers or actions are canceled in task scheduler. Moving the script or batch file and still using them effectively require changes to location variables to reflect new locations to associated

feature classes and the locations of input and output data. A change of computers requires the new computer to have the module arcpy version 3.6.6 or higher installed. The current task scheduler settings for this script can be set for any time required. Currently the script is being executed at 8pm every night.

Weather Forecast Hazard Index Results

The Weather Forecast Hazard Index web application shows the maximum extreme forecast weather for the next 72 hours in the state of Tennessee. At a visibility range of 1:290,000 to a world view, the 5 x 5 km color coded polygons representing level of risk are seen covering the state of Tennessee. At a visibility of 1:290,000 down to room view the 2.5 x 2.5 km polygons showing the level of risk are visible. Clicking on an individual polygon opens a popup. At the top of this popup the weather elements with a risk level are labeled. In addition, the popup shows the forecast maximum extreme weather measurements for all weather elements in that polygon. The legend showing color coding for risk level is in the upper right. Below the web application legend is a short explanation of the measurements each risk level represents. Widgets on the left provide the average of extreme weather elements for the area that has been zoomed to.

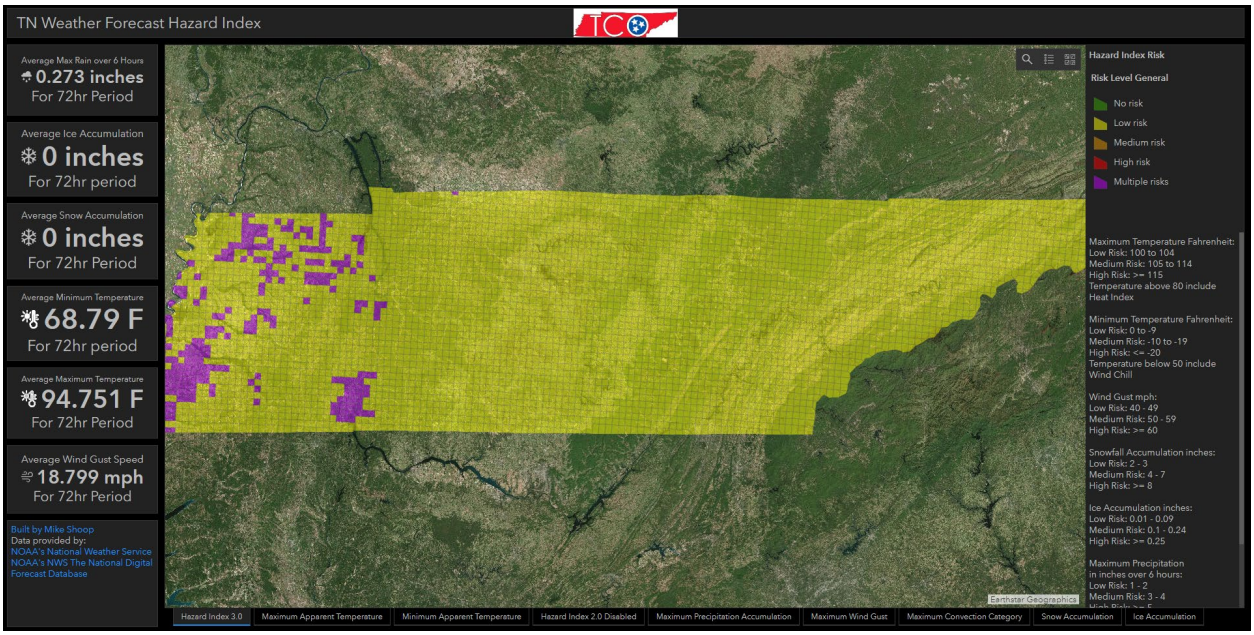


Figure 17. Screenshot of Weather Forecast Hazard Index taken on June 22, 2019

The main tab shows the Weather Forecast Hazard Index. The additional tabs show individual extremes of forecast weather for minimum apparent temperature, maximum apparent temperature, snow accumulation, ice accumulation, maximum wind gust, quantitative precipitation and convection categories for the next 72hours.

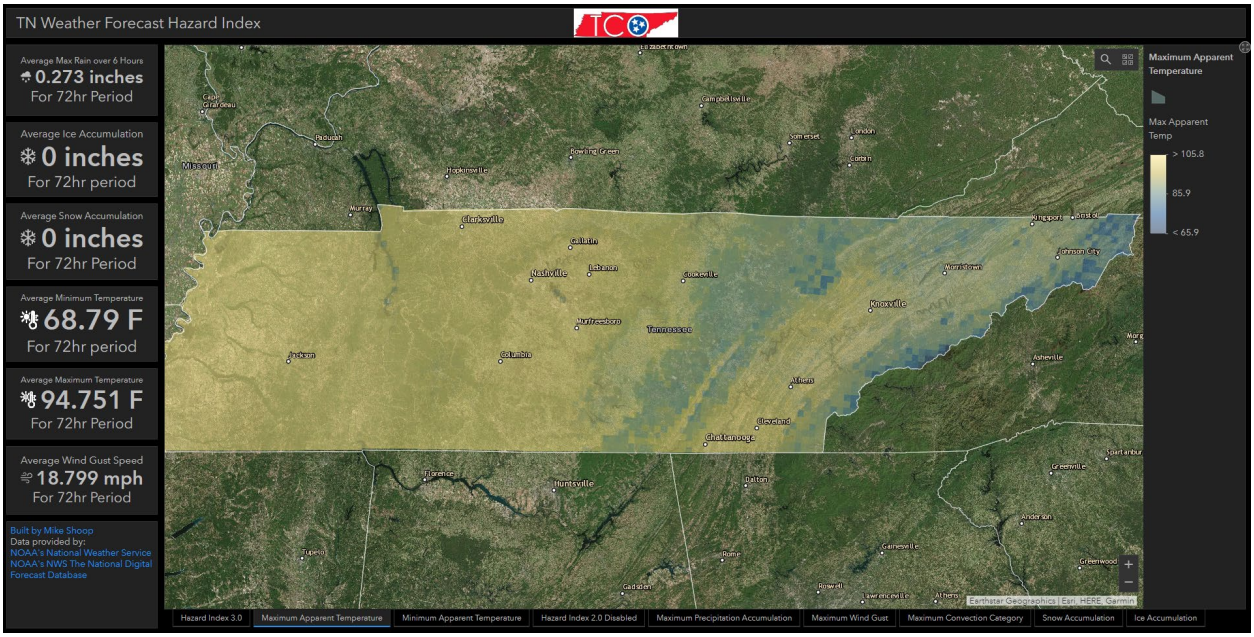


Figure 18. Screenshot of maximum apparent temperature taken on June 22, 2019

The Weather Forecast Hazard Index successfully downloads data from the NDFD website. Each GRIB2 file in the time stamped folder created by the first script shows the time it was downloaded which reflects the trigger time set within task scheduler. The GRIB2 data being mosaicked and converted to vector polygons was qualitatively confirmed to be accurate by comparing the data represented in the combined feature class to data shown at the National Weather Service (National Weather Service Corporate Image Web Team… 2019).

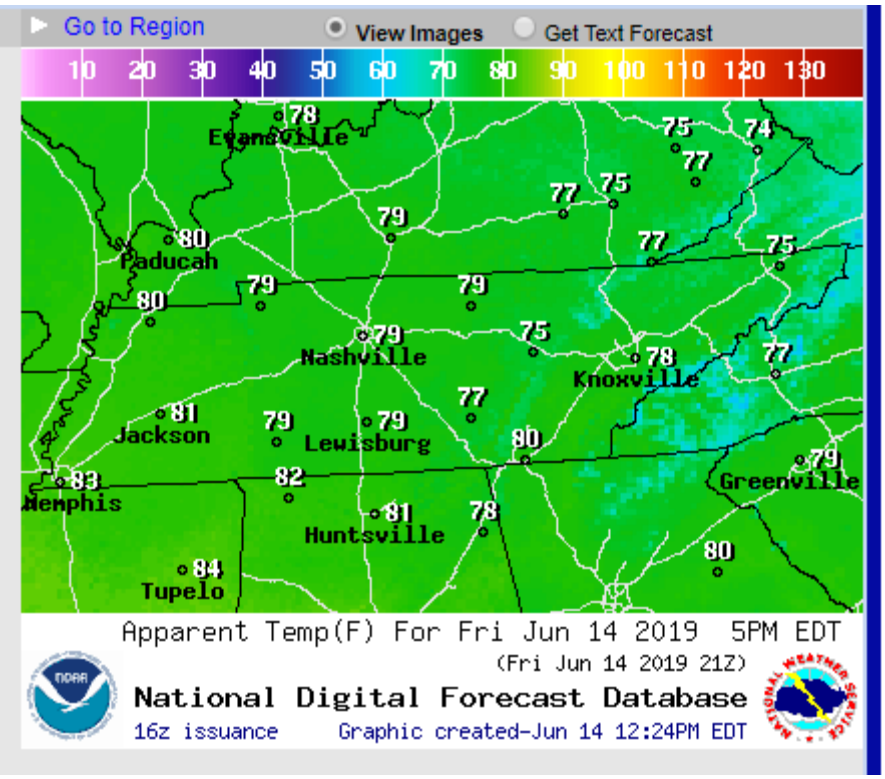


Figure 19. Example of NDFD data used to check final values for the combined feature class

This can be seen in the following figures comparing categorized convection forecast by the weather forecast hazard index and the storm prediction center 1-3 day forecast. The values shown are different but translate to the same data. A table is provided below to show the conversion between the NDFD data and the storm prediction center 1-3 day forecast.

Table 3. Conversion table of SPC Categorical Outlook Legend risk levels to NDFD categorical outlook values. Please note moderate and high conversions are based on documentation from documents from 2009 and are not completely verified

SPC Convection	NDFD categorical Outlook value
TSTM(light green)	Grid value of 2
1: Marginal risk(dark green)	Grid value of 3
2: Slight risk (yellow)	Grid value of 4
3: Enhanced risk (orange)	Grid value of 5
4: Moderate (red)	Grid value of 6-7
5: High (magenta)	Grid value of 8-10

In figure 19, convection categories of three and four can be seen. The category of three is represented as a light tan color while the category of four is shown as a dark red. The pattern created by these two categories can be seen projected on the day one forecast from the storm prediction center below for the state of Tennessee. This is due to the storm prediction center day one forecast slight category being higher than the day two and day three forecasts. For comparison, the 1-, 2-, and 3-day convection forecasts from the NWS site are shown in figures 20-23.

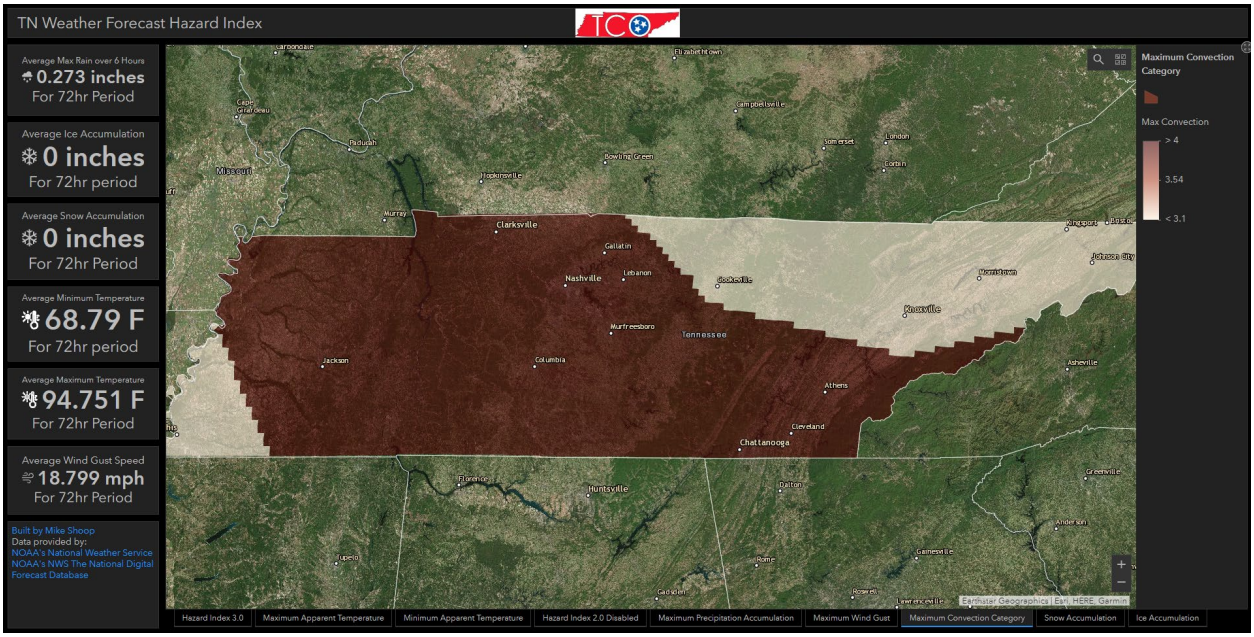
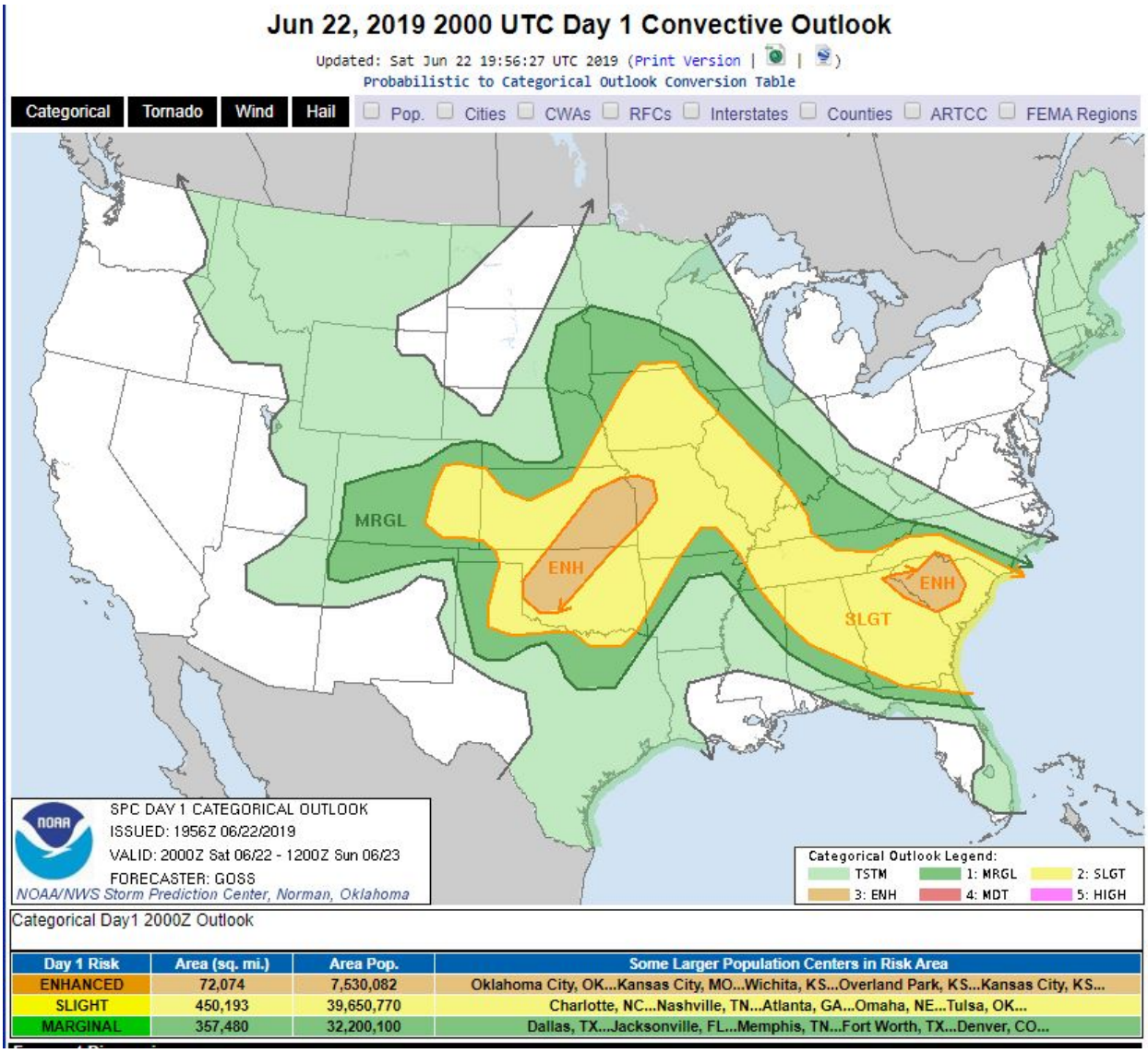


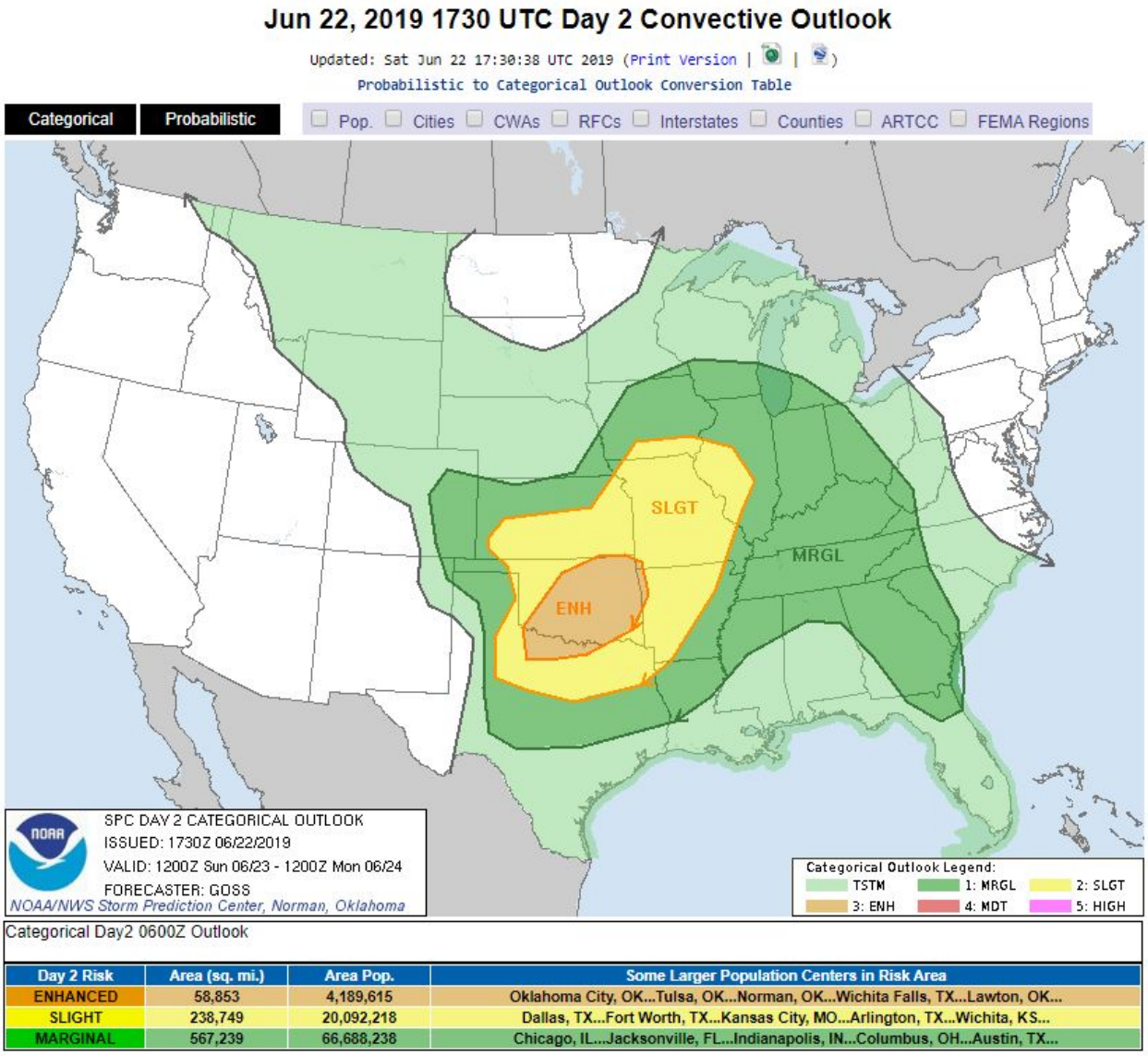
Figure 20. Convection category data from the combined feature class. This image was taken on June 22, 2019



Categorical Day1 2000Z Outlook

Day 1 Risk	Area (sq. mi.)	Area Pop.	Some Larger Population Centers in Risk Area
ENHANCED	72,074	7,530,082	Oklahoma City, OK...Kansas City, MO...Wichita, KS...Overland Park, KS...Kansas City, KS...
SLIGHT	450,193	39,650,770	Charlotte, NC...Nashville, TN...Atlanta, GA...Omaha, NE...Tulsa, OK...
MARGINAL	357,480	32,200,100	Dallas, TX...Jacksonville, FL...Memphis, TN...Fort Worth, TX...Denver, CO...

Figure 21. SPC Day 1 Convection Outlook showing pattern and values from Figure 4.5



Categorical Day2 0600Z Outlook

Day 2 Risk	Area (sq. mi.)	Area Pop.	Some Larger Population Centers in Risk Area
ENHANCED	58,853	4,189,615	Oklahoma City, OK...Tulsa, OK...Norman, OK...Wichita Falls, TX...Lawton, OK...
SLIGHT	238,749	20,092,218	Dallas, TX...Fort Worth, TX...Kansas City, MO...Arlington, TX...Wichita, KS...
MARGINAL	567,239	66,688,238	Chicago, IL...Jacksonville, FL...Indianapolis, IN...Columbus, OH...Austin, TX...

Figure 22. SPC Day 2 Convection Outlook

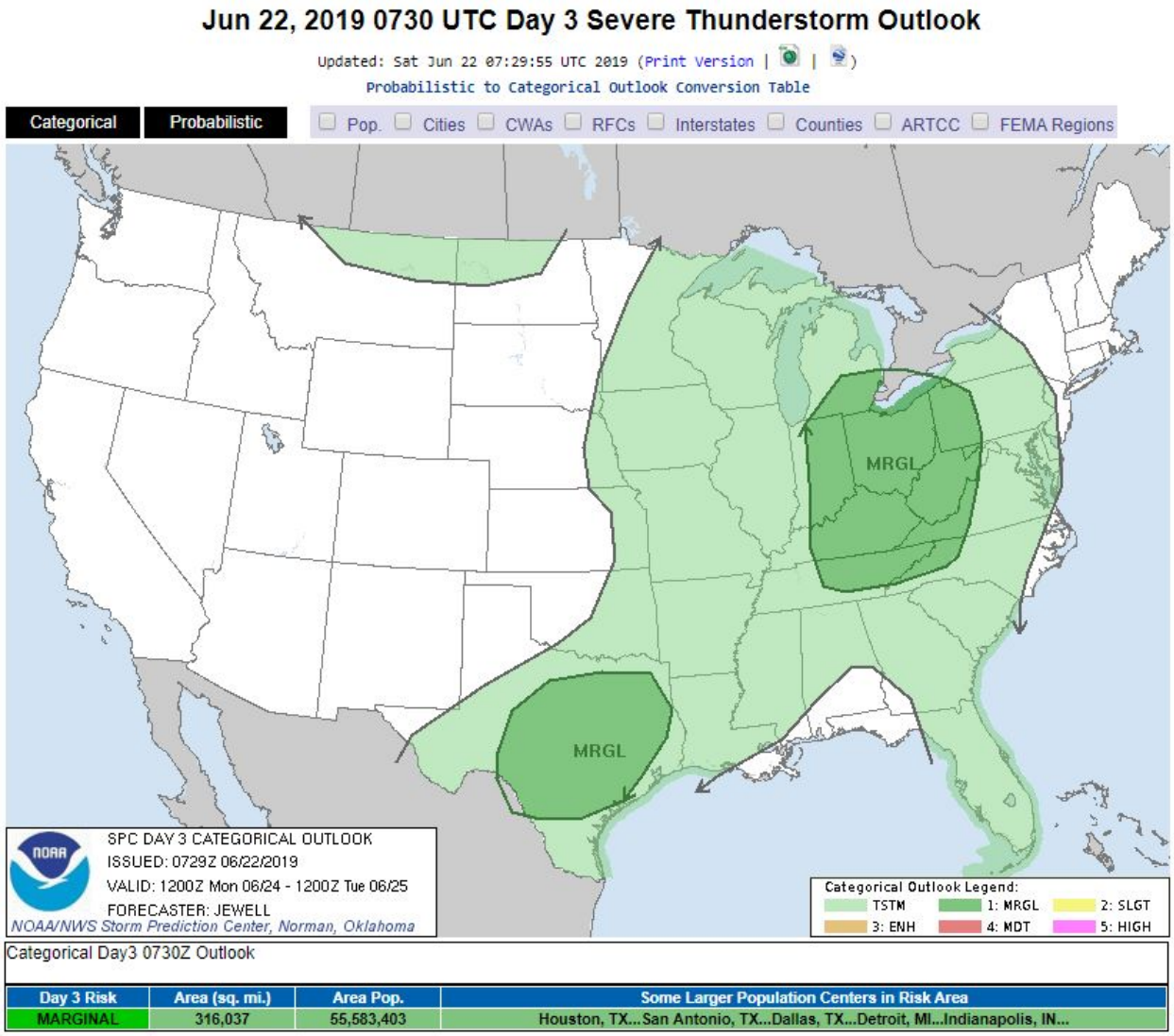


Figure 23. SPC Day 3 Convection Outlook

The scripts provided and associated batch file require changes to location variables to reflect new locations if changes are made to associated feature classes or the locations of input and output data. A change of computers requires the new computer to have the module arcpy version 3.6.6 or higher installed as well as gdal 2.3.3. The current task scheduler settings for this script can be set for any time required. Currently the script is triggered to be executed two times a day. The first trigger is 10am and the second trigger is 10pm.

SPC Storm Report Discussion

During the initial building of this script the process to write the csv files to a local file folder was unsuccessful. To resolve this problem, completed point shapefiles of the daily storm report data were used to generate csv files of the data from the SPC website. How to correctly write to a local file using the urllib.request module was learned during the production of the script for downloading data from the NDFD website. The initial script was written as the final for an independent study to learn Python and was written much earlier than other scripts within this thesis. The script while functional contains many inelegant pieces of code (see Appendix D). An example of this is the use of three 'insert.cursor' functions from the arcpy module at the bottom of the script to replace the Avalon module built at the top of the script. This was done due to the variable wizard needing to change for each csv file. A rework of the script addressed the problem of the ability to write the csv files to a local folder directly as well as fixing the issue stopping the Avalon module from functioning.

Other issues outside of the script existed that had to be resolved. A count of storm report incidents within Tennessee was created in the original mxd map template for the daily storm reports and this needed to show count of storm report types in the state for that day. A similar issue that was encountered when setting up dynamic text of the date for the original mxd map template. The storm report data was for the day before the current date. To solve this problem several steps were taken. The first was the transfer of the original mxd map template to an ArcGIS Pro project. Dynamic text within an ArcGIS Pro project is much more dynamic and allowed the setup of dynamic text for the counts based on rows of data for each of the clipped feature classes. Dynamic text for the date of the storm report was created using the Date field that is updated in the Boundary feature class.

Hazard Index Discussion

In the first script downloading data from the NDFD website included data for not only the continental United States but also the Mid-Atlantic and Central Mississippi Valley subsections (see Appendix D). Ultimately due to the reasons briefly explained in the Hazard Index web application data section and concerns about processing time only the code to pull data for the continental United States remained in the final script.

Originally data downloaded using urllib.request was downloaded to the location of the script. The os module was used to read items in the home folder of the script as a list. A 'for loop' was constructed to go through that list. Inside of the 'for loop,' an 'if' statement was used to identify GRIB2 files. The shutil module was then used to copy GRIB2 files from the home folder of the script to the timestamped file folder created for the data. This was removed in the last version of this script when the setting the working directory using the os module was discovered.

The first challenge in the second script was finding a way to look at the measurements of each raster within the GRIB2 file and to manipulate those rasters. There were several ways this could possibly be done. The module 'gdal' is able to manipulate GRIB2 files but a method to extract individual rasters with this module was not discovered. The module 'pygrib' was also investigated. This method was not used due to the technical issues involved in running this module on a Windows computer (Whitaker 2014). This lead to using the arcpy module function 'extract subdataset management.' The function 'extract subdataset' was not able to pull rasters out of all the GRIB2 files.

Another challenge arose when the arcpy module function extract subdataset management began to pull rasters out of the GRIB2 files. During the 'for loop' if a number in the raster in the

GRIB2 array was requested that didn't exist the script would shut down. Use of the arcpy functions 'list rasters' and 'count' would not show the true count of rasters within the GRIB2 file. The use of 'try and except' also did not prevent the script from shutting down. This was when the 'gdal' module was advised for use from representatives at MDL and SRCC. Learning to install modules took some time and during this time the concepts of using the 'conda' and 'pip' Python installers were introduced. The 'gdal' module successfully counts the number of rasters within the GRIB2 files. Version control became important due to the current version of the 'gdal' module being unable to count the number of rasters within the GRIB2 files.

The script had to be updated on multiple occasions. The first was due to the tornado and hail probability rasters having slightly different cell alignments from the other GRIB2 files. This caused the final product in the hazard index web application to contain large numbers of polygon slivers. To address this the arcpy module 'extract by mask' was used to not only clip the data to the boundaries of Tennessee but to also resample the data to mirror the cell dimensions and placement of the maximum temperature TIFF.

The second update came from suggestions for improvements to the hazard index web application. Originally only the 2.5 x 2.5 km polygons were used. These polygons had difficulty loading in ArcGIS Online, often resulting in many vector polygons not becoming visible when zoomed to the extent of the state. To improve the presentation of the hazard index web application it was suggested that 5 x 5 km polygons should be used for smaller scale views of the data. With this in mind, resampling of the TIFF files before they were converted to vector polygons was coded into the script.

The third update was a change in GRIB2 files being used in this thesis and the rearrangement a critical process (see Appendix D). The arcpy module 'extract by mask' was

moved to execute directly after the 'extract subdataset management.' Its original location was after the GRIB2 files were mosaicked back to together. Moving the execution of this module to an earlier date in the script greatly increased the overall speed of the second script.

The original third script was completed with few challenges (see Appendix D). A review of the initial hazard index web application by climatologists at NWS-Morristown and the SRCC and planners/emergency managers at the Tennessee Department of Transportation and ETSU resulted in several suggestions for improvements. One suggestion was for weather conditions forecast to cause a hazard to be clearly listed. Another was that if multiple hazards were present in a polygon then that should be color-coded. To do this, most of the script was rewritten with more advanced Python coding implemented such as nested dictionaries and lists. The original script only applied a hazard index to the final intersected polygon as opposed to the current script which applies the hazard index to all weather condition polygons. While rewriting the script, the code for deleting extra fields was also rewritten. The new script uses a series of 'startswith' statements. To build a list of fields within the final polygons to delete. The original just had the names of the fields written in a list.

Other Challenges

A challenge faced by both the Storm Report script and several of the Hazard Index scripts was the batch file accessing the Python 3 executable. This was resolved by adding the path of the executable in front of the path to the scripts. Task scheduler not running during its scheduled time also posed a challenge. This was solved by changing the way the task was set up. Instead of a direct path to the script in program/script field, the path to the python executable was placed there and the path to the script was added to the 'Add arguments' field.

Limiting Factors

There were several limiting factors in creating the Storm Reports script and the Hazard Index script. The first was a limited knowledge of Python scripting. An independent study that was an introduction to Python was taken in the spring of 2018 and an advanced Python course was taken in the spring of 2019. Minimal code development help was available due to a dearth of Python scripting experience within the Department of Geosciences.

Late in the thesis project it was learned that rasters could not be transferred to ArcGIS Online without having originated from an enterprise server. Partly due to a limited knowledge of servers and specifically enterprise servers, vector polygons were used in the final web application. This turned out to be more practical due to the ability to use the attribute table of the vector polygons to list large amounts of data.

Looking at the Delaware web application, the representation of measurements on the right panel clearly show forecast weather conditions. A presentation similar to this and outside of ArcGIS Online would have been preferred but the building of a site like this requires Javascript. This is a scripting language that will be learned in the future. Time constraints did not allow for this during the creation of the thesis scripts.

Time was the single biggest limiting factor of this thesis. This is probably the biggest limiting factor for most theses. This thesis was started in August of 2018 due to technical problems with the previous thesis topic that focused on 3D interior mapping. The window for graduation was the summer of 2019. Research had to be performed, new Python modules, functions and statements had to be learned, and scripts had to be written quickly.

Future Research and Product Development

There are several processes to add to the Storm Reports and the Hazard Index web application that would improve automation performance or product functionality. A few new automation of processes and products can also be built.

Automation of the daily storm reports can be expanded to compile the data being pulled from the SPC Storm Reports starting at the beginning of the month and ending on the last day of the month into one shapefile. This shapefile can then be used to generate the total monthly storm reports map pdf. A similar concept to compiling the daily storm reports is showing all the storm report data for the state over a longer period of time such as five or ten years with current data that are added into a web application every day. This is similiar to a product produced and maintained by the state of North Carolina called the Severe Storm Reports Tool (Severe Storm Reports Tool 2004).

A partial or full automation of the observed mean temperatures and precipitation maps for the month could also be developed. The modules 'urllib2' or 'urllib.request' could be used to pull the data from the University of Oregon PRISM website. Subsequently, a similar process to the automation of the daily storm report maps could be used to generate the observed weather (see Appendix B). Percent Normal weather would require raster math of the monthly data against the normal. It is unclear yet how the tables for these maps would be changed.

Currently the full process for updating the Hazard Index web application takes 15-20 minutes to run. Several ideas have been considered to reduce this time. Most of this long processing time occurs during the 'for loop' extracting, converting, and merging the GRIB2 rasters. To reduce this time, we will attempt to use the module multiprocessing to run multiple GRIB2 files through the 'for loop' simultaneously by using multiple individual processors to

each run that part of the script using a different GRIB2 file. This should reduce the processing time of updating the Hazard Index web application by about 40%.

With an established automated system for updating the hazard index web application, incorporating more complex hazards could increase the overall value of the product. The first two complex hazards being examined are heat index and wind chill. Even though these data are available from NDFD, the formula for each is relatively simple. More complex formulas could be built to show more accurate measurements for this hazard for the state of Tennessee. A separate web application for drought index reusing large portions of the script used to automate the hazard index web application has been considered. This could be a valuable product that would require a low amount of production time.

A product that could generate additional interest in the hazard index web application is an automated email that is delivered to a user of the product if the address provided falls within an area forecast to contain a hazard. Currently the automated email coding is being built into each script to alert faculty of script errors. Using risk polygons to identify address points containing email data is a project that will require additional research. This idea came from a patent filed for a GIS-based automated weather alert notification system (Sznaider et al. 2004).

Product Continuity & Preservation

This thesis provides two major product deliverables. For the automated script for daily storm reports the Python script is shown here (see Appendix C). It will also be downloaded onto a designated Department of Geosciences, ETSU computer along with an updated storm report ArcGIS Pro project and a batch file. The batch file will be set to run in Task Scheduler at a time designated by faculty. In addition, a preliminary Dashboard web application was developed for a

possible future product similar to the North Carolina SCO Severe Storm Reports Tool (Severe Storm Reports Tool… 2019).

The WFHI Dashboard web application will be available on ArcGIS Online. The Python scripts for automating the Hazard Index are shown in this thesis (see Appendix C). Like the automated script for the daily storm reports, the Python scripts will also be downloaded onto a designated department computer with the ArcGIS Pro project tied to the web feature layer. A batch file will be set up to run in Task Scheduler at a time designated by faculty. All paths for both projects will be changed to reflect the new location of items in the department computer in which they are placed.

Conclusion

The TCO has short- and long-term goals focused around two questions: 1) How can the monthly climate report be improved upon and produced more efficiently? and 2) What value-added, data-driven products should the TCO develop based on local and statewide needs? These were addressed in the following manner:

- The automation of daily storm report maps meet goals focused around the first question.
- The development of the Weather Forecast Hazard Index web application product meets goals centered around the second question. Most importantly, it provides value in helping decision-makers improve their understanding of potentially dangerous forecasted weather that could effect communities of the state.

Overall this thesis has covered the following:

- Explained the importance of climate and weather data.
- Explained the general hierarchy of climate data services.
- Provided information about the efforts to develop the TCO.

- Provided a brief overview of the Python scripting language.
- Explained in detail how daily storm report maps within the monthly climate report were automated.
- Explained what the hazard index web application is, how it functions, how it was built, and how it was automated using Python scripting.
- Explained challenges encountered during the building of the script to automate the daily storm reports and the scripts for the hazard index web application. This is to give the reader an idea of problems that maybe encountered when repeating the work described here.
- Discussed the limitations encountered when completing both of these projects.
- Discussed future work on both the automation of the daily storm reports and the scripts for the hazard index web application as well as products and automation similar to the work discussed in this Thesis that would add additional value to the TCO.

The WFHI allows the TCO to provide a valuable service to the state in the form of giving decision makers an easily readable tool defining forecast hazards for the state of Tennessee. Perhaps more importantly, Python scripting gives a framework for building additional products in the future with this thesis acting as a guide. The same can also be said for the automation of the daily storm reports script. Not only does it automate a time-consuming process but it provides a framework for automating additional processes.

REFERENCES

2.6. Statements and Expressions [Internet]. c2019. Runestone Interactive. [cited 2019 Apr 1]. Available from: https://runestone.academy/runestone/books/published/thinkcspy/SimplePythonData/StatementsandExpressions.html

AASC Recognized State Climate Office [Internet]. 2015. American Association of State Climatologists; [cited 2019 Mar, 10]. Available from http://stateclimate.org/arsco

About MDL: History [Internet]. 2018. US Department of Commerce & NOAA. [cited 2018 Dec 10]. Available from: https://www.weather.gov/mdl/about_history

About the SPC [Internet]. 2015. National Weather Service. [cited 2019 Mar 10]. Available from https://www.spc.noaa.gov/misc/aboutus.html

ACIS [Internet]. c2017. Applied Climate Information System [cited 2019 Mar, 10] Avaible from http://www.rcc-acis.org/aboutacis_overview.html

Agriculture Essentials [Internet]. c1994-2019. Oklahoma Mesonet [cited 2019 Mar, 17]. Available from http://www.mesonet.org/index.php/agriculture/monitor

Aizenman, H, Michael Grossberg, David Jones, Nick Barnes, Jason Smerdon, Kevin Anchukaitis, and Julien Emile Geay 2015. Web Based Visualization Tool for Climate Data Using Python [Internet] [cited 2019 June 9]. Available from: https://www.researchgate.net/profile/Michael_Grossberg/publication/267245402_Web_Based_Visualization_Tool_for_Climate_Data_Using_Python/links/5592795e08ae47a34910fb17.pdf

Arzberger P., Schroeder P., Beaulieu A., Bowker G., Casey K., Laaksonen L., … Wouters P. 2004. Promoting Access to Public Research Data for Scientific, Economic, and Social Development. Data Science Journal [Internet] [cited 2018 Nov 3]; 135–152. Available from: http://doi.org/10.2481/dsj.3.135

Barreira C., Mer F., Ines A. V. M., Baethgen W. E., Berterretche M., Souza J. S., … Han E. 2018. SIMAGRI: An agro-climate decision support tool. *Computers and Electronics in Agriculture* [Internet] [cited 2019 Jan 10]. 0–1. Available from: http://doi.org/10.1016/j.compag.2018.06.034

Basic date and time types [Internet]. c2001-2019. Python Software Foundation. [cited 2019 Mar 20]. Available from: https://docs.python.org/3/library/datetime.html

Beazley D. M. 2009. Python Essential Reference. Fourth Edition. Addison-Wesley Professional.

Boeing G. 2017. OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. Computers, Environment and Urban Systems[Internet] [cited 2019 Jan 10]. *65*: 126–139. Available from: http://doi.org/10.1016/j.compenvurbsys.2017.05.004

Bonnin G. M., Martin D., Lin B., Parzybok T., & Riley D. 2004. NOAA Atlas 14: Precipitation-Frequency Atlas of the United States. *1*: 1–7.

Brooks M. S. 2013. Accelerating Innovation in Climate Services: the 3E's for climate service providers. Bulletin of the American Meteorological Society [Internet] [cited 2019 Jan 10]. Available from: http://doi.org/10.1175/bams-d-12-00087 130219081617002

Brown J. L. 2014. SDMtoolbox: A python-based GIS toolkit for landscape genetic, biogeographic and species distribution model analyses. *Methods in Ecology and Evolution* [Internet] [cited 2019 Jan 10]; *5*(7): 694–700. Available from: http://doi.org/10.1111/2041-210X.12200

Brown J. L., Bennett J. R., & French C. M. 2017. SDMtoolbox 2.0: The next generation Python-based GIS toolkit for landscape genetic, biogeographic and species distribution model analyses [Internet] [cited 2019 Jan 10]. Peer J. Available from: doi:10.7287/peerj.4095v0.2/reviews/2

Callahan C. 2014. Developing the Delaware Weather Hazard Index (DWHI). [Internet] [cited 2018 Sept 12]. Available from: https://www.ag.ndsu.edu/cpasw/documents/program-pdfs/session-2/05-callahan-cpasw18-tuesdaysession2-de.pdf

Changnon S. A., Lamb P. J., & Hubbard K. G. 1990. Regional Climate Centers: New Institutions for Climate Services and Climate-Impact Research. Bulletin of the American Meteorological Society [Internet] [cited 2019 Jan 10]; 71(4): 527-537. Available from: doi:10.1175/1520-0477(1990)0712.0.co;2

Climate Perspectives Tutorial [Internet]. c2007. Southeast Regional Climate Center. [cited 2019 Mar 31]. Available from: http://www.sercc.com/perspectives_tutorial.php

Climate Service Partners [Internet]. 2018. US Department of Commerce & NOAA. [cited 2019 Mar 10]. Available from: https://www.weather.gov/climateservices/partners

Coletti A., Howe P. D., Yarnal B., & Wood N. J. 2013. A support system for assessing local vulnerability to weather and climate. *Natural Hazards* [Internet] [cited 2019 Feb 15]; *65*(1): 999–1008. Available from: http://doi.org/10.1007/s11069-012-0366-3

Cook B. I., Seager R., & Miller R. L. 2011. Atmospheric circulation anomalies during two persistent North American droughts: 1932–1939 and 1948–1957. Climate Dynamics. 36:11-12. 2339-2355.

Create Thiessen Polygons [Internet]. n.d.. ESRI. [cited 2019 Jan 5]. Available from: https://pro.arcgis.com/en/pro-app/tool-reference/analysis/create-thiessen-polygons.htm

CSV File Reading and Writing [Internet]. c2001-2019. Python Software Foundation. [cited 2019 Mar 20]. Available from: https://docs.python.org/3/library/csv.html

Dateutil - powerful extensions to datetime [Internet]. c2016. [cited 2018 Nov 15]. Available from https://dateutil.readthedocs.io/en/stable/

Degaetano A. T., Brown T. J., Hilberg S. D., Redmond K., Robbins K., Robinson P., . . . Mcguirk M. 2010. Toward Regional Climate Services. Bulletin of the American Meteorological Society. *91*(12): 1633-1644. doi:10.1175/2010bams2936.1

Dima M., & Lohmann G. 2007. A hemispheric mechanism for the Atlantic Multidecadal Oscillation. Journal of Climate. 20(11). 2706-2719.

Dion E., Young D., & Tew M. 2019. NATIONAL DIGITAL FORECAST DATABASE and LOCAL DATABASE DESCRIPTION and SPECIFICATIONS. National Weather Service (US). Issue brief No: 10-201. Available from: https://www.nws.noaa.gov/directives/sym/pd01002001curr.pdf

Dixon P. G., Mercer A. E., Grala K., & Cooke W. H. 2014. Objective identification of tornado seasons and ideal spatial smoothing radii. Earth Interactions [Internet] [cited 2019 Mar 17]; 18(2): 1-15. Availiable from: https://journals.ametsoc.org/doi/full/10.1175/2013EI000559.1 doi:10.1175/2013EI000559.1

Dixon P. G., Mercer A. E., Choi J., & Allen J. S. 2011. Tornado risk analysis: Is Dixie alley an extension of tornado alley?. Bulletin of the American Meteorological Society [Internet] [cited 2019 Mar 17]; 92(4): 433-441. Availiable from: https://journals.ametsoc.org/doi/abs/10.1175/2010BAMS3102.1 doi:10.1175/2010BAMS3102.1

Dutton J., & O'Brien J. c2018. GEOG 489 - Advanced Python Programming for GIS. Pennsylvania State University [Internet] [cited 2019 Mar 17]. Available from: https://www.e-education.psu.edu/geog489/home.html

Faiers G. E., Grymes III J. M., Keim B. D., & Muller R. A. 1994. A reexamination of extreme 24-hour rainfall in Louisiana, USA. Climate Research [Internet] [cited 2019 Mar 15]; 4(4): 25-31. Available from: https://pdfs.semanticscholar.org/581d/0f2282303fb5adc240cdfce21d0e90acedc5.pdf

Forecasts and Service [Internet]. 2017. US Department of Commerce & NOAA. [cited 2019 Mar 10]. Available from: https://www.weather.gov/about/forecastsandservice

Germain H. J. n.d.. Strings. University of Utah [Internet] [cited 2019 Mar 20]. Available from: https://www.cs.utah.edu/~germain/PPS/Topics/strings.html

GIS [Internet]. c2016-2019. ESRI. [cited 2019 Jan 5]. Available from: https://esri.github.io/arcgis-python-api/apidoc/html/arcgis.gis.toc.html

Granger B. E., & Hunter J. D. 2011. Python: An Ecosystem for Scientific Computing. Computing in Science & Engineering [Internet] [cited 2019 Apr 25]; 13–21. Available from: https://pdfs.semanticscholar.org/e53e/c3a7325cdef1c654009df7108f016668a6e2.pdf

GRIB Data Format used for the COSMO-Model System [Internet]. 2013. Consortium for Small Scale Modeling. [cited 2018 Oct 10]. Available from http://www.cosmo-model.org/content/model/documentation/grib/grib_features.htm

Hibma K. 2017 Mar 14. Updating your hosted feature services with ArcGIS Pro and the ArcGIS API for Python. ESRI [Internet] [cited 2019 Mar 19]. Available from: https://www.esri.com/arcgis-blog/products/api-python/analytics/updating-your-hosted-feature-services-with-arcgis-pro-and-the-arcgis-api-for-python/

Hoerling M. P., Kumar A., & Zhong M. 1997. El Niño, La Niña, and the nonlinearity of their teleconnections. Journal of Climate [Internet] [cited 2019 Mar 15]; 10(8): 1769-1786. Available from: https://journals.ametsoc.org/doi/full/10.1175/1520-0442%281997%29010%3C1769%3AENOLNA%3E2.0.CO%3B2 *doi:10.1175/1520-0442(1997)010<1769:ENOLNA>2.0.CO;2*

Houston T. n.d.. Regional Climate Centers. National Oceanic and Atmospheric Administration. [Internet] [cited 2019 Feb 15]. Available from: https://www.ncdc.noaa.gov/customer-support/partnerships/regional-climate-centers

Interactive Map tracking tropical storms [Internet]. 2019. SRCC. [cited 2019 Mar10]. Available from: https://www.srcc.lsu.edu/tropical_desk

Jones, D. A., Wang, W., & Fawcett, R. 2009. High-quality spatial climate data-sets for Australia. Australian Meteorological and Oceanographic Journal [Internet] [cited 2019 Mar 15]; *58*: 233–248. Available from: https://pdfs.semanticscholar.org/e407/6004f330b2f2130ca86402a35c2c732b803d.pdf

Karl T. R., Gleason B. E., Menne M. J., McMahon J. R., Heim Jr R. R., Brewer M. J., ... & Groisman P. Y. 2012. US temperature and drought: Recent anomalies and trends. Eos, Transactions American Geophysical Union [Internet] [cited 2019 Mar 15]; 93(47): 473-474. Available from: https://agupubs.onlinelibrary.wiley.com/doi/epdf/10.1029/2012EO470001 doi:10.1029/2012EO470001

Keim B. D., Muller R. A., & Stone G. W. 2007. Spatiotemporal patterns and return periods of tropical storm and hurricane strikes from Texas to Maine. Journal of climate [Internet] [cited 2019 Mar 15]; 20(14): 3498-3509. Available from: https://journals.ametsoc.org/doi/full/10.1175/JCLI4187.1 doi:10.1175/JCLI4187.1

Kelvin to Fahrenheit [Internet]. 2018. Wight Hat Ltd. [cited 2018 Oct 15]. Available from: https://www.metric-conversions.org/temperature/kelvin-to-fahrenheit.htm

Kentucky Mesonet Agriculture Tool [Internet]. n.d.. Kentucky State Climate Office. [cited 2019 April 1]. Available from: http://kymesonet.org/ag.html

Kilogram force per square meter to Inches of Water conversion [Internet]. 2018. Wight Hat Ltd. [cited 2018 Oct 15]. Available from: https://www.metric-conversions.org/pressure/kilogram-force-per-square-meter-to-inches-of-water.htm

Klein W. H. 2018. the precipitation program of the Techniques Development Laboratory. Bulletin of the American Meteorological Society [Internet] [cited 2019 Mar 15]; *48*(12): 890–895. Available from: https://journals.ametsoc.org/doi/abs/10.1175/1520-0477-48.12.890 doi:10.1175/1520-0477-48.12.890 T

Kumar N. 2019, February 22. GET and POST requests using Python. Geeks for geeks [Internet] [cited 2019 Feb 22]. Available from: https://www.geeksforgeeks.org/get-post-requests-using-python/

Library of Congress. 2009 Sep 28. Sustainability of Digital Formats: Planning for Library of Congress Collections. Library of Congress. [Internet] [cited 2019 Mar 20]. Available from https://www.loc.gov/preservation/digital/formats/fdd/fdd000022.shtml

Logan K. E., Brunsell N. A., Jones A. R., & Feddema J. J. 2010. Assessing spatiotemporal variability of drought in the US central plains. Journal of Arid Environments. [Internet] [cited 2019 Apr 10]; 74(2): 247-255. Available from: https://www.sciencedirect.com/science/article/pii/S0140196309002572 doi: 10.1016/j.jaridenv.2009.08.008

McCabe G. J., Betancourt J. L., Gray S. T., Palecki M. A., & Hidalgo H. G. 2008. Associations of multi-decadal sea-surface temperature variability with US drought. Quaternary International [Internet] [cited 2019 Apr 10]; 188(1): 31-40. Available from: https://www.sciencedirect.com/science/article/pii/S1040618207002017 doi:10.1016/j.quaint.2007.07.001

Mesonet Essentials: An introduction to mesonets, their value, and how they work [Internet] c2019. Campbell Scientific. [cited 2019 Apr 1]. Available from: https://www.campbellsci.com/mesonets

Meters per second to Miles per hour [Internet]. 2018. Wight Hat Ltd. [cited 2018 Oct 15]. Available from: https://www.metric-conversions.org/speed/meters-per-second-to-miles-per-hour.htm

Meters to Inches [Internet]. 2018. Wight Hat Ltd. [cited 2018 Oct 15]. Available from: https://www.metric-conversions.org/length/meters-to-inches.htm

Modules [Internet]. c2001-2019. Python Software Foundation. [cited 2019 Mar 20]. Available from: https://docs.python.org/3/tutorial/modules.html

Moore T. W., & Dixon R. W. 2011. Climatology of tornadoes associated with Gulf Coast-landfalling hurricanes. Geographical Review. [Internet] [cited 2019 Apr 10]; 101(3): 371-395. Available from: https://www.jstor.org/stable/41303640?seq=1#metadata_info_tab_contents

Müller W. A., Frankignoul C., & Chouaib N. 2008. Observed decadal tropical Pacific–North Atlantic teleconnections. Geophysical Research Letters [Internet] [cited 2019 Mar 10]; 35(24): Available from: https://agupubs.onlinelibrary.wiley.com/doi/full/10.1029/2008GL035901

doi:10.1029/2008GL035901

Multidimensional Raster Types [Internet]. c2016. ESRI. [cited 2019 Jan 5]. Available from: http://desktop.arcgis.com/en/arcmap/10.3/manage-data/raster-and-images/multidimensional-raster-types.htm

National Digital Forecast Database (NDFD) [Internet]. n.d.. National Centers for Environmental Information. [cited 2018 Mar 19]. Available from: https://www.ncdc.noaa.gov/data-access/model-data/model-datasets/national-digital-forecast-database-ndfd

National Oceanic and Atmospheric Administration (NOAA) - National Climatic Data Center [Internet]. c2018. Federal Laboratory Consortium for Technology Transfer. [cited 2019 Mar 10]. Available from: https://www.federallabs.org/labs/national-oceanic-and-atmospheric-administration-noaa-national-climatic-data-center

National Weather Service Corporate Image Web Team. n.d.. "Graphical Forecasts - Tennessee." NOAA Graphical Forecast for Tennessee. National Weather Service [Internet] [cited 2019 Mar 20]. Available from: https://graphical.weather.gov/sectors/tennessee.php.

NDFD Spatial Reference System [Internet]. n.d.. US Department of Commerce & NOAA. [cited 2019 Mar 18]. Available from: https://www.weather.gov/mdl/ndfd_srs

Needham H. F., & Keim B. D. 2014. Correlating storm surge heights with tropical cyclone winds at and before landfall. Earth Interactions [Internet] [cited 2019 Feb 20]; 18(7): 1-26. Available from: https://journals.ametsoc.org/doi/full/10.1175/2013EI000527.1 doi:10.1175/2013EI000527.1

Nogueira R. C., & Keim B. D. 2011. Contributions of Atlantic tropical cyclones to monthly and seasonal rainfall in the eastern United States 1960–2007. Theoretical and Applied Climatology [Internet] [cited 2019 Mar 20]; 103(1-2): 213-227. Available from: https://www.researchgate.net/publication/252455669_Contributions_of_Atlantic_tropical_cyclones_to_monthly_and_seasonal_rainfall_in_the_eastern_United_States_1960-2007 doi:10.1007/s00704-010-0292-9

Nogueira R. C., Keim B. D., Brown D. P., & Robbins K. D. 2013. Variability of rainfall from tropical cyclones in the eastern USA and its association to the AMO and ENSO. Theoretical and applied climatology [Internet] [cited 2019 Mar 20]; 112(1-2): 273-283. Available from: https://www.researchgate.net/publication/257449475_Variability_of_rainfall_from_tropical_cyclones_in_the_eastern_USA_and_its_association_to_the_AMO_and_ENSO doi:10.1007/s00704-012-0722-y

Os - Miscellaneous operating system interfaces [Internet]. c2001-2019. Python Software Foundation. [cited 2019 Mar 20]. Available from: https://docs.python.org/3/library/os.html

Package osgeo [Internet]. 2018 Apr 20. gdal. [cited 2019 Mar 18]. Available from: https://gdal.org/python/

Pattis R. E. n.d.. ASCII Table. Carnegie Mellon University [Internet] [cited 2019 Apr 1]. Available from: https://www.cs.cmu.edu/~pattis/15-1XX/common/handouts/ascii.html

Pimpler E. 2017. Programming ArcGIS Pro with Python: Automate your ArcGis Pro geoprocessing tasks with Python. Boerne (TX): Geospatial Training Services.

Powell E. J., & Keim B. D. 2015. Trends in daily temperature and precipitation extremes for the southeastern United States: 1948–2012. Journal of Climate [Internet] [cited 2019 Mar 20]; 28(4): 1592-1612. Available from: https://journals.ametsoc.org/doi/full/10.1175/JCLI-D-14-00410.1 doi:10.1175/JCLI-D-14-00410.1

Quinn S., & Dutton J. 2019. GIS Programming and Software Development. Pennsylvania State University [Internet] [cited 2019 Mar 18]. Available from: https://www.e-education.psu.edu/geog485/node/91

Reading GRIB Files [Internet]. n.d.. National Oceanic and Atmospheric Administration. [cited 2018 Nov 10]. Available from: https://www.cpc.ncep.noaa.gov/products/wesley/reading_grib.html

Reaney S., & Wells P. 2016 Jan 20. Web based SCIMAP. Durham University [Internet] [cited 2018 Dec 10]. Available from: https://community.dur.ac.uk/sim.reaney/?p=500

Regional Climate Centers: Hubs of Local Expertise [Internet]. 2018 Aug 23. National Oceanic and Atmospheric Administration. [cited 2019 Mar 20]. Available from: https://www.ncei.noaa.gov/news/regional-climate-centers-hubs-local-expertise

Roberts J. J., Best B. D., Dunn D. C., Treml E. A., & Halpin P. N. 2010. Marine Geospatial Ecology Tools: An integrated framework for ecological geoprocessing with ArcGIS, Python, R, MATLAB, and C++. Environmental Modelling and Software [Internet] [cited 2019 Dec 10]; *25*(10): 1197–1207. Available from: http://doi.org/10.1016/j.envsoft.2010.03.029 doi:10.1016/j.envsoft.2010.03.029

Rossum G. V., & Drake F. L. 2003. Python language reference manual. Bristol: Network Theory.

Sawale G. J., & Gupta S. R. 2013. Use of Artificial Neural Network in Data Mining For Weather Forecasting. International Journal of Computer Science and Applications [Internet] [cited 2019 Apr 20]; *6*(2): 383–387. Available from: https://pdfs.semanticscholar.org/d3b2/928d0ef70808329a6b3e8dd0b33bbb9674ba.pdf

Severe Storm Reports Tool [Internet]. n.d.. North Carolina Climate Office. [cited 2019 Mar 17]. Available from: http://climate.ncsu.edu/climate/storm_reports

Shutil - High-level file operations [Internet]. c2001-2019. Python Software Foundation. [cited 2019 Mar 20]. Available from: https://docs.python.org/3/library/shutil.html

Snow A. D., Christensen S. D., Swain N. R., Nelson E. J., Ames D. P., Jones N. L., … Zsoter E. 2016. A High-Resolution National-Scale Hydrologic Forecast System from a Global Ensemble Land Surface Model. Journal of the American Water Resources Association [Internet] [cited 2019 Apr 20]; *52*(4): 950–964. Available from: http://doi.org/10.1111/1752-1688.12434 doi:10.1111/1752-1688.12434

State Members [Internet]. n.d.. American Association of State Climatologists. [cited 2019 Mar 10]. Available from: http://stateclimate.org/

Stoner A. M. K., Hayhoe K., & Wuebbles D. J. 2009. Assessing general circulation model simulations of atmospheric teleconnection patterns. Journal of Climate [Internet] [cited 2019 Feb 10]; 22(16): 4348-4372. Available from: https://doi.org/10.1175/2009JCLI2577.1 doi:10.1175/2009JCLI2577.1

Storm Prediction Center Frequently Asked Questions (FAQ) [Internet]. 2019. National Weather Service. [cited 2018 Dec 10]. Available from: https://www.spc.noaa.gov/faq/#6.1

sys - System-specific parameters and functions [Internet]. c2001-2019. Python Software Foundation. [cited 2019 Mar 20]. Available from: https://docs.python.org/3.4/library/sys.html

Sznaider R. J., Chenevert D. P., Hugg R. L., Reece C. F., & Block J. H. 2004. GIS-based automated weather alert notification system. [Internet]. [cited 2019 Apr 20]. Washington (DC): U.S. U.S. Patent and Trademark Office. Patent No. US6753784B1. Available from: https://patents.google.com/patent/US6753784B1/en

The Enhanced Fujita Scale (EF Scale) [Internet]. 2014. National Weather Service. [cited 2018 Dec 10]. Available from: https://www.spc.noaa.gov/efscale/

Time - Time access and conversions [Internet]. c2001-2019. Python Software Foundation. [cited 2019 Mar 20]. Available from: https://docs.python.org/3/library/time.html

The Python Programming Language [Internet]. c2019. Runestone Interactive. [cited 2019 Apr 1]. Available from: https://runestone.academy/runestone/books/published/thinkcspy/GeneralIntro/ThePythonProgrammingLanguage.html

Waclena K. n.d.. Introduction to Python: Class 7: Python on the Web. University of Chicago. [cited 2019 Mar 18]. Available from: https://www2.lib.uchicago.edu/keith/courses/python/class/7/

Warmerdam F. 2019 July 5. GDAL/OGR in Python. Python Software Foundation [Internet] [cited 2018 Nov 20]. Available from: https://pypi.org/project/GDAL/

What is Python? Executive Summary [Internet]. c2001-2019. Python Software Foundation. [cited 2019 Mar 20]. Available from: https://www.python.org/doc/essays/blurb/

What's the difference between Scripting and Programming Languages? [Internet]. n.d.. Geeks for geeks. [cited 2019 Jan 21]. Available from: https://www.geeksforgeeks.org/whats-the-difference-between-scripting-and-programming-languages/

Whitaker J. 2014 Dec 29. Module pygrib. Github [Internet] [cited 2018 Nov 15]. Available from: https://jswhit.github.io/pygrib/docs/

Working with dynamic text [Internet]. c2016. ESRI. [cited 2019 Apr 25]. Available from: http://desktop.arcgis.com/en/arcmap/10.3/map/page-layouts/working-with-dynamic-text.htm

APPENDIX

Appendix A: Tennessee State Climate Office Mission Statement

Proposed Mission Statement:

Tennessee has a diverse climate attributable to its size and landscape and is often geographically and culturally defined by three "grand divisions:" East Tennessee (Appalachian Mountains, Cumberland Mountains, and the ridge-and-valley region), Middle Tennessee (rolling hills), and West Tennessee (flat topography part of the Gulf Coastal Plain). As the climate of the state varies greatly from west to east, it has wide-ranging impacts on many parts of our economy daily. The state climate office for Tennessee was originally established under the direction of the Tennessee Valley Authority (TVA), however it ceased operations in 2006. Over ten years later in 2017, we are attempting to re-establish the office at East Tennessee State University as the Tennessee Climate Office (TCO). Climate influences various sectors of our state economy including agriculture, transportation, tourism, recreation, and the environment. The mission of the TCO is to provide climate-related services to state, local and federal agencies, businesses, and the citizens of Tennessee.

The TCO will initially be housed in the Geosciences complex within Ross Hall on the main campus of East Tennessee State University, where university researchers, government agencies, and private industries come together and create a unique environment for interaction and advancement.

It is expected that the TCO will be actively involved in research that enhances its capabilities to provide public service. Examples include analyses of climate hazards in Tennessee, seasonal weather forecast dissemination, drought monitoring, El Niño/La Niña effects on Tennessee weather and climate, agricultural and water resource management, air quality and environmental management, and natural hazard risk assessment and mitigation. The TCO derives its expertise from the Department of Geosciences at East Tennessee State University, the University of Tennessee Institute of Agriculture, the Department of Agriculture, Geosciences, and Natural Resources at the University of Tennessee-Martin, and the Department of Civil and Environmental Engineering at Vanderbilt University. Undergraduate and graduate students from different universities will participate in the research and extension activities at the TCO every year.

Partnerships with the following agencies are anticipated/expected: *TVA, Tennessee Wildlife Resources Agency, Tennessee Emergency Management Agency, Tennessee Department of Transportation, Tennessee Department of Environment and Conservation, Oak Ridge National Laboratory, National Weather Service (all four locations serving TN), Southern Regional Climate Center, Southeast Regional Climate Center, the National Centers for Environmental Information, Association of American State Climatologists,* and the *Federal Emergency Management Agency*. This is a tentative list and is not intended to be comprehensive at this point in time. Interaction with these organizations enhance our outreach activities.

Mission Areas:

1) Extension

Provide the most accurate climate information to the citizens of Tennessee.

Assist Tennessee state agencies in climate-environment interaction issues and related applications.

Establish, operate, and maintain an extensive meteorological network across Tennessee and archive and disseminate data to the public in a timely fashion.

Assist other extension scientists by integrating climate information into applications such as agricultural and environmental models.

Increase public awareness of variations in Tennessee climate and environment to aid in long-range planning.

2) Research

Study Tennessee's climate and its interaction with the environment.

Investigate the effects of climatic variations on agriculture, air pollution, and natural resources and develop forecasts that assist in resource management.

3) Education

Interact with K-12 schools, community college and university teachers and students, and with other community organizations on different aspects of Tennessee climate and environment.

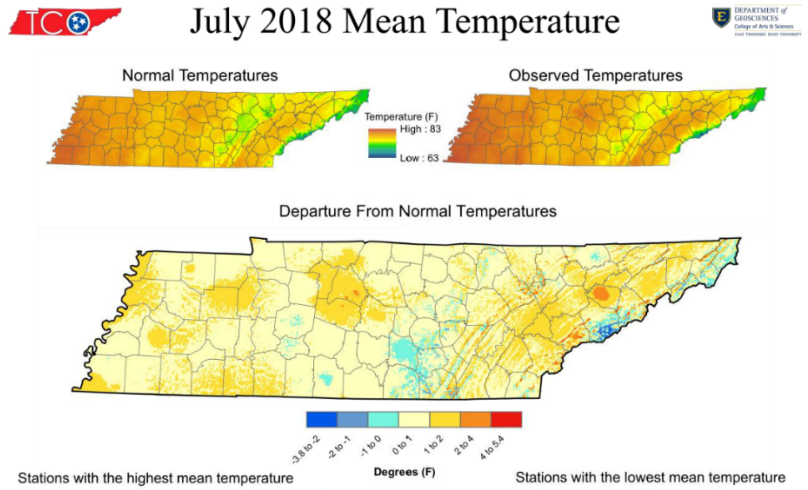
July 2018 Tennessee Climate Summary

Prepared by William Tollefson and Dr. Andrew Joyner

Tennessee Climate Office * East Tennessee State University

Monthly Temperature Summary:

For the month of July mean temperatures for most of the state were ±1°F of normal temperature. In most of climate division 4 (West TN) temperatures were slightly warmer than normal (+2-3°F), and areas around Nashville, and in the Tennessee Valley in East TN were also about 2°F above normal. Some areas in southern middle Tennessee were slightly below normal temperature (-1°F). The monthly mean temperature trend however does not tell the whole story - the first half of the month was quite warm for most areas of the state; with temperatures running up to 4 degrees above normal for the first week of the month. These conditions were part of a heat wave that hit the eastern US, which peaked around the 4th of July, and resulted with heat advisories or special weather statements for heat in all but two Tennessee counties.



Stations with the highest mean temperature

Name	Station Type	Mean Temperature (F)
GERMANTOWN 4SE	COOP	84
MEMPHIS INTERNATIONAL AP	WBAN	83
BROWNSVILLE	COOP	82.2
SHILOH NMP TENNESSEE	RAWS	81.8
NASHVILLE INTL AP	WBAN	81.7

Map Data From: PRISM Climate Group, Oregon State University
1980-2010 Normals Used
Station Data retrieved from SC ACIS2

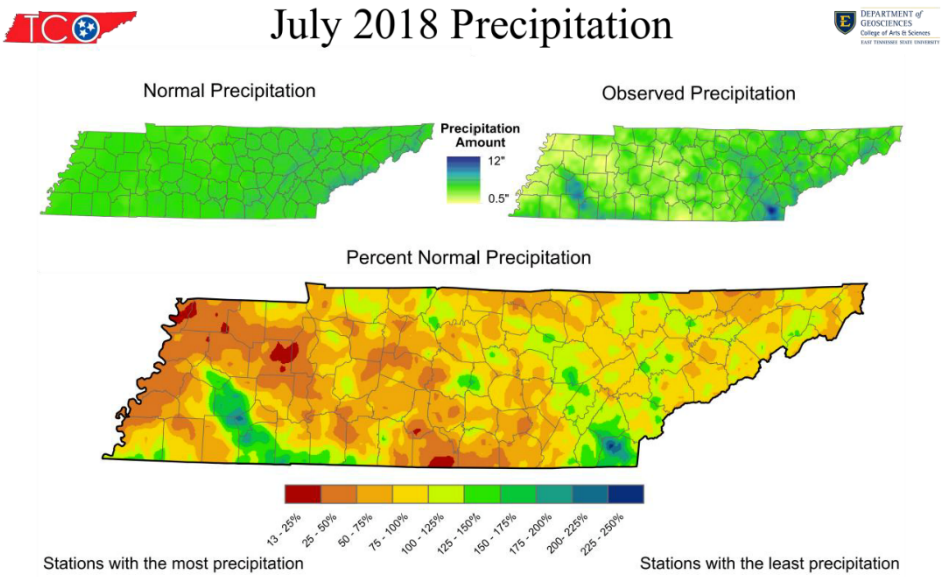
Stations with the lowest mean temperature

Name	Station Type	Mean Temperature (F)
MT LECONTE	COOP	58.8
NEWFOUND GAP	COOP	66.5
ROAN MOUNTAIN 3SW	COOP	69.7
CHEROKEE TENNESSEE	RAWS	71.1
TOWNSEND 5S	COOP	72.1

Monthly Precipitation Summary:

For the month of July most of the state recorded below normal precipitation, with parts of West TN (Lauderdale, Tipton, and Lake Counties along the Mississippi River; and Benton, Decatur, Perry, and Carroll counties along and west of the Tennessee River) running more than 3" below normal. Areas in southeast Tennessee and parts of the Cumberland Plateau; as well as Unicoi, Carter, and Washington counties in northeast TN; and Maddison, Chester, and McNary counties in southwest Tennessee recorded 0 to 3" above normal precipitation this month.

The month started with heavy rains along the Cumberland Plateau, and central Middle TN. For the middle of the month there was widespread, but spotty rainfall from scattered storms and storm complexes which produced a range of observed rainfall from ¼ of an inch to 8 inches across the state (it was rainfall in this period that resulted in a monthly surplus in Maddison, Chester, and McNary counties). Then at the end of the month, heavy rains soaked East TN, while many areas in West Tennessee recorded no measureable precipitation. The drying trend in the second half of the month lead to the introduction of D0 drought conditions in West Tennessee, along the Mississippi River.



Stations with the most precipitation

Name	Station Type	Total Precipitation (in)
BENTON 2.3 ESE	CoCoRaHS	12.75
MEDON 3.5 SSW	CoCoRaHS	10.38
JACKSON EXP STA	COOP	8.77
CROSSVILLE 6.9 S	CoCoRaHS	8.38
FARRAGUT 1.0 WNW	CoCoRaHS	8.26

Map Data From: PRISM Climate Group, Oregon State University, 1980-2010 Normals Used
Station Data retrieved from SC ACIS2

Stations with the least precipitation

Name	Station Type	Total Precipitation (in)
CLARKSVILLE OUTLAW AP	WBAN	0.27
MURFREESBORO 3.2 WSW	CoCoRaHS	0.43
WATERTOWN PUB SAFETY	COOP	0.46
HUNTINGDON WATER PLANT	COOP	0.75
CLARKSVILLE NO.2	COOP	0.83

Top Tenn (warmest, coldest, wettest, driest stations of the month):

Hottest Stations (highest max temp):

Name	Station Type	Max Temperature (F)
FRANKLIN SEWAGE PLANT	COOP	100
GERMANTOWN 4SE	COOP	100
LEBANON	COOP	99
NASHVILLE BERRY FIELD	COOP	99
SHILOH NMP TENNESSEE	RAWS	99
NASHVILLE INTL AP	WBAN	99
MURFREESBORO 5 N	COOP	98
CLARKSVILLE NO.2	COOP	98
CLARKSVILLE WWTP	COOP	98
CAMDEN	COOP	98
THOMPSON STATION	COOP	98
WHITE HOUSE	COOP	98
CAMDEN TOWER TENNESSEE	RAWS	98
MEMPHIS WFO	WBAN	98

Coldest Stations (lowest min temp):

Name	Station Type	Lowest Temperature (F)
MT LECONTE	COOP	44
ROAN MOUNTAIN 3SW	COOP	52
NEWFOUND GAP	COOP	53
CROSSVILLE 7 NW	WBAN	55
COALMONT	COOP	55
CHEROKEE TENNESSEE	RAWS	56
JAMESTOWN	COOP	56
TAZEWELL	COOP	56
CROSSVILLE MEMORIAL AP	WBAN	57
BRISTOL AP	WBAN	57
BLEDSOE SF TENNESSEE	RAWS	57
WOODBURY 1 WNW	COOP	57
FRANKLIN SEWAGE PLANT	COOP	57
NORRIS	COOP	57
ALLARDT	COOP	57
ONEIDA	COOP	57
NEWCOMB	COOP	57
WATERTOWN PUB SAFETY	COOP	57

Warmest Stations (highest mean temps):

Name	Station Type	Mean Temperature (F)
GERMANTOWN 4SE	COOP	84
MEMPHIS INTERNATIONAL AP	WBAN	83
BROWNSVILLE	COOP	82.2
SHILOH NMP TENNESSEE	RAWS	81.8
NASHVILLE INTL AP	WBAN	81.7
MEMPHIS WFO	WBAN	81.6
NASHVILLE BERRY FIELD	COOP	81.3
CAMDEN TOWER TENNESSEE	RAWS	81.1
JACKSON EXP STA	COOP	81.1
SELMER	COOP	80.9
CAMDEN	COOP	80.9

Coolest Stations (lowest mean temps):

Name	Station Type	Mean Temperature (F)
MT LECONTE	COOP	58.8
NEWFOUND GAP	COOP	66.5
ROAN MOUNTAIN 3SW	COOP	69.7
CHEROKEE TENNESSEE	RAWS	71.1
TOWNSEND 5S	COOP	72.1
GATLINBURG 2 SW	COOP	72.6
CROSSVILLE 7 NW	WBAN	73
COALMONT	COOP	73.5
ALLARDT	COOP	74.1
TAZEWELL	COOP	74.1

Wettest Stations (highest rainfall totals):

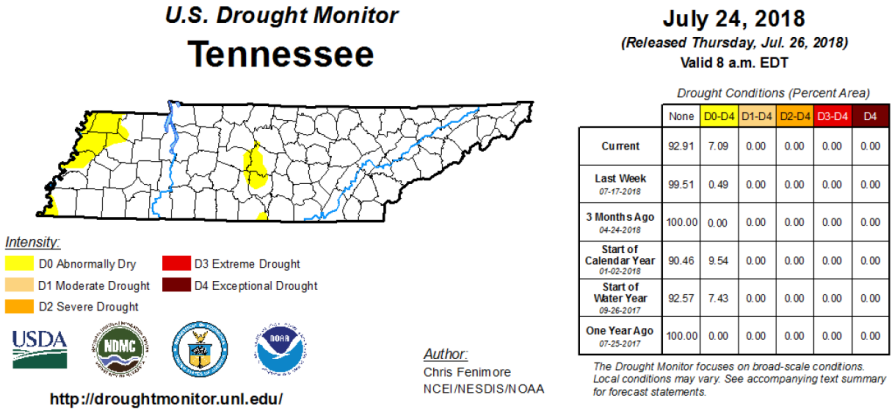
Name	Station Type	Total Precipitation (in)
BENTON 2.3 ESE	CoCoRaHS	12.75
MEDON 3.5 SSW	CoCoRaHS	10.38
JACKSON EXP STA	COOP	8.77
CROSSVILLE 6.9 S	CoCoRaHS	8.38
FARRAGUT 1.0 WNW	CoCoRaHS	8.26
HENDERSON 4 W	COOP	7.86
JACKSON MCKELLAR- SIPES AP	WBAN	7.81
HARRIMAN 4.5 SW	CoCoRaHS	7.73
GADSDEN 4.9 SSE	CoCoRaHS	7.64
NEWFOUND GAP	COOP	7.41

Driest Stations (lowest rainfall totals):

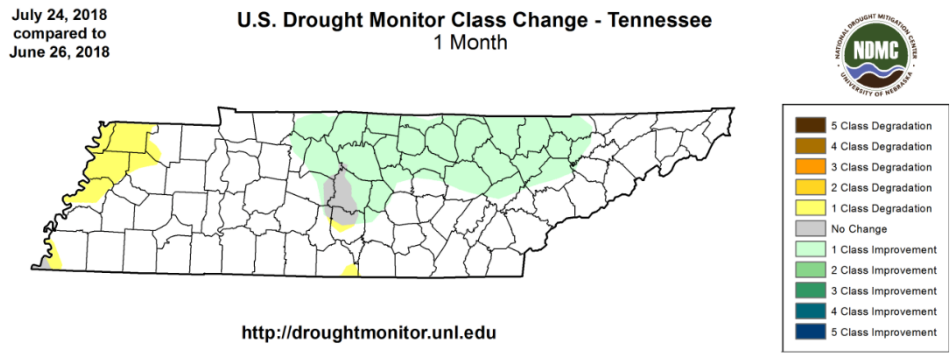
Name	Station Type	Total Precipitation (in)
CLARKSVILLE OUTLAW AP	WBAN	0.27
MURFREESBORO 3.2 WSW	CoCoRaHS	0.43
WATERTOWN PUB SAFETY	COOP	0.46
HUNTINGDON WATER PLANT	COOP	0.75
CLARKSVILLE NO.2	COOP	0.83
LEWISBURG 6.2 SSE	CoCoRaHS	0.97
UNION CITY	COOP	1.21
SHELBY BOTTOMS NATURE CENTER	COOP	1.24
LEWISBURG EXP STA	COOP	1.32
CAMDEN	COOP	1.4
KINGSTON SPRINGS 0.3 ENE	CoCoRaHS	1.4

Drought Monitor:

As of the last Drought Monitor release of the month, D0 drought conditions were introduced to portions of north-west Tennessee, and portions of Middle Tennessee, covering 7.09% of the state.

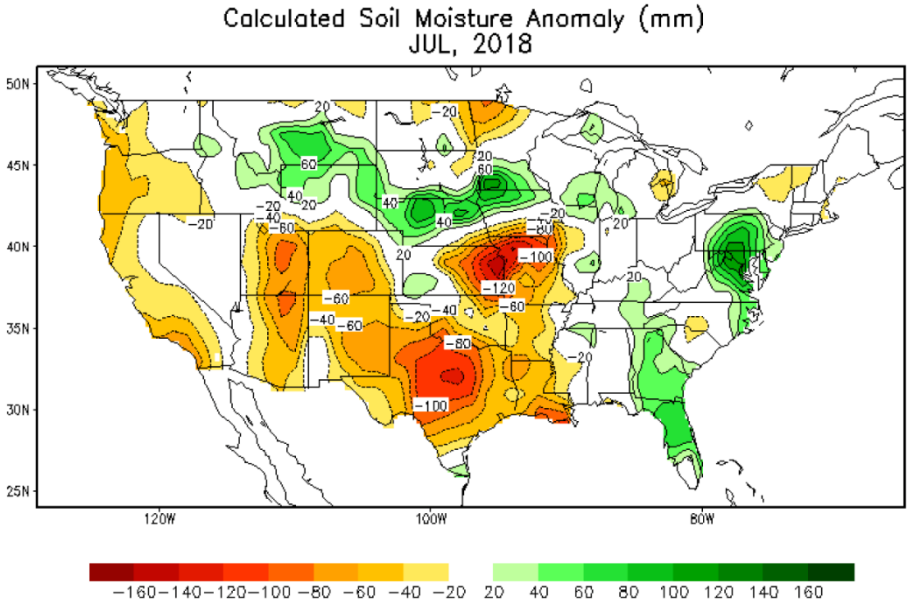


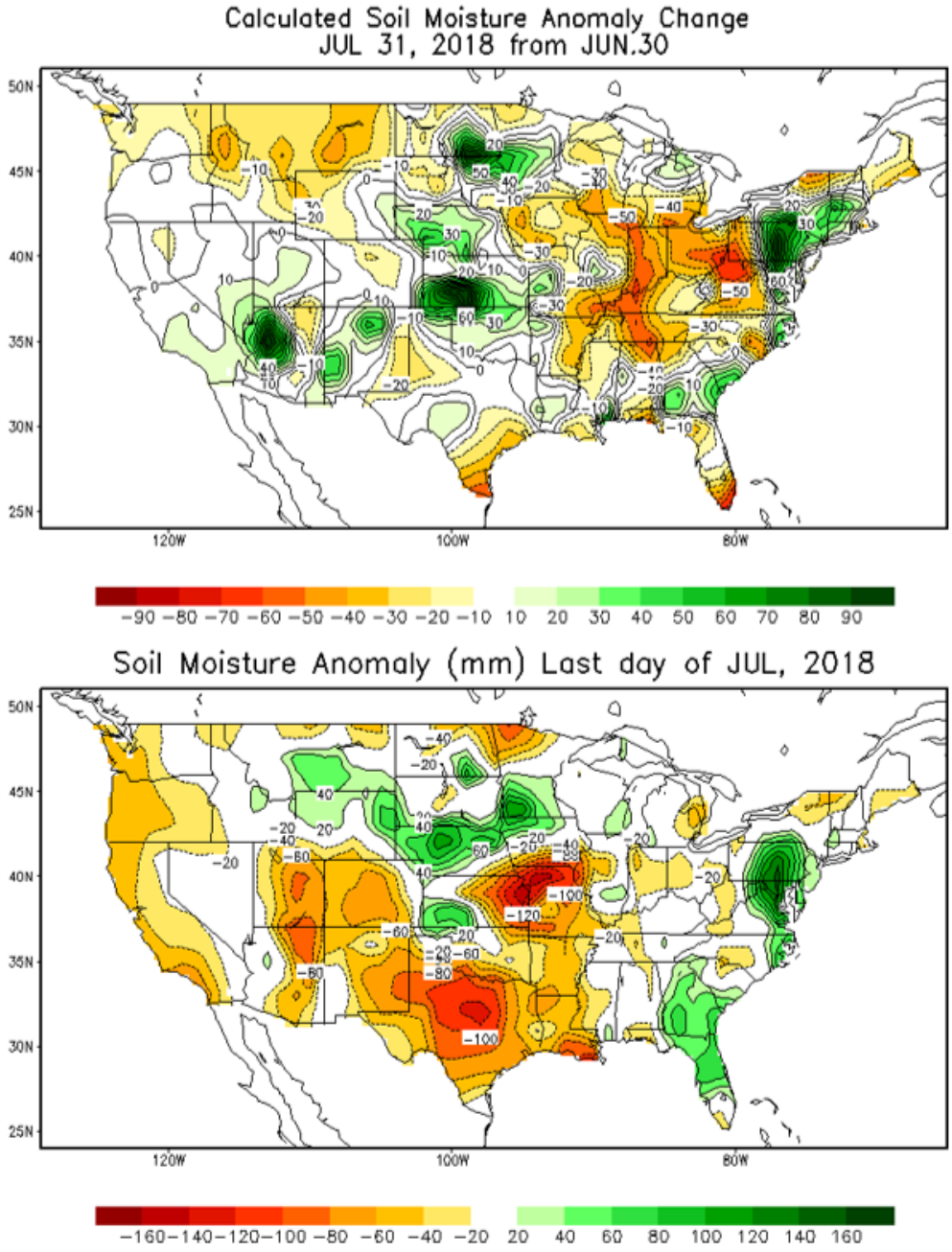
Looking back at changes from the end of June, a large area of D0 conditions in the upper Cumberland Plateau and northern Middle Tennessee were cleared by heavy rains at the beginning of the month. However drying trends at the end of the month will likely see expansion of D0 conditions in the first Drought Monitor release of August.



Soil Moisture:

The monthly mean soil moisture anomaly shows most of Tennessee around or slightly above normal. However, there was a major drying trend through July, with all areas of the state recording decreases in soil moisture (ranging from 10 to 50mm). By the end of the month large areas in Middle Tennessee were recording negative anomalies (below normal soil moisture).




Calculated Soil Moisture Anomaly Change
JUL 31, 2018 from JUN.30
50N
45N
40N
35N
30N
25N
120W
100W
80W
-90 -80 -70 -60 -50 -40 -30 -20 -10 10 20 30 40 50 60 70 80 90
Soil Moisture Anomaly (mm) Last day of JUL, 2018
50N
45N
40N
35N
30N
25N
120W
100W
80W
-160 -140 -120 -100 -80 -60 -40 -20 20 40 60 80 100 120 140 160

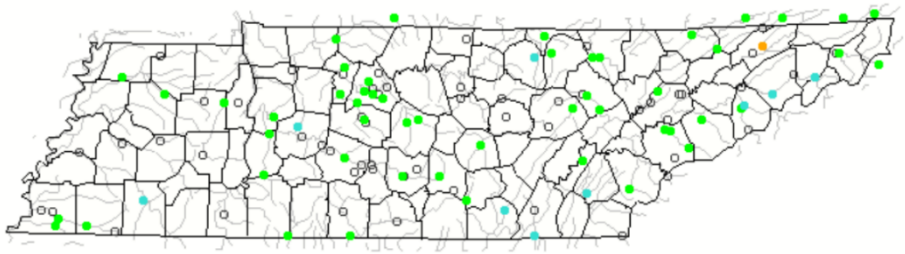
Streamflow:

Monthly average streamflow was normal for almost all streams in the state, with one stream gauge reporting below normal flow and a few reporting above normal flow. However, the drying trend during the month led to more gauges reporting lower than normal flow by the end of the month.

Map of 28-day average streamflow compared to historical streamflow for the day of the year (Tennessee)

Tennessee ▾ or Water-Resources Regions ▾

Tuesday, July 31, 2018



USGS

Search USGS streamgage

Choose a data retrieval option and select a location on the map

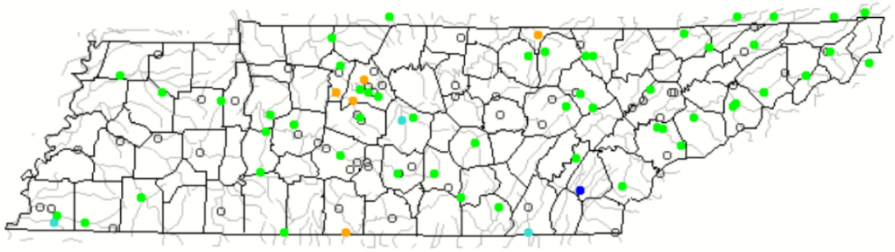
List of all stations **Single station** **Nearest stations**

Explanation - Percentile classes							
Low	<10 Much below normal	10-24 Below normal	25-75 Normal	76-90 Above normal	>90 Much above normal	High	Not-ranked

Map of daily streamflow compared to historical streamflow for the day of the year (Tennessee)

Tennessee ▾ or Water-Resources Regions ▾ All Days

Tuesday, July 31, 2018



USGS

Search USGS streamgage

Choose a data retrieval option and select a location on the map

List of all stations **Single station** **Nearest stations** **Peak flow**

Explanation - Percentile classes							
Low	<10 Much below normal	10-24 Below normal	25-75 Normal	76-90 Above normal	>90 Much above normal	High	Not-ranked

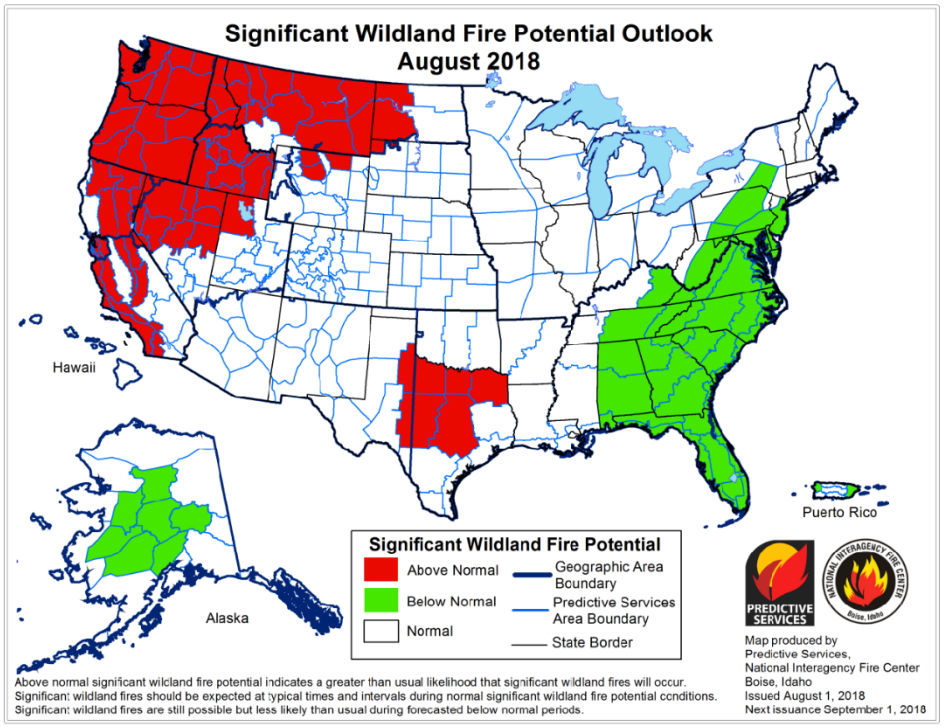
Miscellaneous:

Crop Conditions from USDA: For agricultural concerns in the state, July was a tale of the haves and the have-nots with rainfall. Many farmers were hindered from field work by consistent rain/stormy patterns, while many others started to report crop stress from lack of rain. Row crop farmers were busy spraying for pests, while forage producers worked on their second cutting of hay. Pasture conditions declined slightly as pastures in drier areas began to show signs of stress due to lack of moisture. Topsoil moisture rated 8 percent very short, 33 percent short, 53 percent adequate, and 6 percent surplus. Subsoil moisture rated 6 percent very short, 28 percent short, 65 percent adequate, and 1 percent surplus, at the end of the month.

Corn, cotton, soybean, and tobacco crops were all at or ahead of 5-year average progress at the end of the month. The cotton, corn, and soybean crops had the largest percentages of 'Excellent' and 'Good' conditions reported, while pastures had the largest percentage of 'Poor' conditions.

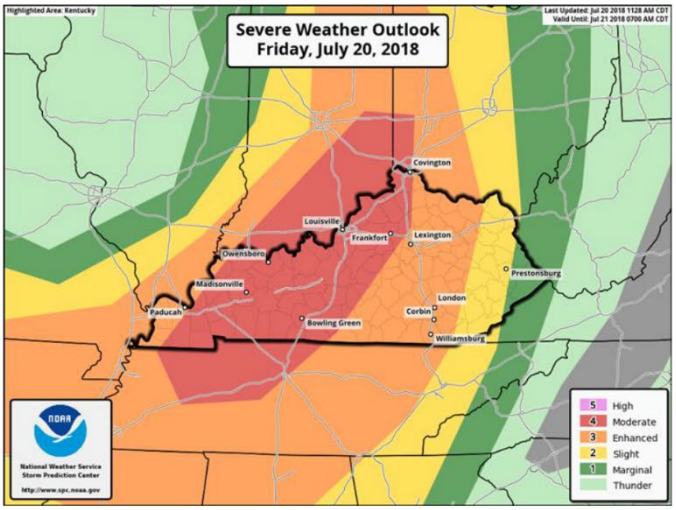
CROP PROGRESS					CONDITION					
Crop	This Week	Last Week	2017	5 Year Avg.	Item	Very Poor	Poor	Fair	Good	Excellent
	Percent					Percent				
Corn – Silking	96	95	98	96	Corn	3	5	18	56	18
Corn – Doughing	78	64	66	63	Tobacco	0	2	36	59	3
Cotton – Squaring	98	97	96	91	Cotton	1	1	8	62	28
Cotton – Setting Bolls	70	50	62	52	Pasture	1	14	28	46	11
Soybeans – Blooming	84	72	83	68	Soybeans	1	4	21	57	17
Soybeans – Setting Pods	55	38	50	40	Hay	1	5	35	48	11
Tobacco – Topped	50	30	43	39						

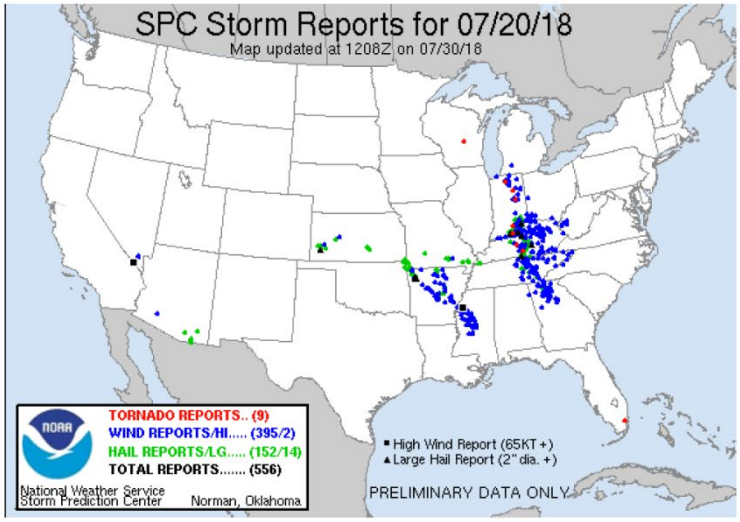
Fire Danger: Fire danger in East and Middle Tennessee is expected to be lower than normal during the month of August. However, due to drying conditions in the western part of the state they will have a normal risk.



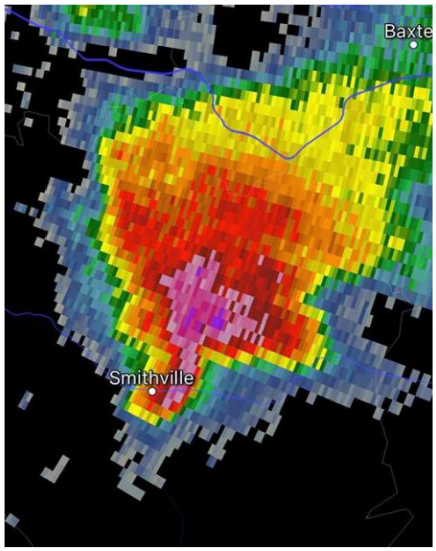
Storm of the month:

The storm of the month for July was the severe weather outbreak of July 20th. This event was forecasted by the NOAA Storm Prediction Center, who put out a moderate outlook for severe weather on the morning of the 20th. The storm system kicked off in northern Indiana in the early afternoon and then moved south-southeast producing a line of severe storms across southern Indiana, central/eastern Kentucky, and Middle and East Tennessee.





The line of severe storms moved into Tennessee after sunset on the 20th, with a large squall line in eastern Tennessee, and a few discrete supercells present on the western end of the line on the Cumberland Plateau.



Radar image of supercell near Smithville, TN (Source NWS Nashville Facebook)

Trees fallen on night watchman's truck at Warrior's Path State Park in Kingsport (image source: WJHL)

Several trees were blown down at Warrior's Path State Park in Kingsport, one of which fell on the campground night watchman's truck after he had driven around the park to warn campers to seek shelter before the storms hit. He was pinned inside the truck, and it took three people to get him out,

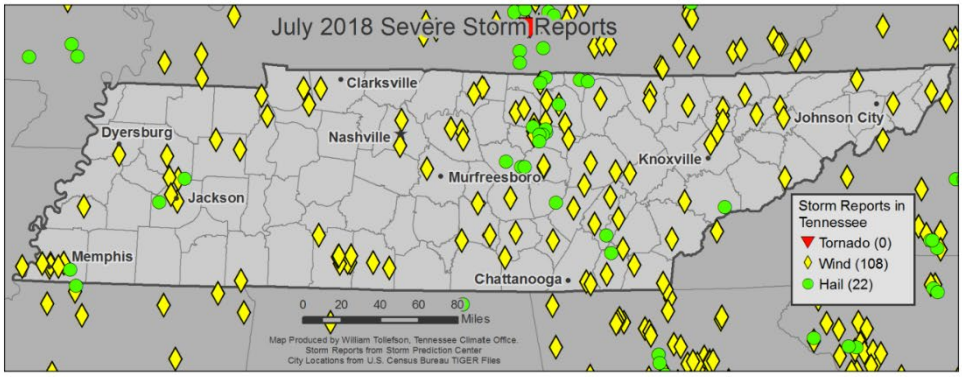
but luckily he was not injured. Another tree fell on a camper, and a woman inside was taken to the hospital, but was released the next morning with no serious injuries.

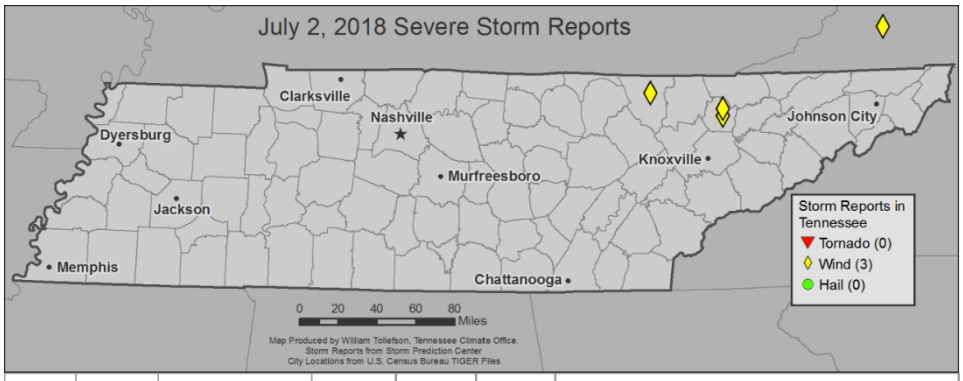
Fallen tree in Elizabethton with one fatality reported (Image Source: WJHL)

A tree fell on a house in Elizabethton, and unfortunately killed a 75 year-old resident of the home.

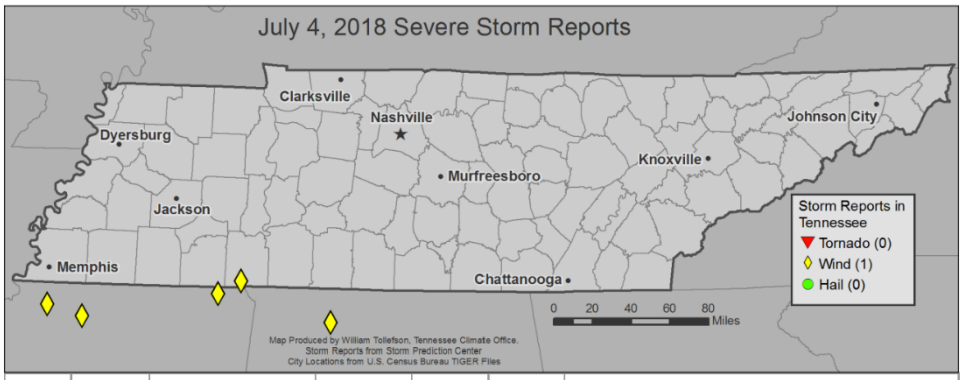
Storm reports:

Severe storms produced damage on 11 days during the month, with a total of 108 severe wind reports and 22 reports of hail 1 inch or larger.

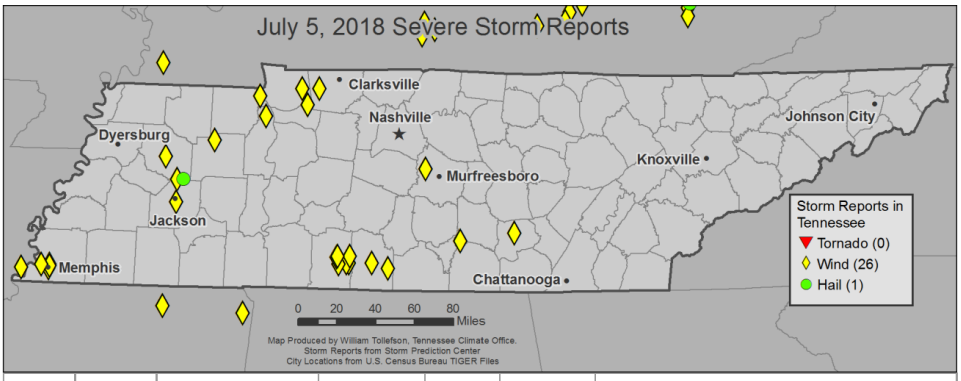




Time	Speed	Location	County	Lat	Lon	Comments
0:10	-	3 N MAYNARDVILLE	UNION	36.29	-83.81	HICKORY VALLEY ROAD...BARN WITH METAL ROOF AND 3 SIDED CARPORT DESTROYED. SOME SIDING REMOVED FROM HOME. SEVERAL TREES DOWN. (MRX)
1:40	-	SHARPS CHAPEL	UNION	36.34	-83.81	THREE TREES DOWN ACROSS THE COUNTY...AT LEAST ONE NEAR SHARPS CHAPEL (MRX)
1:40	-	3 SE ONEIDA	SCOTT	36.47	-84.48	ONE TREE DOWN ON PANIT ROCK RD. (MRX)



Time	Speed	Location	County	Lat	Lon	Comments
1:30	UNK	PICKWICK LANDING STATE	HARDIN	35.05	-88.23	TREE DOWN ON CAR... MINOR INJURIES REPORTED. TIME ESTIMATED BY RADAR. (MEG)

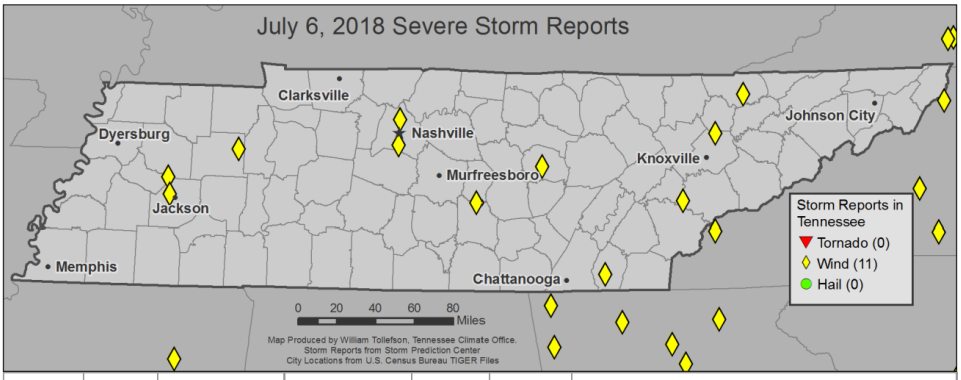


Time	Size	Location	County	Lat	Lon	Comments
2:25	1"	MEDINA	GIBSON	35.8	-88.77	QUARTER SIZED HAIL IN MEDINA. (MEG)

Time	Speed	Location	County	Lat	Lon	Comments
19:54	-	TULLAHOMA	COFFEE	35.37	-86.22	FACEBOOK PICTURE OF A TREE DOWN ON LYNN STREET IN TULLAHOMA. TIME ESTIMATED BASED ON RADAR. (OHX)
20:36	-	7 N ELKTON	GILES	35.16	-86.88	DELAYED REPORT FROM THURSDAY JULY 5TH. TREES DOWN BLOCKING THE NORTHBOUND LANES OF I-65 AT MILE MARKER 11 CLOSING THE INTERSTATE FOR A FEW HOURS. (OHX)
20:55	-	PULASKI	GILES	35.2	-87.03	REPORT OF TREES DOWN ON EAST JEFFERSON STREET. (OHX)
21:15	-	6 ESE LAWRENCEBURG	LAWRENCE	35.2	-87.24	REPORTS OF A ROOF BLOWN OFF SHED AT WESTSIDE ROAD IN EXTREME EASTERN LAWRENCE COUNTY. (OHX)
21:20	-	6 ESE LAWRENCEBURG	LAWRENCE	35.2	-87.24	REPORTS OF ROOF BLOW OFF SHEAD AT WESTSIDE ROAD IN EXTREME EASTERN LAWRENCE COUNTY. (OHX)
21:20	-	5 SE LAWRENCEBURG	LAWRENCE	35.2	-87.26	REPORT OF A TREE DOWN BLOCKING DURHAM ROAD. (OHX)
21:30	-	4 S LAWRENCEBURG	LAWRENCE	35.19	-87.33	REPORT OF TREE DOWN BLOCKING THE ROADWAY AT 116 DUNN LEOMA ROAD (OHX)
21:34	-	1 SW LAWRENCEBURG	LAWRENCE	35.24	-87.35	REPORT OF TREE DOWN BLOCKING SKYLINE DRIVE IN LAWRENCEBURG. (OHX)
21:34	-	6 E LAWRENCEBURG	LAWRENCE	35.25	-87.23	REPORT OF A TREE BLOCKING DURHAM ROAD. (OHX)
21:34	-	LAWRENCEBURG	LAWRENCE	35.25	-87.34	REPORTS OF TREES BLOCKING LEOMA ROAD NEAR LEOMA AND

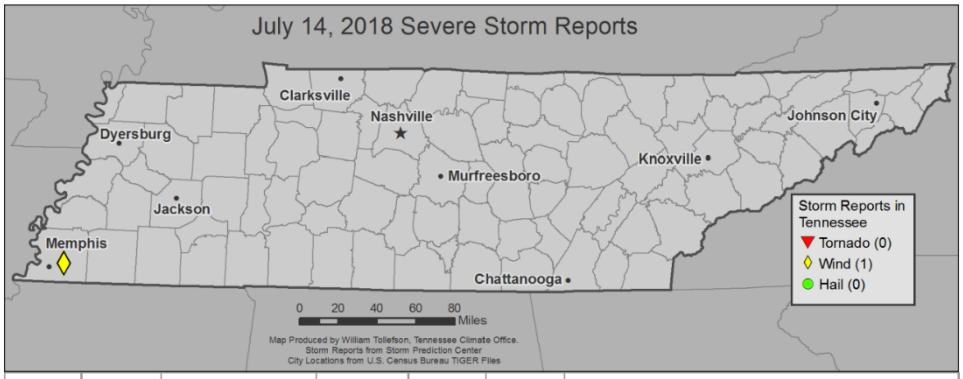
						SKYLINE DRIVE IN LAWRENCEBURG. (OHX)
22:09	-	9 WSW PARK LANE	MONTGOMERY	36.5	-87.53	REPORT OF TREES AND TREE LIMBS DOWN ABOUT 3 MILES SOUTH OF FORT CAMPBELL. ONE TREE FELL ONTO A TRUCK IN THE SPRING CREEK SUBDIVISION. (OHX)
22:43	-	INDIAN MOUND	STEWART	36.5	-87.69	TREES DOWN ON HIGHWAY 46 (OHX)
22:44	-	CUMBERLAND CITY	STEWART	36.38	-87.64	REPORT OF MULTIPLE TREES DOWN IN CUMBERLAND CITY WITH ONE TREE FALLING ON A HOME. (OHX)
22:49	-	CUMBERLAND CITY	STEWART	36.38	-87.64	TSPOTTER REPORT OF TREES AND POWER LINES DOWN ON THE 1500 BLOCK OF CUMBERLAND CITY ROAD IN STEWART COUNTY. (OHX)
23:20	-	5 NE BIG SANDY	BENTON	36.29	-88.02	LARGE TREE LIMBS DOWN ACROSS NORTHERN PARTS OF BENTON COUNTY. TIME ESTIMATED FROM RADAR. (MEG)
0:07	-	PARIS LANDING STATE PARK	HENRY	36.44	-88.08	THREE LARGE OAK TREES DOWN BEHIND THE POOL AT PARIS LANDING. PENNY SIZED HAIL FELL AS WELL. (MEG)
0:53	-	5 SSW SMYRNA	RUTHERFORD	35.91	-86.54	FACEBOOK PHOTO SHOWED TREE SNAPPED ON LONG SHADOW COURT (OHX)
1:15	-	3 SSE MCKENZIE	CARROLL	36.1	-88.49	TREE DOWN ON POWERLINES NEAR INTERSECTION OF HIGHWAY 22 AND HICO ROAD. (MEG)
2:10	-	TRENTON	GIBSON	35.97	-88.94	MULTIPLE TREES DOWN AND POWER LINES...ESPECIALLY HIGHWAY 45. (MEG)
2:19	-	1 E MIDTOWN MEMPHIS	SHELBY	35.13	-89.97	LARGE TREE DOWN BLOCKING WALNUT GROVE AT GOODLETT ROAD. (MEG)
2:22	-	2 S MIDTOWN MEMPHIS	SHELBY	35.1	-89.98	UPROOTED TREES AND POWER LINES DOWN NEAR THE PINK PALACE MUSEUM. (MEG)
2:24	-	7 S FRAYSER	SHELBY	35.12	-89.97	CONSIDERABLE DAMAGE TO A LARGE GAS STATION SIGN AT THE EXXON STATION ALONG POPLAR AVENUE NEAR OAK COURT MALL. TIME ESTIMATED FROM RADAR. (MEG)
2:25	-	3 W MEDINA	GIBSON	35.8	-88.83	SIGNIFICANT DAMAGE TO A MOBILE HOME ROOF OFF OF HIGHWAY 186. (MEG)
2:30	-	ALTAMONT	GRUNDY	35.43	-85.73	DELAYED REPORT FROM THURSDAY JULY 5TH. NUMEROUS TREES BLOWN DOWN IN AND AROUND ALTAMONT WITH SOME TREES

						FALLING ONTO HOMES AND CARS. SEVERAL ROADS BLOCKED BY FALLEN TRE (OHX)
2:35	-	DOWNTOWN MEMPHIS	SHELBY	35.13	-90.05	SEVERAL TREES DOWN ACROSS THE COUNTY. (MEG)
2:35	-	E JACKSON	MADISON	35.63	-88.83	SEVERAL TREES DOWN ALONG HIGHWAY 412 EAST OF JACKSON. (MEG)

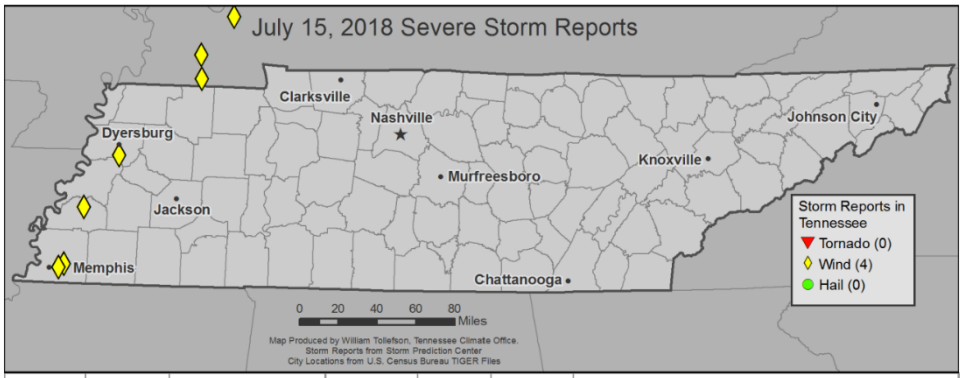


Time	Speed	Location	County	Lat	Lon	Comments
16:47	-	OAK HILL	DAVIDSON	36.08	-86.79	TSPOTTER REPORT OF TREE DOWN IN OAK HILL. TIME ESTIMATED BASED ON RADAR. (OHX)
17:37	-	3 E WHITES CREEK	DAVIDSON	36.27	-86.78	SEVERAL TREES DOWN IN THE BELLSHIRE AREA INCLUDING ONE ON BRICK CHURCH PIKE. (OHX)
17:55	-	8 SE BRADYVILLE	CANNON	35.65	-86.07	TSPOTTER REPORT OF TREES DOWN ON HIGHWAY 53 ONE MILE NORTH OF THE COFFEE COUNTY LINE. TIME ESTIMATED BASED ON RADAR. (OHX)
18:10	-	HOLLOW ROCK	CARROLL	36.03	-88.27	LARGE LIMBS RIPPED FROM TREES AND ONE TREE BENT OVER PERMANENTLY. (MEG)
18:15	-	1 S SPARTA	WHITE	35.92	-85.47	FACEBOOK PHOTOS SHOWED A CEDAR TREE WAS SNAPPED AND LARGE TREE BRANCHES WERE BLOWN DOWN ON KNOWLES STREET IN SPARTA (OHX)
18:17	-	1 S HUMBOLDT	GIBSON	35.81	-88.91	TREE BLOCKING WINDY CITY RD. (MEG)
18:45	-	5 NW JACKSON	MADISON	35.68	-88.89	TREE DAMAGE AND CARPORT DAMAGE NEAR INTERSECTION OF PLEASANT PLAINS RD AND ASHPORT RD. (MEG)
20:37	-	GREENBACK	LOUDON	35.65	-84.18	NUMEROUS TREES DOWN ACROSS SOUTHEAST LOUDON COUNTY NEAR GREENBACK. TIME ESTIMATED BY RADAR. (MRX)

20:40	-	5 SSW CLEVELAND	BRADLEY	35.11	-84.9	WIND DAMAGE TO BARN/SHED ROOF OFF OF BLUE SPRINGS RD. (MRX)
22:30	-	2 SW TAZEWELL	CLAIBORNE	36.44	-83.6	SEVERAL TREES DOWN NEAR 1ST AVENUE IN NEW TAZEWELL. (MRX)
23:37	-	8 NW MASCOT	KNOX	36.15	-83.87	COUPLE OF TREES DOWN. (MRX)

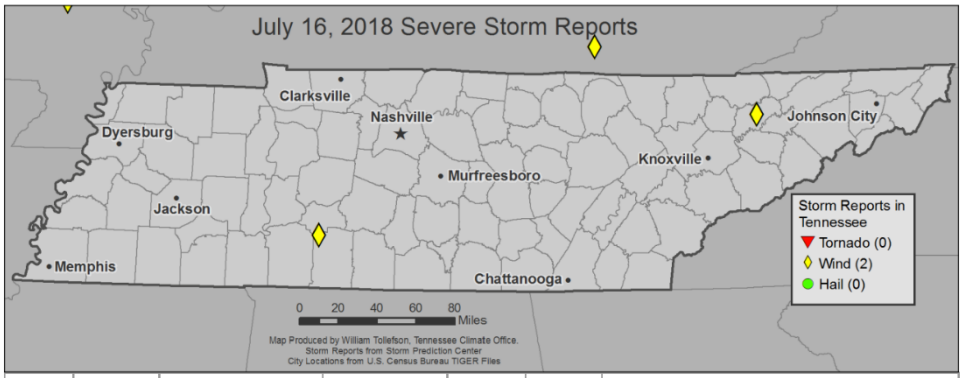


Time	Speed	Location	County	Lat	Lon	Comments
23:40	-	5 NW GERMANTOWN	SHELBY	35.14	-89.85	A FEW MEDIUM SIZE TREES DOWN ACROSS SHELBY FARMS PARK. TIME ESTIMATED OFF RADAR. (MEG)

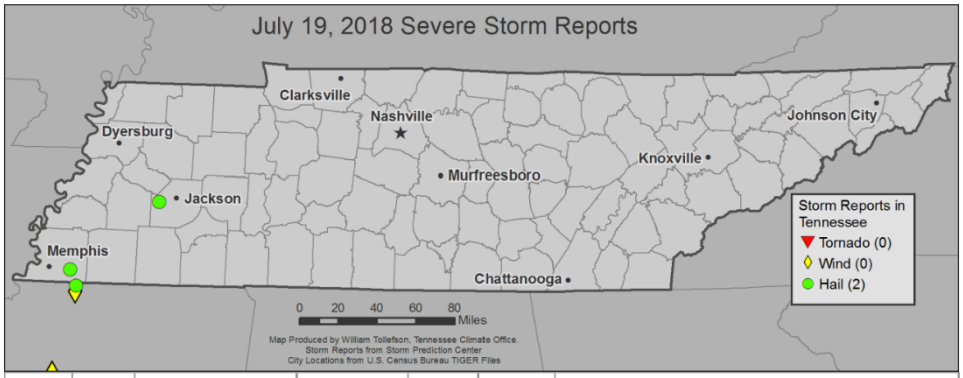


Time	Speed	Location	County	Lat	Lon	Comments
20:45	-	5 S DYERSBURG	DYER	35.97	-89.38	SEVERAL TREE LIMBS BLOCKING THE ROAD NEAR THE FOWLKES COMMUNITY. (MEG)
23:39	60	2 W COVINGTON	TIPTON	35.57	-89.69	(MEG)
23:40	-	5 NW GERMANTOWN	SHELBY	35.14	-89.85	A FEW MEDIUM SIZE TREES DOWN ACROSS SHELBY FARMS PARK. TIME ESTIMATED OFF RADAR. (MEG)

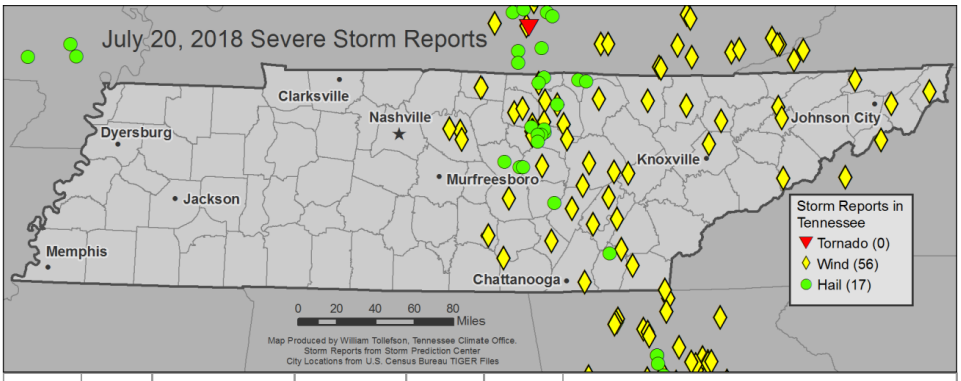
0:40	-	3 NNW SOUTHEAST MEMPHIS	SHELBY	35.11	-89.9	LARGE TREE DOWN ACROSS SAINT NICK DR. (MEG)



Time	Speed	Location	County	Lat	Lon	Comments
12:09	-	15 NW LAWRENCEBURG	LAWRENCE	35.4	-87.53	A LARGE TREE FELL ON A FENCE AND SEVERAL BRANCHES WERE DOWN ALONG THE NATCHEZ TRACE. (OHX)
21:23	-	2 NE RUTLEDGE	GRAINGER	36.29	-83.5	COUPLE OF TREES DOWN NEAR RUTLEDGE AND TO THE NORTHEAST BETWEEN RUTLEDGE AND BEAN STATION. (MRX)



Time	Size	Location	County	Lat	Lon	Comments
1:22	1"	N GERMANTOWN	SHELBY	35.09	-89.79	QUARTER SIZED HAIL AROUND HOUSTON LEVEE ROAD AND WOLF RIVER. (MEG)
3:00	1"	1 W HUNTERSVILLE	MADISON	35.62	-89	QUARTER SIZED HAIL ON PROVIDENCE ROAD SOUTH OF I-40. (MEG)



Time	Size	Location	County	Lat	Lon	Comments
0:30	1"	4 NE CELINA	CLAY	36.59	-85.45	ESTIMATED 1 INCH HAIL REPORTED IN PEA RIDGE (OHX)
0:32	1"	BYRDSTOWN	PICKETT	36.57	-85.13	FACEBOOK PHOTOS SHOWED DIME TO QUARTER SIZE HAIL IN BYRDSTOWN (OHX)
0:50	1"	4 ESE BYRDSTOWN	PICKETT	36.56	-85.06	FACEBOOK PHOTOS SHOWED QUARTER SIZE HAIL IN THE CEDAR GROVE AREA (OHX)
1:07	1"	CELINA	CLAY	36.55	-85.5	QUARTER SIZE HAIL REPORTED IN CELINA. (OHX)
1:20	1"	2 SSW CUMMINS FALLS STA	PUTNAM	36.22	-85.57	QUARTER SIZE HAIL REPORTED NEAR CUMBY ROAD AND GAINESBORO GRADE. (OHX)
1:24	1.75"	1 SSW ALGOOD	PUTNAM	36.18	-85.45	DELAYED REPORT FROM FRIDAY JULY 20. GOLFBALL SIZE HAIL REPORTED NEAR ALGOOD ELEMENTARY SCHOOL. (OHX)
1:25	1.25"	ALGOOD	PUTNAM	36.2	-85.45	(OHX)
1:31	1.25"	COOKEVILLE	PUTNAM	36.16	-85.51	TSPOTTER REPORT OF HAIL IN COOKEVILLE. LARGER THAN A QUARTER IN COMPARISON. (OHX)
1:34	1"	LIVINGSTON	OVERTON	36.39	-85.33	(OHX)
1:35	1.5"	1 E COOKEVILLE	PUTNAM	36.16	-85.48	REPORT OF 1.5 INCH HAIL IN COOKEVILLE. (OHX)
1:35	1.5"	COOKEVILLE	PUTNAM	36.16	-85.51	REPORT OF 1.5 INCH HAIL IN COOKEVILLE. (OHX)
2:22	1"	3 S COOKEVILLE	PUTNAM	36.11	-85.51	REPORT OF QUARTER SIZE HAIL SOUTH OF COOKEVILLE IN PUTNAM COUNTY. (OHX)
5:36	1"	6 WSW CHARLESTON	BRADLEY	35.27	-84.86	(MRX)
10:10	1.25"	SMITHVILLE	DE KALB	35.96	-85.82	IMAGE SHOWED HAIL COVERING PATIO WITH SOME HAIL UP TO HALF DOLLAR SIZE. (OHX)
10:15	1.75"	8 ESE SMITHVILLE	DE KALB	35.92	-85.68	TWITTER REPORT OF HAIL UP TO GOLF BALL SIZE IN EASTERN DE KALB COUNTY (OHX)
10:16	1.75"	9 SW BAKERS CROSSROADS	WHITE	35.92	-85.65	TWITTER REPORT OF PEA TO GOLFBALL SIZE HAIL ON THE DE KALB/WHITE COUNTY LINE (OHX)

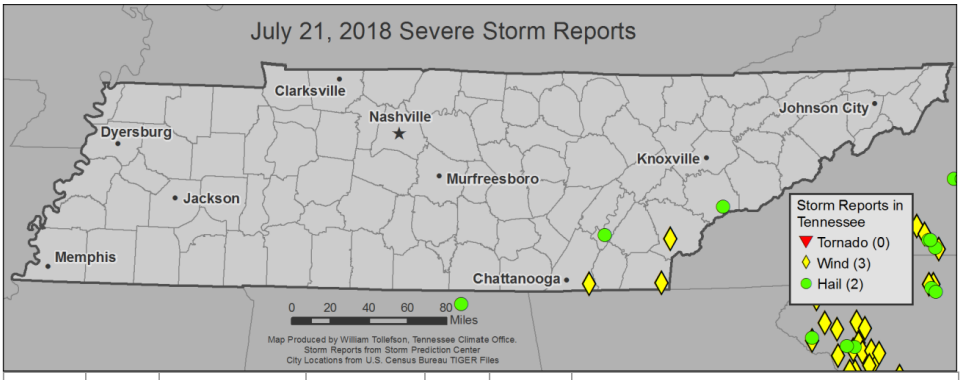
10:43	1"	8 SE SPENCER	VAN BUREN	35.65	-85.36	QUARTER SIZE HAIL REPORTED AT FALL CREEK FALLS CAMPGROUND (OHX)

Time	Speed	Location	County	Lat	Lon	Comments
0:44	-	7 WNW LIVINGSTON	OVERTON	36.42	-85.44	TREE REPORTED DOWN NEAR 170 STANDING STONE PARK HIGHWAY NEAR HILLHAM. POWER OUTAGES REPORTED AS WELL. (OHX)
0:49	-	5 WSW GAINESBORO	JACKSON	36.33	-85.73	SEVERAL TREES REPORTED DOWN IN THE FLYNNS LICK AREA. (OHX)
1:08	-	CELINA	CLAY	36.55	-85.5	SEVERAL TREES DOWN ON MOSS ARCOT ROAD. (OHX)
1:16	-	JAMESTOWN	FENTRESS	36.43	-84.94	MINOR ROOF DAMAGE TO A HOUSE IN JAMESTOWN OFF OF HWY 52 EAST. TREE DOWN ON STEVENS ROAD. (OHX)
1:25	-	LIVINGSTON	OVERTON	36.39	-85.33	NUMEROUS TREES DOWN IN LIVINGSTON INCLUDING ON PRESTON STREET...WEST MAIN STREET...MONTEREY HIGHWAY...1972 VILLA DRIVE...AND 900 NORTH CHURCH STREET. POWER LINES ALSO D (OHX)
1:30	-	7 N MONTEREY	OVERTON	36.24	-85.27	TREES DOWN ON RUSHING SPRINGS ROAD (OHX)
1:33	-	1 SSE CUMMINS FALLS STA	JACKSON	36.24	-85.56	TREES DOWN WITH POWER OUTAGES ON CUMMINS MILL RD. (OHX)
1:35	-	5 N ALGOOD	PUTNAM	36.26	-85.45	TREES DOWN NEAR ZEB WARREN ROAD AND BEN MASON ROAD. (OHX)
1:40	-	LA FOLLETTE	CAMPBELL	36.37	-84.13	SEVERAL TREES DOWN ACROSS THE COUNTY. (MRX)
1:49	-	LA FOLLETTE	CAMPBELL	36.37	-84.13	TREE DOWN ON SOUTH HIGHWAY 25. (MRX)
2:15	-	4 NW COOKEVILLE	PUTNAM	36.19	-85.55	DELAYED REPORT FROM FRIDAY JULY 20. SEVERAL TREES BLOWN DOWN ON WAKEFIELD DRIVE NEAR COUNTY FARM ROAD (OHX)
2:17	-	4 WNW COOKEVILLE	PUTNAM	36.19	-85.57	DELAYED REPORT FROM FRIDAY JULY 20. SEVERAL TREES SNAPPED AND UPROOTED AROUND THE POST OAK SHADE CHURCH AT THOMAS ROAD AND PIPPIN ROAD INTERSECTION (OHX)
2:20	-	2 WNW COOKEVILLE	PUTNAM	36.17	-85.54	DELAYED REPORT FROM FRIDAY JULY 20. CANE CREEK PARK MOUNTAIN BIKE TRAIL CLOSED FOR 2 TO 3 WEEKS DUE TO NUMEROUS TREES DOWN BLOCKING THE TRAIL. (OHX)
2:23	-	BEAN STATION	GRAINGER	36.34	-83.28	SEVERAL TREES DOWN IN GRAINGER COUNTY... TENNESSEE. (MRX)

2:27	-	MAYNARDVILLE	UNION	36.25	-83.81	SEVERAL TREES DOWN ACROSS THE MAYNARDVILLE... TENNESEE AREA. (MRX)
2:38	-	KINGSPORT	SULLIVAN	36.53	-82.56	TRAFFIC SIGNALS AND WIRES DOWN IN KINGSPORT DUE TO THE HIGH WINDS. (MRX)
2:45	-	HALLS	KNOX	36.08	-83.93	SEVERAL TREES DOWN. ONE TREE OVER A CAR. (MRX)
2:50	73	4 NE MORRISTOWN	HAMBLEN	36.26	-83.25	MEASURED REPORT OF 73 MPH AT WHITESBURG... TN. (MRX)
2:51	-	7 SSW LAKE TANSI	CUMBERLAND	35.78	-85.1	REPORTS OF TREES DOWN AND A TRANSFORMER DOWN ON VANDEVER ROAD IN CUMBERLAND COUNTY (OHX)
3:12	-	ELIZABETHTON	CARTER	36.34	-82.23	SEVERAL TREES DOWN ACROSS CARTER COUNTY... TENNESSEE. (MRX)
3:31	-	COSBY	COCKE	35.81	-83.25	SEVERAL TREES DOWN ACROSS THE COSBY AREA DUE TO HIGH WINDS. (MRX)
3:36	-	5 SW MOUNTAIN CITY	JOHNSON	36.42	-81.87	SEVERAL TREES DOWN AROUND THE COMMUNITY OF BUTLER... TENNESSEE. (MRX)
3:40	-	SPRING CITY	RHEA	35.69	-84.86	DAMAGE TO A HOME IN SPRING CITY. TIME IS ESTIMATED. (MRX)
4:06	-	DECATUR	MEIGS	35.53	-84.79	SEVERAL TREES DOWN ACROSS THE CITY OF DECATUR... TENNESSEE. (MRX)
4:20	-	SPRING CITY	RHEA	35.69	-84.86	NUMEROUS TREES AND POWER LINES DOWN ACROSS THE NORTHERN HALF OF THE COUNTY. PENNY SIZE HAIL AS WELL. (MRX)
4:35	-	PIKEVILLE	BLEDSOE	35.61	-85.2	SEVERAL TREES DOWN ACROSS THE PIKEVILLE... TN AREA. (MRX)
4:39	-	DECATUR	MEIGS	35.53	-84.79	TREES DOWN COUNTY-WIDE ACROSS MEIGS COUNTY... TENNESSEE. (MRX)
4:57	-	GAINESBORO	JACKSON	36.36	-85.65	LARGE TREE BLOCKING HIGHWAY 56 SOUTH LEAVING GAINSBORO GOING TOWARD BAXTER AT HALE HOLLOW. (OHX)
5:10	-	DAYTON	RHEA	35.49	-85.01	MULTIPLE TREES DOWN. (MRX)
5:15	-	BENTON	POLK	35.18	-84.65	2 TREES DOWN HIGHWAY 64. (MRX)
5:15	-	4 NW COOKEVILLE	PUTNAM	36.2	-85.56	A POWER POLE WAS BLOWN DOWN AT CUMBY ROAD AND PIPPIN ROAD. THIS WAS REPORTED ON THE PUTNAM COUNTY EMA FACEBOOK SITE. TIME WAS ESTIMATED BY RADAR. (OHX)
5:25	-	PIKEVILLE	BLEDSOE	35.61	-85.2	SEVERAL TREES DOWN AND A POWER LINE DOWN ACROSS THE ROAD. (MRX)

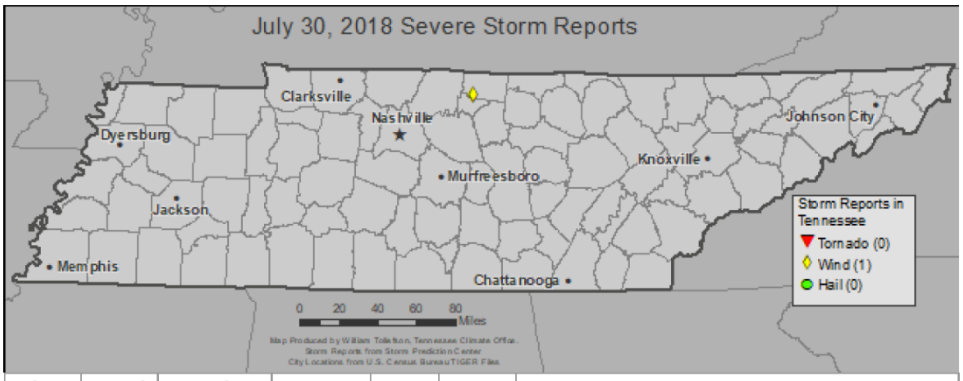
5:40	-	3 W COOKEVILLE	PUTNAM	36.16	-85.56	TREES AND POWER LINES WERE BLOWN DOWN ALONG PIPPIN ROAD. REPORT WAS POSTED ON PUTNAM COUNTY EMA FACEBOOK. TIME WAS ESTIMATED BY RADAR. (OHX)
5:40	-	LAFAYETTE	MACON	36.52	-86.03	NUMEROUS TREES AND SOME POWER LINES DOWN ACROSS MACON COUNTY FROM LAFAYETTE EASTWARD (OHX)
5:48	-	MONTEREY	PUTNAM	36.13	-85.24	DELAYED REPORT FROM SATURDAY JULY 21. NUMEROUS TREES BLOWN DOWN ALL ACROSS PUTNAM COUNTY. (OHX)
5:50	-	HUNTSVILLE	SCOTT	36.41	-84.49	A FEW TREES DOWN NEAR HUNTSVILLE (MRX)
5:54	-	CALHOUN	MCMINN	35.3	-84.75	2 TREES DOWN ON THE SOUTHERN END OF MCMINN COUNTY. (MRX)
5:59	-	COOKEVILLE	PUTNAM	36.16	-85.5	SEVERAL TREES REPORTEDLY BLOWN DOWN IN COOKEVILLE (OHX)
6:30	-	5 ESE CRAB ORCHARD	CUMBERLAND	35.88	-84.81	DELAYED REPORT FROM SATURDAY JULY 21. SEVERAL TREES BLOWN DOWN IN THE OZONE AND WESTEL COMMUNITIES. (OHX)
6:30	-	CROSSVILLE	CUMBERLAND	35.95	-85.04	HIGHWAY DEPARTMENT REPORTED NUMEROUS TREES DOWN... BLOCKING ROADS ACROSS CUMBERLAND COUNTY. (OHX)
6:35	-	LA FOLLETTE	CAMPBELL	36.37	-84.13	SEVERAL TREES DOWN JUST EAST OF LA FOLLETTE (MRX)
6:38	-	3 W TUCKERS CROSSROADS	WILSON	36.19	-86.22	TREE ACROSS ROADWAY AT 4033 BLUEBIRD ROAD (OHX)
6:40	-	ROCKWOOD	ROANE	35.87	-84.68	SEVERAL TREES DOWN AROUND THE ROCKWOOD AREA (MRX)
6:40	-	LEBANON	WILSON	36.21	-86.32	DELAYED REPORT FROM SATURDAY JULY 21. TREE FELL ON A JEEP IN LEBANON. (OHX)
6:42	-	SPARTA	WHITE	35.93	-85.47	DELAYED REPORT FROM SATURDAY JULY 21. TREE UPROOTED AT THE WHITE COUNTY PUBLIC LIBRARY (OHX)
6:45	-	3 SW LINWOOD	WILSON	36.13	-86.21	LARGE TREE ACROSS ROADWAY ON SPARTA PIKE AT YOUNG ROAD (OHX)
7:15	-	MCMINNVILLE	WARREN	35.69	-85.78	DELAYED REPORT FROM SATURDAY JULY 21. TREES DOWN THROUGHOUT WARREN COUNTY WITH SOME FALLING ON HOUSES. (OHX)
7:37	-	PIKEVILLE	BLEDSOE	35.61	-85.2	NUMEROUS REPORTS OF TREES DOWN ACROSS THE NORTHERN PART OF THE COUNTY. (MRX)
7:55	-	7 ESE MANCHESTER	COFFEE	35.41	-85.97	TREES REPORTED DOWN ACROSS EASTERN COFFEE COUNTY (OHX)

7:55	-	DAYTON	RHEA	35.49	-85.01	SEVERAL TREES DOWN ACROSS THE SOUTHERN PART OF RHEA COUNTY. (MRX)
8:00	-	DUNLAP	SEQUATCHIE	35.37	-85.39	SEVERAL TREES DOWN COUNTY-WIDE. (MRX)
8:15	-	CALHOUN	MCMINN	35.3	-84.75	TREE DOWN ON HIGHWAY 163 (MRX)
8:24	-	OOLTEWAH	HAMILTON	35.06	-85.09	ONE TREE DOWN IN OOLTEWAH (MRX)
8:25	-	5 WSW TRACY CITY	GRUNDY	35.24	-85.83	DELAYED REPORT FROM SATURDAY JULY 21. TREE BLOWN DOWN IN MONTEAGLE (OHX)
8:40	-	COPPERHILL	POLK	34.99	-84.36	SEVERAL TREES SNAPPED AND UPROOTED NEAR THE OCOEE WHITEWATER CENTER. (MRX)
11:08	-	PIKEVILLE	BLEDSOE	35.61	-85.2	MULTIPLE TREES DOWN. (MRX)



Time	Size	Location	County	Lat	Lon	Comments
21:35	1"	BIG SPRING	MEIGS	35.4	-84.9	(MRX)
23:20	1"	CADES COVE	BLOUNT	35.6	-83.81	(MRX)

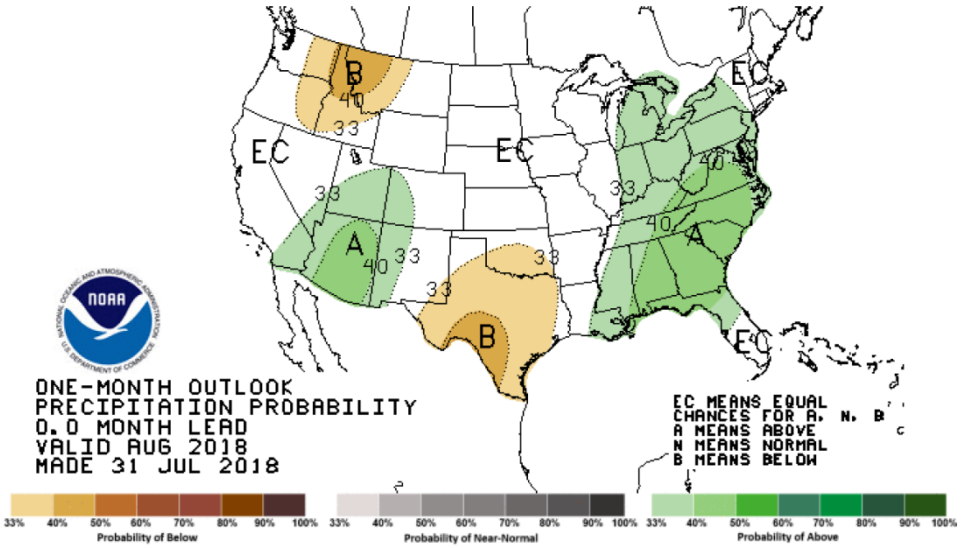
Time	Speed	Location	County	Lat	Lon	Comments
12:24	-	TELLICO PLAINS	MONROE	35.37	-84.3	SEVERAL TREES AND POWER LINES DOWN. (MRX)
22:05	-	COLLEGEDALE	HAMILTON	35.04	-85.05	MULTIPLE TREES AND POWERLINES DOWN. (MRX)
22:40	-	DUCKTOWN	POLK	35.04	-84.39	SEVERAL TREES DOWN ACROSS THE COUNTY. (MRX)



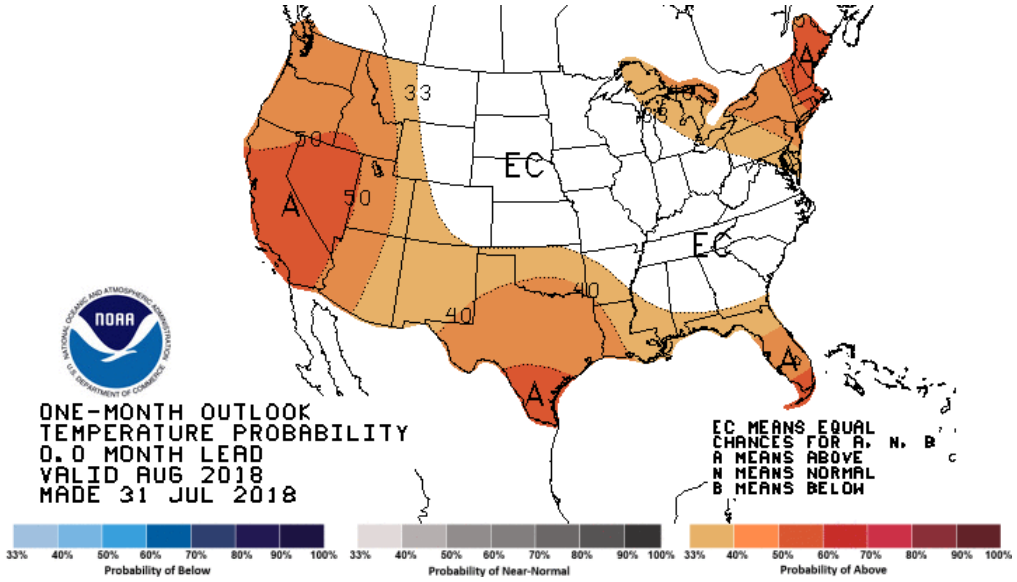
Time	Speed	Location	County	Lat	Lon	Comments
23:55	-	6 SW LAFAYETTE	MACON	36.47	-86.11	TREES DOWN ON BEECH HILL. TIME ESTIMATED VIA RADAR. (OHX)

CPC outlooks for the next month:

The Climate Prediction Center outlook for precipitation for August shows a likely continuance of the pattern from the second half of July. With continued wetter than average conditions across the eastern third of the state, but with a slight chance of being wetter than average in the rest of the state as well (except the area right along the Mississippi River).



The temperature outlook indicates that the entire state has equal chances of being at, above, or below normal for August; indicating a lack of a strong climatological control on our temperatures for the next month.


ONE-MONTH OUTLOOK
TEMPERATURE PROBABILITY
0.0 MONTH LEAD
VALID AUG 2018
MADE 31 JUL 2018
EC MEANS EQUAL
CHANCES FOR A, N, B
A MEANS ABOVE
N MEANS NORMAL
B MEANS BELOW
33
50
50
40
40
EC
EC
A
A
A
A
33% 40% 50% 60% 70% 80% 90% 100%
Probability of Below
33% 40% 50% 60% 70% 80% 90% 100%
Probability of Near-Normal
33% 40% 50% 60% 70% 80% 90% 100%
Probability of Above

Appendix C: Python Script Automation

Python Script for Automation of Daily Storm Reports:

Automation of Daily Storm Reports script:

```
import urllib.request
import os
import time
import csv
from datetime import timedelta, datetime

#Modules
def Oz(FileName, stump, Time, F_Scale, Location, County, State, Lat, Lon, Comments):
    arcpy.CreateFeatureclass_management(bombsite, FileName, "POINT", "", "", "", spatialref)
    arcpy.AddField_management(stump, Time, "SHORT")
    arcpy.AddField_management(stump, F_Scale, "TEXT")
    arcpy.AddField_management(stump, Location, "TEXT")
    arcpy.AddField_management(stump, County, "TEXT")
    arcpy.AddField_management(stump, State, "TEXT")
    arcpy.AddField_management(stump, Lat, "FLOAT")
    arcpy.AddField_management(stump, Lon, "FLOAT")
    arcpy.AddField_management(stump, Comments, "TEXT", "", "", fieldlength)
    arcpy.AddField_management(stump, "Date", "TEXT")
```

```
    arcpy.AddField_management(stump, "Count", "SHORT")



def Avalon(Tango, Stump, Time, F_Scale, Location, County, State, Lat, Lon, Comments,
Wizard):
    with arcpy.da.InsertCursor(Stump, ("SHAPE@XY", Time, F_Scale, Location, County, State,
Lat, Lon, Comments))as wizard:
        for players in Tango:
            Timez = players[0]
            TimeInt2 = int(Timez)
            F_Scalez = players[1]
            Countyz = players[3]
            State = players[4]
            Latz = players[5]
            numLat = float(Latz)
            Lonz = players[6]
            numLon = float(Lonz)
            Commentsz = players[7]
            Locationz = players[2]
            wizard.insertRow(((float(numLon), float(numLat)), TimeInt2, F_Scalez, Locationz,
Countyz, State, numLat, numLon, Commentsz))

#time setup
```

```
timestream = datetime.now() - timedelta(days= 1)
timestr = timestream.strftime("%m%d%y")
propmonth = timestream.strftime('%B')
propday = timestream.strftime('%d')
propyear = timestream.strftime('%Y')
dytextdate = propmonth + ' ' + propday + ', ' + propyear
begin_time = time.time()
#add day of week at beginning?


#Variables
placeholder_spot =
r"E:\Grad_School\Archive_2\Thesis\State_Climate_Office\TCO_Automation\NOAA_DATA_P
ULL"
basicurl = r"http://www.spc.noaa.gov/climo/reports"
torncsv = 'yesterday_torn.csv'
hailcsv = 'yesterday_hail.csv'
windcsv = 'yesterday_wind.csv'
tornrawscv = 'yesterday_raw_torn.csv'
weatherlistOne = [torncsv, windcsv, hailcsv, tornrawscv]
weatherlist = [torncsv, windcsv, hailcsv]
spatialsurogate =
r"E:\Grad_School\Archive_2\Thesis\State_Climate_Office\TCO_Automation\NOAA_DATA_P
ULL\SpatialReference\SpatialReference.shp"
```

```
spatialref = arcpy.Describe(spatialsurogate).spatialReference
fieldlength = 300
partarch =
r'E:\Grad_School\Archive_2\Thesis\State_Climate_Office\TCO_Automation\NOAA_DATA_P
ULL\Archive'
bombsite = partarch + '\\' + 'SPC_Daily_' + timestr
boundary =
r'E:\Grad_School\Archive_2\Thesis\State_Climate_Office\TCO_Automation\NOAA_DATA_P
ULL\Boundary\TNBoundary.shp'

#Folder creation
try:
    os.mkdir(bombsite)
    print('folder created')
except:
    print('folder already made')

#Set workspace environment for Arcpy and os
curdir = os.getcwd()
print(curdir)
newdir = os.chdir(bombsite)
curdir = os.getcwd()
print(curdir)
```

```
arcpy.env.workspace = bombsite
arcpy.env.overwriteOutput = True


#Pulling data from SPC website
for climateOne in weatherlistOne:
    stringcobbleone = basicurl + '/' + climateOne
    print(stringcobbleone)
    req = urllib.request.Request(stringcobbleone)
    reqopen = urllib.request.urlopen(req)
    output = open(climateOne, 'wb')
    output.write(reqopen.read())
    output.close


#Setting up to read csv files
for climate in weatherlist:
    newstringcobble = bombsite + '\\' + climate
    print(newstringcobble)
    climatestr = climate.replace('.csv', '')
    newreader = open(newstringcobble, 'r')
    print(newreader)
    csvreader = csv.reader(newreader, delimiter= ',')
    print(csvreader)
    header = csvreader.__next__()
```

```
print(header)
filenameT = climatestr + "_" + timestr + ".shp"
filenameTclip = climatestr + '_clip_' + timestr + '.shp'
print(filenameT)
shortnameT = climatestr + ".shp"
shortnameTclip = climatestr + '_clip_' + ".shp"
print(shortnameT)
fullpath = bombsite + '\\' + filenameT
fullpathclip = bombsite + '\\' + filenameTclip
print(fullpath)
branchT = placeholder_spot + '\\' + shortnameT
branchTclip = placeholder_spot + '\\' + shortnameTclip
print(branchT)

#establishing file headers and providing input for modules & running them
if climate == torncsv:
    Time = "Time"
    F_Scale = "F_Scale"
    Location = "Location"
    County = "County"
    State = "State"
    Lat = "Lat"
    Lon = "Lon"
```

```
        Comments = "Comments"
        wizard = "gandalf"

        Time1 = header.index("Time")
        F_Scale1 = header.index("F_Scale")
        Location1 = header.index("Location")
        County1 = header.index("County")
        State1 = header.index("State")
        Lat1 = header.index("Lat")
        Lon1 = header.index("Lon")
        Comments1 = header.index("Comments")
        Oz(filenameT, fullpath,Time, F_Scale, Location, County, State, Lat, Lon, Comments)
        print ("oz1")
        Avalon(csvreader, fullpath, Time,F_Scale, Location, County, State, Lat, Lon, Comments,
wizard)
        print("Avalon1")

    elif climate == hailcsv:
        TimeH = "Time"
        Size = "Size"
        LocationH = "Location"
        CountyH = "County"
        StateH = "State"
```

```
        LatH = "Lat"
        LonH = "Lon"
        CommentsH = "Comments"
        wizard = "warlock"

        Time2 = header.index("Time")
        F_Scale2 = header.index("Size")
        Location2 = header.index("Location")
        County2 = header.index("County")
        State2 = header.index("State")
        Lat2 = header.index("Lat")
        Lon2 = header.index("Lon")
        Comments2 = header.index("Comments")
        Oz(filenameT, fullpath, TimeH, Size, LocationH, CountyH, StateH, LatH, LonH,
CommentsH)
        print ("oz2")
        Avalon(csvreader, fullpath, TimeH, Size, LocationH, CountyH, StateH, LatH, LonH,
CommentsH, wizard)
        print("Avalon2")

    elif climate == windcsv:
        TimeW = "Time"
        Speed = "Speed"
```

```
        LocationW = "Location"
        CountyW = "County"
        StateW = "State"
        LatW = "Lat"
        LonW = "Lon"
        CommentsW = "Comments"
        wizard = "mage"

        Time3 = header.index("Time")
        F_Scale3 = header.index("Speed")
        Location3 = header.index("Location")
        County3 = header.index("County")
        State3 = header.index("State")
        Lat3 = header.index("Lat")
        Lon3 = header.index("Lon")
        Comments3 = header.index("Comments")
        Oz(filenameT, fullpath, TimeW, Speed, LocationW, CountyW, StateW, LatW, LonW,
CommentsW)
        print("oz complete")
        Avalon(csvreader, fullpath, TimeW, Speed, LocationW, CountyW, StateW, LatW, LonW,
CommentsW, wizard)
        print("Avalon complete")
    newreader.close
```

```
    #clip, calculate, copy without date
    arcpy.analysis.Clip(fullpath, boundary, fullpathclip, None)
    print('clip complete')
    SPC_fields = arcpy.ListFields(fullpathclip)
    time_expr = '"{}"'.format(dytextdate)
    arcpy.management.CalculateField(boundary, 'Date', time_expr, 'PYTHON3')
    arcpy.management.CalculateField(fullpathclip, 'Date', time_expr, 'PYTHON3')
    arcpy.management.CalculateField(fullpath, 'Date', time_expr, 'PYTHON3')
    arcpy.MakeFeatureLayer_management(fullpathclip, "myfeatures")
    result = arcpy.GetCount_management("myfeatures")
    result_expr = '"{}"'.format(result)
    arcpy.management.CalculateField(fullpathclip, 'Count', result_expr, 'PYTHON3')
    print('field calculations complete')
    arcpy.Copy_management(fullpath, branchT)
    arcpy.Copy_management(fullpathclip, branchTclip)
    print('copy for presentation in map')


#printing out stormreport map
maploc = placeholder_spot + '\StormReport_v2.aprx'
stormproject = arcpy.mp.ArcGISProject(maploc)
storm_report_map = stormproject.listLayouts("Storm_Report_Map*")[0]
PNG_date = 'Storm_Report_' + timestr + '.png'
```

```
PNG_spot = placeholder_spot + '\StormReport_pdf' + '\\' + PNG_date

storm_report_map.exportToPNG(PNG_spot, resolution = 600)

print('Map has successfully printed.')

print ('%s total seconds' % (time.time() - begin_time))
```

Batch File Code:

```
cd "C:\Users\ZMJS5\AppData\Local\ESRI\conda\envs\my_arcgispro-py3"

"C:\Users\ZMJS5\AppData\Local\ESRI\conda\envs\my_arcgispro-py3\python.exe"
"E:\Grad_School\Archive_2\Thesis\State_Climate_Office\TCO_Automation\NOAA_DATA_P
ULL\NOAA_Scriptcsv_merge_060319_v2.py"
```

Python Scripts for Automation of Hazard Index Web Application:

Downloading data from the NDFD website script:

```
#script for Python3

#import requests
import os
import time
import arcpy
import urllib.request
import urllib.error
```

```
print ('imports successful')


timestr = time.strftime('%m%d%y')


#GRIB2 data names and variables
ndfd_dsapt = 'ds.apt.bin'
ndfd_dsconhazo = 'ds.conhazo.bin'
ndfd_dscritfireo = 'ds.critfireo.bin'
ndfd_dsdryfireo = 'ds.dryfireo.bin'
ndfd_dsiceaccum = 'ds.iceaccum.bin'
ndfd_dsmaxrh = 'ds.maxrh.bin'
ndfd_dsmaxt = 'ds.maxt.bin'
ndfd_dsminrh = 'ds.minrh.bin'
ndfd_dsmint = 'ds.mint.bin'
ndfd_dsphail = 'ds.phail.bin'
ndfd_dspop12 = 'ds.pop12.bin'
ndfd_dsptornado = 'ds.ptornado.bin'
ndfd_dsptotsvrtstm = 'ds.ptotsvrtstm.bin'
ndfd_dsptotxsvrtstm = 'ds.ptotxsvrtstm.bin'
ndfd_dsptstmwinds = 'ds.ptstmwinds.bin'
ndfd_dspxhail = 'ds.pxhail.bin'
ndfd_dspxtornado = 'ds.pxtornado.bin'
```

ndfd_dspxtstmwinds = 'ds.pxtstmwinds.bin'

ndfd_qpf = 'ds.qpf.bin'

ndfd_dsrhm = 'ds.rhm.bin'

ndfd_dssky = 'ds.sky.bin'

ndfd_dssnow = 'ds.snow.bin'

ndfd_dstcwspdabv34c = 'ds.tcwspdabv34c.bin'

ndfd_dstcwspdabv34i = 'ds.tcwspdabv34i.bin'

ndfd_dstcwspdabv50c = 'ds.tcwspdabv50c.bin'

ndfd_dstcwspdabv50i = 'ds.tcwspdabv50i.bin'

ndfd_dstcwspdabv64c = 'ds.tcwspdabv64c.bin'

ndfd_dstcwspdabv64i = 'ds.tcwspdabv64i.bin'

ndfd_dstd = 'ds.td.bin'

ndfd_dstemp = 'ds.temp.bin'

ndfd_dswaveh = 'ds.waveh.bin'

ndfd_dswdir = 'ds.wdir.bin'

ndfd_dswgust = 'ds.wgust.bin'

ndfd_dswspd = 'ds.wspd.bin'

ndfd_dswwa = 'ds.wwa.bin'

ndfd_dswx = 'ds.wx.bin'

ndfd_lsl = 'ls-l'

ndfd_lslt = 'ls-lt'

```
ndfd_conus =
r'http://tgftp.nws.noaa.gov/SL.us008001/ST.opnl/DF.gr2/DC.ndfd/AR.conus/VP.001-003'
#continuous US

print('variables set')

superlist = [ndfd_dsiceaccum, ndfd_qpf, ndfd_dsconhazo, ndfd_dssnow,
        ndfd_dsapt, ndfd_dswgust, ndfd_dsptornado]

ndfd_conusf =
r'E://Grad_School//Archive_2//Thesis//State_Climate_Office//TCO_Automation//ndfd_conus' +
timestr

#Dated folder creation
try:
   os.mkdir(ndfd_conusf)
except:
   print('folder already exists')

#Assigning where data will be copied to
curdir = os.getcwd()
print(curdir)
newdir = os.chdir(ndfd_conusf)
```

```
curdir = os.getcwd()

print(curdir)


#Download of data

for load in superlist:

    stringCoble = str(ndfd_conus + '/' + str(load))

    print (stringCoble)

    filer = urllib.request.Request(stringCoble)

    responseapt = urllib.request.urlopen(filer)

    print('this is were im telling it to go', responseapt)

    print('were we actually go', responseapt.geturl())

    output = open(load, 'wb')

    output.write(responseapt.read())

    output.close


print('hahahahaha yes!')
```

GRIB2 data converted to vector polygons script

```
import arcpy

import os

import time

from arcpy.sa import*

arcpy.CheckOutExtension("Spatial")
```

```
import gdal


#Variables, folder making, time and overwrite
homefile = r'E://Grad_School//Archive_2//Thesis//State_Climate_Office//TCO_Automation'
boundary =
r'E://Grad_School//Archive_2//Thesis//State_Climate_Office//TCO_Automation//Transfer_file.g
db//TNBoundary'
transfer_file =
r'E://Grad_School//Archive_2//Thesis//State_Climate_Office//TCO_Automation//Transfer_file_
newvariables.gdb'
tifflist = []
mergeraster = []
clippedgroup = []
singlegroup = []
vectorgroup = []


binlist = ['ds.apt.bin', 'ds.snow.bin', 'ds.wgust.bin', 'ds.qpf.bin','ds.conhazo.bin', 'ds.iceaccum.bin']
timestr = time.strftime("%m%d%y")
binpaths =
r'E://Grad_School//Archive_2//Thesis//State_Climate_Office//TCO_Automation//ndfd_conus' +
timestr
bombsite = homefile + '//' + 'NDFD_Daily_Rasters_' + timestr
print('imports and variables setup')
```

```
begin_time = time.time()
fehrclip = homefile + '//' + 'temp_max_TNfehr.tif'
arcpy.env.snapRaster = fehrclip
try:
    os.mkdir(bombsite)
    print('folder created')
except:
    print('file already made')

arcpy.env.workspace = bombsite
arcpy.env.overwriteOutput = True

arcpy.CreateFileGDB_management(bombsite, 'NDFD_resample_' + timestr)#these used? need
time gdb

resamplelocation = bombsite + '//' + 'NDFD_resample_' + timestr + '.gdb' #these used? need time
gdb

#GRIB2 pulling out rasters and converting to mosaicked tiffs
for weather in binlist:
    ndfd_data = binpaths + '//' + weather
    if weather == 'ds.wgust.bin':
            forcast = 'wg_'
```

```
elif weather == 'ds.apt.bin':
        forcast = 'apt_'
elif weather == 'ds.snow.bin':
        forcast = 'sn_'
elif weather == 'ds.iceaccum.bin':
        forcast = 'ic_'
elif weather == 'ds.wspd.bin':
        forcast = 'wsp_'
elif weather == 'ds.qpf.bin':
        forcast = 'ra_'
elif weather == 'ds.conhazo.bin':
        forcast = 'con_'
gdalopen = gdal.Open(ndfd_data, gdal.GA_ReadOnly)
numero = gdalopen.RasterCount
print(numero)
numa = 0
merge_name = forcast + '.tif'

if numero <= 1:
    print('only one record in ' + weather)
    subout = forcast + str(numa)
    suboutpath = bombsite + '//' + subout
    castle = arcpy.Describe(ndfd_data)
```

```
    keep = castle.spatialReference
    print(subout + ' created')
    asciioutput = bombsite + '//' + forcast + str(numa) + '_ascii.txt'
    arcpy.RasterToASCII_conversion(ndfd_data, asciioutput)
    print('raster to ascii worked')
    tiffnopath = forcast + str(numa) + '_ti.tif'
    tiffoutput = str(bombsite) + '//' + tiffnopath
    arcpy.ASCIIToRaster_conversion(asciioutput, tiffoutput)
    print('ascii to raster worked')
    arcpy.DefineProjection_management(tiffoutput, keep)
    print(tiffnopath + ' proj. defined')
    singlegroup.append(tiffnopath)

else:
    for num in range(numero):
        print(numa)
        subout = forcast + str(numa)
        suboutpath = bombsite + '//' + subout
        arcpy.ExtractSubDataset_management(ndfd_data, suboutpath, numa)
        print(str(numa) + 'extract completed')
        castle = arcpy.Describe(suboutpath)
        keep = castle.spatialReference
        remask = arcpy.sa.ExtractByMask(suboutpath, fehrclip)
```

```
        print('extract done')
        test_save = suboutpath + '_clip'
        remask.save(test_save)
        asciioutput = bombsite + '//' + forcast + str(numa) + '_ascii.txt'
        arcpy.RasterToASCII_conversion(test_save, asciioutput)
        print('raster to ascii done')
        tiffnopath = forcast + str(numa) + '_ti.tif'
        tiffoutput = str(bombsite) + '//' + tiffnopath
        tifflist.append(tiffoutput)
        arcpy.ASCIIToRaster_conversion(asciioutput, tiffoutput, "FLOAT")
        print('ascii to raster done')
        arcpy.DefineProjection_management(tiffoutput, keep)
        numa += 1



    if weather == 'ds.apt.bin':
        m_temp_min = 'temp_min.tif'
        m_temp_max = 'temp_max.tif'
        try:
            arcpy.MosaicToNewRaster_management(tifflist, bombsite, m_temp_max, '#',
'32_BIT_FLOAT', '#', '1', 'MAXIMUM', 'FIRST')
```

```
            arcpy.MosaicToNewRaster_management(tifflist, bombsite, m_temp_min, '#',
'32_BIT_FLOAT', '#', '1', 'MINIMUM', 'FIRST')
            mergeraster.append(m_temp_min)
            mergeraster.append(m_temp_max)
        except:
            print('ds file failed')

    elif weather == 'ds.wgust.bin' or weather == 'ds.wspd.bin' or weather == 'ds.qpf.bin' or
weather == 'ds.conhazo.bin':
        try:
            arcpy.MosaicToNewRaster_management(tifflist, bombsite, merge_name, '#',
'32_BIT_FLOAT', '#', '1', 'MAXIMUM', 'FIRST')
            mergeraster.append(merge_name)
        except:
            print('ds file failed')
    else:
        arcpy.MosaicToNewRaster_management(tifflist, bombsite, merge_name, '#',
'32_BIT_FLOAT', '#', '1', 'SUM', 'FIRST')
        mergeraster.append(merge_name)
    print(weather + ' merged')
    del tifflist[:]

print('giant mergers done')
```

```
print('now for general cleaning up')
print('......')
print('......')
print(mergeraster)

# Converting tiffs from international measurements to US measurements
for TN in mergeraster:
    if TN.startswith('temp_min') or TN.startswith('temp_max'):
        fullblister = bombsite + '//' + TN
        fehr = Raster(fullblister)
        heit = (fehr-273.15) * (9/5) + 32
        notiff = TN.replace('.tif', '')
        notifffehr = notiff + 'fehr.tif'
        addfehr = bombsite + '//' + notifffehr
        heit.save(addfehr)
        clippedgroup.append(notifffehr)
        vectorgroup.append(notifffehr)
        print(TN + ' converted to F')
    #kg/m2 to inches
    elif TN.startswith('ic_'):
        icetransfer = bombsite + '//' + TN
        frozen = Raster(icetransfer)
        iceconvert = frozen * 0.039370
```

```
        icestrip = TN.replace('.tif', '')
        notiffice = icestrip + 'inch.tif'
        addice = bombsite + '//' + notiffice
        iceconvert.save(addice)
        clippedgroup.append(notiffice)
        vectorgroup.append(notiffice)
        print(TN + ' converted to inches from kgm2')
    #meters to inches
    elif TN.startswith('sn_'):
        snowtransfer = bombsite + '//' + TN
        snowman = Raster(snowtransfer)
        snowconvert = snowman * 39.370
        snowstrip = TN.replace('.tif', '')
        notiffsnow = snowstrip + 'inch.tif'
        addsnow = bombsite + '//' + notiffsnow
        snowconvert.save(addsnow)
        clippedgroup.append(notiffsnow)
        vectorgroup.append(notiffsnow)
        print(TN + ' converted from meters to inches')
    #ms-1 to mph
    elif TN.startswith('wg_') or TN.startswith('wsp_'):
        windtransfer = bombsite + '//' + TN
        windgust = Raster(windtransfer)
```

```
    windconvert = windgust * 2.236936271
    windstrip = TN.replace('.tif', '')
    notiffwind = windstrip + 'mph.tif'
    addwind = bombsite + '//' + notiffwind
    windconvert.save(addwind)
    clippedgroup.append(notiffwind)
    vectorgroup.append(notiffwind)
    print(TN + ' converted from ms-1 to mph')
#kg/m2 to inches
elif TN.startswith('ra_'):
    raintransfer = bombsite + '//' + TN
    rain = Raster(raintransfer)
    rainconvert = rain * 0.039370080320721
    rainstrip = TN.replace('.tif', '')
    notiffrain = rainstrip + 'inches.tif'
    addrain = bombsite + '//' + notiffrain
    rainconvert.save(addrain)
    clippedgroup.append(notiffrain)
    vectorgroup.append(notiffrain)
    print(TN + ' converted from kg/m2 to inches')
else:
    clippedgroup.append(TN)
    vectorgroup.append(TN)
```

```
        print(TN + 'clipped to TN boundary')
print('......')
print('......')


for single in singlegroup:
    singlestrip = single.replace('.tif', '')
    resampler = singlestrip + 'clipp.tif'
    singleoutput = bombsite + '//' + resampler
    singlepath = bombsite + '//' + single
    singlemask = arcpy.sa.ExtractByMask(singlepath, fehrclip)
    print(single + ' extracted by mask')
    singlemask.save(singleoutput)
    clippedgroup.append(resampler)
    vectorgroup.append(resampler)
print('singles now are ready to convert with multi rasters')


for clgr in clippedgroup:
    clgrpath = bombsite + '//' + clgr
    clgrstrip = clgr.replace('.tif', '')
    clgradd = clgrstrip + '5'
    clgroutraster = resamplelocation + '//' +  clgradd
    arcpy.management.Resample(clgrpath, clgroutraster, "5000 5000", "NEAREST")
```

```
    vectorgroup.append(clgradd)



arcpy.env.workspace = transfer_file
arcpy.env.overwriteOutput = True

#Converting tiffs to vector polygons
for vec in vectorgroup:
    if vec.endswith('5'):
        TN_clippedfull = resamplelocation + '//' + vec
        TN_point_strip = vec
    else:
        TN_clippedfull = bombsite + '//' + vec
        TN_point_strip = vec.replace('.tif', '')

    TN_point_path = transfer_file + '//' + TN_point_strip
    arcpy.conversion.RasterToPoint(TN_clippedfull, TN_point_path, "Value")
    print(TN_point_strip + ' done')

    TN_poly = TN_point_strip + 'poly'
    TN_poly_path = transfer_file + '//' + TN_poly
    arcpy.analysis.CreateThiessenPolygons(TN_point_path, TN_poly_path, "ALL")
    print(TN_poly + ' done')
```

```
    TN_polyclip = TN_point_strip + 'polyclip'
    TN_polyclip_path = transfer_file + '//' + TN_polyclip
    arcpy.analysis.Clip(TN_poly_path, boundary, TN_polyclip_path, None)
    print(TN_polyclip + ' done')



arcpy.CheckInExtension("Spatial")
print('all done with raster conversions and editing :)')
print ('%s total seconds' % (time.time() - begin_time))
```

Creating a Hazard Index Script:

```
import arcpy
import os
import time



#Variables
origin =
r'E:\Grad_School\Archive_2\Thesis\State_Climate_Office\TCO_Automation\Transfer_file_new
variables.gdb'
```

```
archive =
r'E:\Grad_School\Archive_2\Thesis\State_Climate_Office\TCO_Automation\Archive.gdb'
intersectlist = []
haznumlist = []
intersectlistre = []
deletefields = []
timestr = time.strftime("%m%d%y")

##########################################################################################
#####################################
#SQL statements for selecting hazard index data
lowhitemp = '"temp_max_grid" BETWEEN 100 AND 104.999'
midhitemp = '"temp_max_grid" BETWEEN 105 AND 114.999'
highhitemp = '"temp_max_grid" >= 115'

lowlowtemp = '"temp_min_grid" BETWEEN 0 AND -9.999'
midlowtemp = '"temp_min_grid" BETWEEN -10 AND -19.999'
highlowtemp = '"temp_min_grid" <= -20'

lowwgust = '"wg__grid" BETWEEN 40 AND 49.999'
midwgust = '"wg__grid" BETWEEN 50 AND 59.999'
highwgust = '"wg__grid" >= 60'
```

```
lowrainfall = '"ra__grid" BETWEEN 1 and 2.999'

midrainfall = '"ra__grid" BETWEEN 3 and 4.999'

highrainfall = '"ra__grid" >= 5'


lowsnowfall = '"sn__grid" BETWEEN 2 AND 3.999'

midsnowfall = '"sn__grid" BETWEEN 4 AND 7.999'

highsnowfall = '"sn__grid" >= 8'


lowiceaccum = '"ic__grid" BETWEEN 0.01 AND 0.099'

midiceaccum = '"ic__grid" BETWEEN 0.1 AND 0.249'

highiceaccum = '"ic__grid" >= 0.25'


lowwspd = '"wsp__grid" BETWEEN 40 AND 49.999'

midwspd = '"wsp__grid" BETWEEN 50 AND 59.999'

highwspd = '"wsp__grid" >= 60'


lowcon = '"con__grid" BETWEEN 3 AND 5'

midcon = '"con__grid" BETWEEN 5.99 AND 6'

highcon = '"con__grid" BETWEEN 6.99 AND 10'


Nada = 'haz_index IS NULL'
```

```
################################################################################
########################################
#Nested Dictionary with SQL statements
weatherlist =  {'ic_inchpolyclip': {lowiceaccum: 'low,', midiceaccum: 'medium,', highiceaccum:
'high,'},
        'sn_inchpolyclip': {lowsnowfall: 'low,', midsnowfall: 'medium,', highsnowfall: 'high,'},
        'temp_maxfehrpolyclip': {lowhitemp: 'low,', midhitemp: 'medium,', highhitemp:
'high,'},
        'temp_minfehrpolyclip': {lowlowtemp: 'low,', midlowtemp: 'medium,', highlowtemp:
'high,'},
        'wg_mphpolyclip': {lowwgust: 'low,', midwgust: 'medium,', highwgust: 'high,'},
        'ra_inchespolyclip': {lowrainfall: 'low,', midrainfall: 'medium,', highrainfall: 'high,'},
        'con_polyclip': {lowcon: 'low,', midcon: 'medium', highcon: 'high,'},
        'ic_inch5polyclip': {lowiceaccum: 'low,', midiceaccum: 'medium,', highiceaccum:
'high,'},
        'sn_inch5polyclip': {lowsnowfall: 'low,', midsnowfall: 'medium,', highsnowfall: 'high,'},
        'temp_maxfehr5polyclip': {lowhitemp: 'low,', midhitemp: 'medium,', highhitemp:
'high,'},
        'temp_minfehr5polyclip': {lowlowtemp: 'low,', midlowtemp: 'medium,', highlowtemp:
'high,'},
        'wg_mph5polyclip': {lowwgust: 'low,', midwgust: 'medium,', highwgust: 'high,'},
        'ra_inches5polyclip': {lowrainfall: 'low,', midrainfall: 'medium,', highrainfall: 'high,'},
        'con_5polyclip': {lowcon: 'low,', midcon: 'medium', highcon: 'high,'}}
```

```
################################################################################
################################################
#Creation of hazard index fields and using SQL statements to define measurement within hazard
index
arcpy.env.workspace = origin
arcpy.env.overwriteOutput = True
print('modules and variables done')


for weather, dictindex in weatherlist.items():
    clearday = weather.replace('polyclip', '')
    path = origin + '\\' + weather
    if clearday.endswith('5'):
        intersectlistre.append(path)
    else:
        intersectlist.append(path)
    if weather.startswith('temp_max'):
        conname = ' maximum temperature'
        namestrip = 'temp_max'
    elif weather.startswith('temp_min'):
        conname = ' minimum temperature'
        namestrip = 'temp_min'
    elif weather.startswith('ra'):
```

```
    conname = ' rain in 6 hours'
    namestrip = 'ra_'
elif weather.startswith('con'):
    conname = ' categorical convection'
    namestrip = 'con_'
elif weather.startswith('sn'):
    namestrip = 'sn_'
    conname = ' snowfall'
elif weather.startswith('ic'):
    conname = ' ice accumulation'
    namestrip = 'ic_'
elif weather.startswith('wg'):
    namestrip = 'wg_'
    conname = ' wind gust'
elif weather.startswith('wsp'):
    namestrip = 'wsp_'
    conname = ' max wind speed'
gridname = namestrip + '_grid'
hazname = namestrip + '_index'
haznum = namestrip + '_num'
haznumlist.append(haznum)
arcpy.management.AddField(path, gridname, 'DOUBLE')
arcpy.management.AddField(path, hazname, 'TEXT')
```

```
arcpy.management.AddField(path, haznum, 'LONG')
arcpy.management.CalculateField(path, gridname, '!grid_code!', 'PYTHON3')
hdfd_HI_individuals = arcpy.ListFields(path)
arcpy.MakeFeatureLayer_management(path, 'weather')

for ind in dictindex:
    arcpy.SelectLayerByAttribute_management('weather', 'NEW_SELECTION', ind)
    ind_haz_index = dictindex[ind] + conname
    ind_haz_fullquote = '"{}"'.format(ind_haz_index)
    arcpy.management.CalculateField('weather', hazname, ind_haz_fullquote, 'PYTHON3')
    print(ind_haz_index + ' calculated for haz_index')
    if ind_haz_index.startswith('high'):
        haznumber = 5
    elif ind_haz_index.startswith('medium'):
        haznumber = 4
    elif ind_haz_index.startswith('low'):
        haznumber = 3
    else:
        haznumber = 0
    arcpy.management.CalculateField('weather', haznum, haznumber, 'PYTHON3')
    print(str(haznumber) + ' calculated for haz_num')
Nado =  hazname + ' IS NULL'
arcpy.SelectLayerByAttribute_management('weather', 'NEW_SELECTION', Nado)
```

```
    arcpy.management.CalculateField('weather', haznum, 0, 'PYTHON3')
    arcpy.Delete_management('weather')
    print(weather + ' has field' + gridname + ' added')
print('weatherlist fields built')

##########################################################################################
#####################################################
#Merging the vector polygons together, deleting extra fields and archiving
intersect_vector = origin + '\\' + 'hdfd_HI_data'
intersectre_vector = origin + '\\' + 'hdfd_HI_data_5'
listforintersect = [[intersectlist, intersect_vector], [intersectlistre, intersectre_vector]]
for inter in listforintersect:
    enterlist = inter[0]
    interfile = inter[1]
    arcpy.analysis.Intersect(enterlist, interfile, 'ALL', None, 'INPUT')
    arcpy.management.AddField(interfile, 'haz_index', 'LONG')
    arcpy.env.workspace = interfile
    hdfd_HI_datafields = arcpy.ListFields(interfile)
    arcpy.MakeFeatureLayer_management(interfile, 'hazardindex')
    expression = '!ic__num!' ' + ' '!sn__num!' ' + ' '!temp_max_num!' ' + ' '!temp_min_num!' ' + '
'!ra__num!' ' + ' '!wg__num!' ' + ' '!con__num!'
    arcpy.SelectLayerByAttribute_management('hazardindex', 'NEW_SELECTION', Nada)
    arcpy.management.CalculateField('hazardindex', 'haz_index', expression, 'PYTHON3')
```

```
    arcpy.Delete_management("hazardindex")


    for java in hdfd_HI_datafields:
        if java.name.startswith('FID'):
            deletefields.append(java.name)
        elif java.name.startswith('Input'):
            deletefields.append(java.name)
        elif java.name.startswith('point'):
            deletefields.append(java.name)
        elif java.name.startswith('grid'):
            deletefields.append(java.name)
    arcpy.management.DeleteField(interfile, deletefields)


    splittwo = os.path.basename(interfile)
    uniontwo = os.path.join(archive,splittwo)
    archivefile = uniontwo + timestr
    arcpy.CopyFeatures_management(interfile, archivefile)


print('All done and ready for transfer')
```

Posting data to ArcGIS Online Script:

```
# coding: utf-8
```

```
# In[ ]:



import arcpy
import os, sys
from arcgis.gis import GIS

prjPath = r'E:\Grad_School\Archive_2\Thesis\State_Climate_Office\Final
scripts\VectorTile.aprx'
#sd_fs_name = 'Hazard_Index48hrs_WFL1'

sd_fs_name = ['Hazard_Index48hrs_WFL1', 'Updated_HazardIndex_WFL1']
portal = 'http://www.arcgis.com'
user = 'mshoop'
password = 'mom2468392'

shrOrg = True
shrEveryone = False
shrGroups = ''

relPath = sys.path[0]
sddraft = r'E:\Grad_School\Archive_2\Thesis\State_Climate_Office\Final
scripts\WebUpdate.sddraft'
```

```
sd = r'E:\Grad_School\Archive_2\Thesis\State_Climate_Office\Final scripts\WebUpdate.sd'

print('Creating SD file')
arcpy.env.overwriteOutput = True
prj = arcpy.mp.ArcGISProject(prjPath)
mp = prj.listMaps()[0]

for stan in sd_fs_name:

    arcpy.mp.CreateWebLayerSDDraft(mp, sddraft, stan, 'MY_HOSTED_SERVICES',
'FEATURE_ACCESS', '', True, True)
    arcpy.StageService_server(sddraft, sd)

    print('Connecting to {}'.format(portal))
    gis = GIS(portal, user, password)

    print('Search for original SD on portal..')
    sdItem = gis.content.search('{} AND owner:{}'.format(stan, user), item_type= 'Service
Definition')[0]
    print('Found SD: {}, ID: {} n Uploading and overwriting...'.format(sdItem.title, sdItem.id))
    sdItem.update(data=sd)
    print('Overwriting existing feature service..')
    fs = sdItem.publish(overwrite=True)
```

```
    if shrOrg or shrEveryone or shrGroups:
        print('Setting sharing options..')
        fs.share(org=shrOrg, everyone=shrEveryone, groups=shrGroups)

    print('Finished updating: {} - ID: {}'.format(fs.title, fs.id))
```

Batch File Code:

```
cd "C:\Users\ZMJS5\AppData\Local\ESRI\conda\envs\my_arcgispro-py3"

"C:\Users\ZMJS5\AppData\Local\ESRI\conda\envs\my_arcgispro-py3\python.exe"
"E:\Grad_School\Archive_2\Thesis\State_Climate_Office\TCO_Automation\NDFD_reducedpul
l_Conus_only_Py3_Rev_3.py"

"C:\Users\ZMJS5\AppData\Local\ESRI\conda\envs\my_arcgispro-py3\python.exe"
"E:\Grad_School\Archive_2\Thesis\State_Climate_Office\TCO_Automation\GRIB_to_VectorTi
le5_resample_rev4_TNGIC.py"

"C:\Users\ZMJS5\AppData\Local\ESRI\conda\envs\my_arcgispro-py3\python.exe"
"E:\Grad_School\Archive_2\Thesis\State_Climate_Office\TCO_Automation\HazardIndex_Vect
orTile7_1302019_TNGIC.py"
```

"C:\Users\ZMJS5\AppData\Local\ESRI\conda\envs\my_arcgispro-py3\python.exe"
"E:\Grad_School\Archive_2\Thesis\State_Climate_Office\TCO_Automation\Autopost_jupyterty
ped2_NewNDFD.py"

Appendix D: Former Scripts

Former Script for Automation of Daily Storm Reports:

```
import arcpy
import csv
try:
    import urllib
    import urllib2
except:
    import urllib.request
    import urllib.error
import os
import time
from datetime import timedelta
from datetime import datetime
#import requests


print("Time to make the chimichagas")


def Avalon(Tango, Stump, Time, F_Scale, Location, County, State, Lat, Lon, Comments):
    with arcpy.da.InsertCursor(Stump, ("SHAPE@XY", Time, F_Scale, Location, County, State,
Lat, Lon, Comments))as wizard:
        for players in Tango:
            Timez = players[0]
```

```
        TimeInt2 = int(Timez)
        F_Scalez = players[1]
        Countyz = players[3]
        State = players[4]
        Latz = players[5]
        numLat = float(Latz)
        Lonz = players[6]
        numLon = float(Lonz)
        Commentsz = players[7]
        Locationz = players[2]
        print(players)
        wizard.insertRow(((float(numLon), float(numLat)), TimeInt2, F_Scalez, Locationz,
Countyz, State, numLat, numLon, Commentsz))

def Oz(FileName,stump,Time,F_Scale, Location, County, State, Lat, Lon, Comments):
    arcpy.CreateFeatureclass_management(Placeholder_Spot, FileName, "POINT", "", "",
"",spatialref)
    arcpy.AddField_management(stump, Time, "SHORT")
    arcpy.AddField_management(stump, F_Scale, "TEXT")
    arcpy.AddField_management(stump, Location, "TEXT")
    arcpy.AddField_management(stump, County, "TEXT")
    arcpy.AddField_management(stump, State, "TEXT")
    arcpy.AddField_management(stump, Lat, "FLOAT")
```

```
    arcpy.AddField_management(stump, Lon, "FLOAT")
    arcpy.AddField_management(stump, Comments, "TEXT", "", "",fieldlength)



timestream = datetime.now() - timedelta(days= 1)
timestr = timestream.strftime("%m%d%y")

#Variables
WeatherReport = r"http://www.spc.noaa.gov/climo/reports/yesterday_torn.csv"
hailReport = r"http://www.spc.noaa.gov/climo/reports/yesterday_hail.csv"
windReport = r"http://www.spc.noaa.gov/climo/reports/yesterday_wind.csv"
Weathercsv = 'yesterday_torn.csv'
hailcsv = 'yesterday_hail.csv'
windcsv = 'yesterday_wind.csv'
#CloudCity = r"Y:\TCO\GIS_Data\Monthly_Precip_Obs\2018_03\files_StormReportMap"
CloudCity =
r"E:\Grad_School\Archive_2\Thesis\State_Climate_Office\NOAA_Daily\Test_Data"
Lando = os.path.join(str(CloudCity), "StormReportMap" + "_" + timestr +".pdf")
TN_border =
r'E:\Grad_School\Archive_2\Thesis\State_Climate_Office\NOAA_Daily\TN_Border\DataShoop
_TNBorder_Lab.shp'
```

```
spatialsurogate =
r"E:\Grad_School\Archive_2\IndependantStudyPython\Lesson4\Project_data\SpatialReference.s
hp"
spatialref = arcpy.Describe(spatialsurogate).spatialReference

Placeholder_Spot =
r"E:\Grad_School\Archive_2\Thesis\State_Climate_Office\NOAA_Daily\Test_Data"
arcpy.env.workspace = Placeholder_Spot
arcpy.env.overwriteOutput = True




adventure = urllib2.urlopen (WeatherReport)
#wildstory = open(Weathercsv, 'wb')
#wildstory.write(adventure.read())
#wildstory.close
print (adventure)



halo = urllib2.urlopen (hailReport)
#masterchief = open(hailcsv, 'wb')
```

```
#masterchief.write(halo.read())

#masterchief.close

print (halo)



Djinn = urllib2.urlopen (windReport)

#Genie = open(windcsv, 'wb')

#Genie.write(Djinn.read())

#Genie.close

print (Djinn)



Story = [adventure,halo,Djinn]



csvreader = csv.reader(adventure, delimiter= ',')

print (csvreader)

csvreaderhalo = csv.reader(halo, delimiter= ',')

print (csvreaderhalo)

csvreaderDjinn = csv.reader(Djinn, delimiter= ',')
```

```
print (csvreaderDjinn)

cvss = [csvreader,csvreaderhalo,csvreaderDjinn]


header = csvreader.next()

print (header)

headerhalo = csvreaderhalo.next()

print (headerhalo)

headerDjinn = csvreaderDjinn.next()

print (headerDjinn)


#with open(Weathercsv, 'w') as f: f.write(adventure.read()) #This doesnt seem to load anything


print("csv stuff done")



fieldlength = 300


#Fields Tornado

Time = "Time"

F_Scale = "F_Scale"

Location = "Location"

County = "County"

State = "State"
```

```
Lat = "Lat"

Lon = "Lon"

Comments = "Comments"


#Fields Hail

TimeH = "Time"

Size = "Size"

LocationH = "Location"

CountyH = "County"

StateH = "State"

LatH = "Lat"

LonH = "Lon"

CommentsH = "Comments"


#Fields Wind

TimeW = "Time"

Speed = "Speed"

LocationW = "Location"

CountyW = "County"

StateW = "State"

LatW = "Lat"

LonW = "Lon"

CommentsW = "Comments"
```

```
Time1 = header.index("Time")

F_Scale1 = header.index("F_Scale")

Location1 = header.index("Location")

County1 = header.index("County")

State1 = header.index("State")

Lat1 = header.index("Lat")

Lon1 = header.index("Lon")

Comments1 = header.index("Comments")


Time2 = headerhalo.index("Time")

F_Scale2 = headerhalo.index("Size")

Location2 = headerhalo.index("Location")

County2 = headerhalo.index("County")

State2 = headerhalo.index("State")

Lat2 = headerhalo.index("Lat")

Lon2 = headerhalo.index("Lon")

Comments2 = headerhalo.index("Comments")


Time3 = headerDjinn.index("Time")

F_Scale3 = headerDjinn.index("Speed")

Location3 = headerDjinn.index("Location")
```

```
County3 = headerDjinn.index("County")

State3 = headerDjinn.index("State")

Lat3 = headerDjinn.index("Lat")

Lon3 = headerDjinn.index("Lon")

Comments3 = headerDjinn.index("Comments")



#Naming Shapefiles and excel

FileNameT = "YesterdayTorn" + ".shp"

FileNameH = "YesterdayHail" + ".shp"

FileNameW = "YesterdayWind" + ".shp"

ShortNameT = "YesterdayTorn" + timestr + ".shp"

ShortNameH = "YesterdayHail" + timestr + ".shp"

ShortNameW = "YesterdayWind" + timestr + ".shp"



#csvtpath = os.path.join(str(Placeholder_Spot), csvT)

#csvhpath = os.path.join(str(Placeholder_Spot), csvH)

#csvwpath = os.path.join(str(Placeholder_Spot), csvW)



#Pathing Data
```

```
stumpT = os.path.join(str(Placeholder_Spot), FileNameT)
stumpH = os.path.join(str(Placeholder_Spot), FileNameH)
stumpW = os.path.join(str(Placeholder_Spot), FileNameW)

branchT = os.path.join(str(CloudCity), ShortNameT)
branchH = os.path.join(str(CloudCity), ShortNameH)
branchW = os.path.join(str(CloudCity), ShortNameW)

superstump = [stumpT, stumpH, stumpW]
#superbranch = [branchT, branchH, branchW]



#Oz module
Oz(FileNameT,stumpT,Time,F_Scale, Location, County, State, Lat, Lon, Comments)
print ("oz1")
Oz(FileNameH,stumpH,TimeH,Size, LocationH, CountyH, StateH, LatH, LonH, CommentsH)
print ("oz2")
Oz(FileNameW,stumpW,TimeW,Speed, LocationW, CountyW, StateW, LatW, LonW,
CommentsW)
print("oz complete")
```

```
#Failed Avalon module
#Avalon(csvreader, stumpT, Time,F_Scale, Location, County, State, Lat, Lon, Comments)
#print("Avalon1")
#Avalon(csvreaderhalo, stumpH, TimeH, Size, LocationH, CountyH, StateH, LatH, LonH,
CommentsH)
#print("Avalon2")
#Avalon(csvreaderDjinn, stumpW, TimeW, Speed, LocationW, CountyW, StateW, LatW,
LonW, CommentsW)



#manual Avalon run
with arcpy.da.InsertCursor(stumpT, ("SHAPE@XY", Time, F_Scale, Location, County, State,
Lat, Lon, Comments))as wizard:
    for players in csvreader:
        Timez = players[0]
        TimeInt2 = int(Timez)
        F_Scalez = players[1]
        Countyz = players[3]
        State = players[4]
        Latz = players[5]
        numLat = float(Latz)
        Lonz = players[6]
        numLon = float(Lonz)
```

```
        Commentsz = players[7]
        Locationz = players[2]
        print(players)
        wizard.insertRow(((float(numLon), float(numLat)), TimeInt2, F_Scalez, Locationz,
Countyz, State, numLat, numLon, Commentsz))

print("Avalon1")

with arcpy.da.InsertCursor(stumpH, ("SHAPE@XY", TimeH, Size, LocationH, CountyH,
StateH, LatH, LonH, CommentsH))as Warlock:
    for players in csvreaderhalo:
        Timez = players[0]
        TimeInt2 = int(Timez)
        F_Scalez = players[1]
        Countyz = players[3]
        State = players[4]
        Latz = players[5]
        numLat = float(Latz)
        Lonz = players[6]
        numLon = float(Lonz)
        Commentsz = players[7]
        Locationz = players[2]
        print(players)
```

```
        Warlock.insertRow(((float(numLon), float(numLat)), TimeInt2, F_Scalez, Locationz,
Countyz, State, numLat, numLon, Commentsz))

print("Avalon2")

with arcpy.da.InsertCursor(stumpW, ("SHAPE@XY", TimeW, Speed, LocationW, CountyW,
StateW, LatW, LonW, CommentsW))as Mystic:
    for players in csvreaderDjinn:
        Timez = players[0]
        TimeInt2 = int(Timez)
        F_Scalez = players[1]
        Countyz = players[3]
        State = players[4]
        Latz = players[5]
        numLat = float(Latz)
        Lonz = players[6]
        numLon = float(Lonz)
        Commentsz = players[7]
        Locationz = players[2]
        print(players)
        Mystic.insertRow(((float(numLon), float(numLat)), TimeInt2, F_Scalez, Locationz,
Countyz, State, numLat, numLon, Commentsz))
```

```
print("Avalon complete")



#Copies for mxd
#arcpy.CopyFeatures_management(stumpT, branchT)
#arcpy.CopyFeatures_management(stumpH, branchH)
#arcpy.CopyFeatures_management(stumpW, branchW)

mxd =
arcpy.mapping.MapDocument(r"E:\Grad_School\Archive_2\IndependantStudyPython\Final_Pro
ject\Storm_Report_Map.mxd")
#arcpy.mapping.ExportToPDF(mxd,Lando)

#Round about way to build csv from csv pull since dont have permissions
#arcpy.TableToExcel_conversion(stumpT, csvtpath)
#arcpy.TableToExcel_conversion(stumpH, csvhpath)
#arcpy.TableToExcel_conversion(stumpW, csvwpath)

tester = arcpy.ListFeatureClasses()

for ndated in tester:
    datedloc = Placeholder_Spot + '\\' + ndated
    stripped = ndated.replace('.shp', '')
```

```
    clipped = Placeholder_Spot + '\\' + stripped + '_TNclipped.shp'
    if ndated.startswith('Yesterday'):
        arcpy.analysis.Clip(datedloc, TN_border, clipped)
    else:
        pass
print("and that is how you do it")
```

Former Script for Downloading data from the NDFD website script:

```
import urllib
import urllib2
import requests
import shutil
import os
import time
import arcpy

print ("imports successful")



timestr = time.strftime("%m%d%y")

ndfd_dsapt = 'ds.apt.bin'
ndfd_dsconhazo = 'ds.conhazo.bin'
```

ndfd_dscritfireo = 'ds.critfireo.bin'

ndfd_dsdryfireo = 'ds.dryfireo.bin'

ndfd_dsiceaccum = 'ds.iceaccum.bin'

ndfd_dsmaxrh = 'ds.maxrh.bin'

ndfd_dsmaxt = 'ds.maxt.bin'

ndfd_dsminrh = 'ds.minrh.bin'

ndfd_dsmint = 'ds.mint.bin'

ndfd_dsphail = 'ds.phail.bin'

ndfd_dspop12 = 'ds.pop12.bin'

ndfd_dsptornado = 'ds.ptornado.bin'

ndfd_dsptotsvrtstm = 'ds.ptotsvrtstm.bin'

ndfd_dsptotxsvrtstm = 'ds.ptotxsvrtstm.bin'

ndfd_dsptstmwinds = 'ds.ptstmwinds.bin'

ndfd_dspxhail = 'ds.pxhail.bin'

ndfd_dspxtornado = 'ds.pxtornado.bin'

ndfd_dspxtstmwinds = 'ds.pxtstmwinds.bin'

ndfd_qpf = 'ds.qpf.bin'

ndfd_dsrhm = 'ds.rhm.bin'

ndfd_dssky = 'ds.sky.bin'

ndfd_dssnow = 'ds.snow.bin'

ndfd_dstcwspdabv34c = 'ds.tcwspdabv34c.bin'

ndfd_dstcwspdabv34i = 'ds.tcwspdabv34i.bin'

ndfd_dstcwspdabv50c = 'ds.tcwspdabv50c.bin'

```
ndfd_dstcwspdabv50i = 'ds.tcwspdabv50i.bin'

ndfd_dstcwspdabv64c = 'ds.tcwspdabv64c.bin'

ndfd_dstcwspdabv64i = 'ds.tcwspdabv64i.bin'

ndfd_dstd = 'ds.td.bin'

ndfd_dstemp = 'ds.temp.bin'

ndfd_dswaveh = 'ds.waveh.bin'

ndfd_dswdir = 'ds.wdir.bin'

ndfd_dswgust = 'ds.wgust.bin'

ndfd_dswspd = 'ds.wspd.bin'

ndfd_dswwa = 'ds.wwa.bin'

ndfd_dswx = 'ds.wx.bin'

ndfd_lsl = 'ls-l'

ndfd_lslt = 'ls-lt'


ndfd_midAt =

r'http://tgftp.nws.noaa.gov/SL.us008001/ST.opnl/DF.gr2/DC.ndfd/AR.midatlan/VP.001-003'

ndfd_crmissvy =

r'http://tgftp.nws.noaa.gov/SL.us008001/ST.opnl/DF.gr2/DC.ndfd/AR.crmissvy/VP.001-003'

ndfd_conus =

r'http://tgftp.nws.noaa.gov/SL.us008001/ST.opnl/DF.gr2/DC.ndfd/AR.conus/VP.001-003'

#continuous US


#for stuff in os.listdir(ndfd):
```

```
    #print (stuff)
    #dirlist.append(stuff)
#print(dirlist)

regionlist = [ndfd_crmissvy, ndfd_midAt]


test_site = r'E://Grad_School//Archive_2//Thesis//State_Climate_Office//Bin_Test_Site'
test_site2 =
r'E://Grad_School//Archive_2//Thesis//State_Climate_Office//Bin_Test_Site//crmissvy'

print("destinations correct")

superlist = [ndfd_dsapt, ndfd_dsconhazo, ndfd_dscritfireo, ndfd_dsdryfireo, ndfd_dsiceaccum,
ndfd_dsmaxrh, ndfd_dsmaxt, ndfd_dsminrh, ndfd_dsmint, ndfd_dsphail, ndfd_dspop12,
ndfd_dsptornado, ndfd_dsptotsvrtstm, ndfd_dsptotxsvrtstm, ndfd_dsptstmwinds, ndfd_dspxhail,
ndfd_dspxtornado, ndfd_dspxtstmwinds, ndfd_qpf, ndfd_dsrhm, ndfd_dssky, ndfd_dssnow,
ndfd_dstcwspdabv34c, ndfd_dstcwspdabv34i, ndfd_dstcwspdabv50c, ndfd_dstcwspdabv50i,
ndfd_dstcwspdabv64c, ndfd_dstd, ndfd_dstemp, ndfd_dswaveh, ndfd_dswdir, ndfd_dswgust,
ndfd_dswspd, ndfd_dswwa, ndfd_dswx, ndfd_lsl, ndfd_lslt]
```

```
ndfd_crmissvyf =
'E://Grad_School//Archive_2//Thesis//State_Climate_Office//Bin_Test_Site//ndfd_crmissvy' +
timestr
ndfd_midAtf =
'E://Grad_School//Archive_2//Thesis//State_Climate_Office//Bin_Test_Site//ndfd_midAt' +
timestr
try:
    os.mkdir(ndfd_crmissvyf)
except:
    pass #already exists
try:
    os.mkdir(ndfd_midAtf)
except OSError:
    pass #already exists
try:
    arcpy.CreateFileGDB_management(str(ndfd_crmissvyf), 'ndfd_crmissvy' + timestr)
except:
    pass #already exists
try:
    arcpy.CreateFileGDB_management(str(ndfd_midAtf), 'ndfd_midAtf' + timestr)
except:
    pass #already exists
```

```
for region in regionlist:
    if region == ndfd_crmissvy:
        arcpy.env.workspace = ndfd_crmissvyf
    else:
        arcpy.env.workspace = ndfd_midAtf

    for load in superlist:
        stringCoble = str(ndfd_midAt + '/' + str(load))

        print (stringCoble)
        filer = urllib2.Request(stringCoble)

        responseapt = urllib2.urlopen(filer)
        print("this is were im telling it to go", responseapt)
        print("were we actually go", responseapt.geturl())
        workspace = test_site2
        output = open(load, 'wb')
        output.write(responseapt.read())
        output.close

        #renamer = str(strip) + timestr + '.bin'
    for stuff in os.listdir(test_site):
        if stuff.endswith('.bin'):
```

```
            first_name = str(test_site + '//' + str(stuff))
        #strip = load.replace('.bin', '')
            if region == ndfd_crmissvy:
                namer = 'ndfd_crmissvy' + timestr
                gdb = 'ndfd_crmissvy' + timestr + '.gdb'
            else:
                namer = 'ndfd_midAt' + timestr
                gdb = 'ndfd_midAt' + timestr + '.gdb'

            final_name = str(test_site + '//' + namer + '//' + stuff)
            shutil.copy(first_name, final_name)
        else:
            pass
        #arcpy.management.CopyRaster(first_name, final_name, None, None, -1.797693e+308,
"NONE", "NONE", None, "NONE", "NONE", "GRID", "NONE")

print('hahahahaha yes!')
```

Former Script for GRIB2 data converted to vector polygons script:

```
import arcpy
import os
import time
```

```
from arcpy.sa import*
arcpy.CheckOutExtension("Spatial")
import gdal

homefile = r'E://Grad_School//Archive_2//Thesis//State_Climate_Office//Final scripts'
boundary = r'E://Grad_School//Archive_2//Thesis//State_Climate_Office//Final
scripts//Transfer_file.gdb//TNBoundary'
transfer_file = r'E://Grad_School//Archive_2//Thesis//State_Climate_Office//Final
scripts//Transfer_file.gdb'
tifflist = []
mergeraster = []
clippedgroup = []
singlegroup = []

binlist = ['ds.temp.bin', 'ds.snow.bin', 'ds.wgust.bin',
'ds.ptornado.bin','ds.iceaccum.bin','ds.phail.bin']
timestr = time.strftime("%m%d%y")
binpaths = r'E://Grad_School//Archive_2//Thesis//State_Climate_Office//Final
scripts//ndfd_conus' + timestr
bombsite = homefile + '//' + 'NDFD_Daily_Rasters_' + timestr
print('imports and variables setup')

try:
```

```
    os.mkdir(bombsite)
    print('folder created')
except:
    print('file already made')


arcpy.env.workspace = bombsite
arcpy.env.overwriteOutput = True



for weather in binlist:
    ndfd_data = binpaths + '//' + weather
    if weather == 'ds.wgust.bin':
            forcast = 'wg_'
    elif weather == 'ds.temp.bin':
            forcast = 'tp_'
    elif weather == 'ds.snow.bin':
            forcast = 'sn_'
    elif weather == 'ds.iceaccum.bin':
            forcast = 'ic_'
    elif weather == 'ds.ptornado.bin':
            forcast = 'to_'
    elif weather == 'ds.phail.bin':
            forcast = 'ha_'
```

```
    #elif weather == 'ds.qpf.bin':
            #forcast = 'ra_'
    gdalopen = gdal.Open(ndfd_data, gdal.GA_ReadOnly)
    numero = gdalopen.RasterCount
    print(numero)
    numa = 0
    merge_name = forcast + '.tif'
#look into putting clip before raster to ascii

    if numero <= 1:
        print('only one record in ' + weather)
        subout = forcast + str(numa)
        suboutpath = bombsite + '//' + subout
        castle = arcpy.Describe(ndfd_data)
        keep = castle.spatialReference
        print(subout + ' created')
        asciioutput = bombsite + '//' + forcast + str(numa) + '_ascii.txt'
        arcpy.RasterToASCII_conversion(ndfd_data, asciioutput)
        print('raster to ascii worked')
        tiffnopath = forcast + str(numa) + '_ti.tif'
        tiffoutput = str(bombsite) + '//' + tiffnopath
        arcpy.ASCIIToRaster_conversion(asciioutput, tiffoutput)
        print('ascii to raster worked')
```

```
    arcpy.DefineProjection_management(tiffoutput, keep)
    print(tiffnopath + ' proj. defined')
    singlegroup.append(tiffnopath)

else:
    for num in range(numero):
        print(numa)
        subout = forcast + str(numa)
        suboutpath = bombsite + '//' + subout
        arcpy.ExtractSubDataset_management(ndfd_data, suboutpath, numa)
        castle = arcpy.Describe(suboutpath)
        keep = castle.spatialReference
        print('subset ' + subout + ' created')
        asciioutput = bombsite + '//' + forcast + str(numa) + '_ascii.txt'
        arcpy.RasterToASCII_conversion(suboutpath, asciioutput)
        print('raster to ascii worked')
        tiffnopath = forcast + str(numa) + '_ti.tif'
        tiffoutput = str(bombsite) + '//' + tiffnopath
        tifflist.append(tiffoutput)
        arcpy.ASCIIToRaster_conversion(asciioutput, tiffoutput, "FLOAT")
        print('ascii to raster worked')
        arcpy.DefineProjection_management(tiffoutput, keep)
        print(tiffnopath + ' proj. defined')
```

```
        numa += 1
    print('while subsets for ' + weather + ' worked')




    if weather == 'ds.temp.bin':
        m_temp_min = 'temp_min.tif'
        m_temp_max = 'temp_max.tif'
        try:
            arcpy.MosaicToNewRaster_management(tifflist, bombsite, m_temp_max, '#',
'32_BIT_FLOAT', '#', '1', 'MAXIMUM', 'FIRST')
            print('temp minimum done')
            arcpy.MosaicToNewRaster_management(tifflist, bombsite, m_temp_min, '#',
'32_BIT_FLOAT', '#', '1', 'MINIMUM', 'FIRST')
            print('temp maximum done')
            mergeraster.append(m_temp_min)
            mergeraster.append(m_temp_max)
        except:
            print('ds file failed')


    elif weather == 'ds.wgust.bin':
        try:
```

```
            arcpy.MosaicToNewRaster_management(tifflist, bombsite, merge_name, '#',
'32_BIT_FLOAT', '#', '1', 'MAXIMUM', 'FIRST')
            mergeraster.append(merge_name)
            print(merge_name + ' merged using maximum')
            print('wgust merged')
        except:
            print('ds file failed')
    else:
        arcpy.MosaicToNewRaster_management(tifflist, bombsite, merge_name, '#',
'32_BIT_FLOAT', '#', '1', 'SUM', 'FIRST')
        mergeraster.append(merge_name)
        print(merge_name + ' mergered using sum')



    del tifflist[:]
    print(tifflist)
print('giant mergers done')
print('now for general cleaning up')
print('......')
print('......')
print(mergeraster)
```

```
for TN in mergeraster:
    TN_raster = bombsite + '//' + TN
    TN_strip = TN.replace('.tif', '')
    TN_clipped = TN_strip + '_TN.tif'
    TN_clippedfull = bombsite + '//' + TN_clipped
    #arcpy.management.Clip(TN_raster, "427735.804551858 1124048.01885298
1233262.83569153 1367916.69433694", TN_clippedfull, boundary)
    arcpy.env.snapRaster = boundary
    remask = arcpy.sa.ExtractByMask(TN_raster, boundary)
    remask.save(TN_clippedfull)
    print('clipped ' + TN)
    #Kelvin to Fehr
    if TN.startswith('temp_min') or TN.startswith('temp_max'):
        fullblister = bombsite + '//' + TN_clipped
        fehr = Raster(fullblister)
        heit = (fehr-273.15) * (9/5) + 32
        notiff = TN_clipped.replace('.tif', '')
        notifffehr = notiff + 'fehr.tif'
        addfehr = bombsite + '//' + notifffehr
        heit.save(addfehr)
        clippedgroup.append(notifffehr)
        print(TN_clipped + ' converted to F')
    #kg/m2 to inches
```

```
elif TN_clipped.startswith('ic_'):
    icetransfer = bombsite + '//' + TN_clipped
    frozen = Raster(icetransfer)
    iceconvert = frozen * 0.039370
    icestrip = TN_clipped.replace('.tif', '')
    notiffice = icestrip + 'inch.tif'
    addice = bombsite + '//' + notiffice
    iceconvert.save(addice)
    clippedgroup.append(notiffice)
    print(TN_clipped + ' converted to inches from kgm2')
#meters to inches
elif TN_clipped.startswith('sn_'):
    snowtransfer = bombsite + '//' + TN_clipped
    snowman = Raster(snowtransfer)
    snowconvert = snowman * 39.370
    snowstrip = TN_clipped.replace('.tif', '')
    notiffsnow = snowstrip + 'inch.tif'
    addsnow = bombsite + '//' + notiffsnow
    snowconvert.save(addsnow)
    clippedgroup.append(notiffsnow)
    print(TN_clipped + ' converted from meters to inches')
#ms-1 to mph
elif TN_clipped.startswith('wg_'):
```

```
        windtransfer = bombsite + '//' + TN_clipped
        windgust = Raster(windtransfer)
        windconvert = windgust * 2.236936271
        windstrip = TN_clipped.replace('.tif', '')
        notiffwind = windstrip + 'mph.tif'
        addwind = bombsite + '//' + notiffwind
        windconvert.save(addwind)
        clippedgroup.append(notiffwind)
        print(TN_clipped + ' converted from ms-1 to mph')
    else:
        clippedgroup.append(TN_clipped)
        print(TN_clipped + 'clipped to TN boundary')
print('......')
print('......')

fehrclip = bombsite + '//' + 'temp_max_TNfehr.tif'
arcpy.env.snapRaster = fehrclip
for single in singlegroup:
    singlestrip = single.replace('.tif', '')
    resampler = singlestrip + 'clipp.tif'
    singleoutput = bombsite + '//' + resampler
    singlepath = bombsite + '//' + single
    singlemask = arcpy.sa.ExtractByMask(singlepath, fehrclip)
```

```
    print(single + ' extracted by mask')
    singlemask.save(singleoutput)
    clippedgroup.append(resampler)
print('singles now are ready to convert with multi rasters')


arcpy.env.workspace = transfer_file
arcpy.env.overwriteOutput = True

for clgr in clippedgroup:
    TN_clippedfull = bombsite + '//' + clgr
    TN_point_strip = clgr.replace('.tif', '')
    TN_point_path = transfer_file + '//' + TN_point_strip

    arcpy.conversion.RasterToPoint(TN_clippedfull, TN_point_path, "Value")
    print(TN_point_strip + ' done')

    TN_poly = TN_point_strip + 'poly'
    TN_poly_path = transfer_file + '//' + TN_poly
    arcpy.analysis.CreateThiessenPolygons(TN_point_path, TN_poly_path, "ALL")
    print(TN_poly + ' done')

    TN_polyclip = TN_point_strip + 'polyclip'
```

```
    TN_polyclip_path = transfer_file + '//' + TN_polyclip
    arcpy.analysis.Clip(TN_poly_path, boundary, TN_polyclip_path, None)
    print(TN_polyclip + ' done')



arcpy.CheckInExtension("Spatial")
print('all done with raster conversions and editing :)')
```

Former Script for Creating a Hazard Index Script:

```
import arcpy

origin = r'E:\Grad_School\Archive_2\Thesis\State_Climate_Office\Final
scripts\Transfer_file.gdb'
weatherlist = ['ha_0_ticlipppolyclip', 'ic__TNinchpolyclip', 'sn__TNinchpolyclip',
'temp_max_TNfehrpolyclip', 'temp_min_TNfehrpolyclip', 'to_0_ticlipppolyclip',
'wg__TNmphpolyclip']
intersectlist = []

arcpy.env.workspace = origin
arcpy.env.overwriteOutput = True
print('modules and variables done')
```

```
for weather in weatherlist:
    path = origin + '\\' + weather
    intersectlist.append(path)
    if weather.startswith('temp'):
        namestrip = weather.replace('_TNfehrpolyclip', '')
    elif weather.endswith('_ticlipppolyclip'):
        namestrip = weather.replace('_ticlipppolyclip', '')
    elif weather.endswith('_TNinchpolyclip'):
        namestrip = weather.replace('_TNinchpolyclip', '')
    elif weather.endswith('_TNmphpolyclip'):
        namestrip = weather.replace('_TNmphpolyclip', '')
    gridname = namestrip + '_grid'
    arcpy.management.AddField(path, gridname, 'DOUBLE')
    arcpy.management.CalculateField(path, gridname, '!grid_code!', 'PYTHON3')
    print(weather + ' has field' + gridname + ' added')
print('weatherlist done')


intersect_vector = origin + '\\' + 'hdfd_HI_data'
arcpy.analysis.Intersect(intersectlist, intersect_vector, 'ALL', None, 'INPUT')
print('Intersect added')
arcpy.management.AddField(intersect_vector, 'haz_index', 'TEXT')
arcpy.env.workspace = intersect_vector
```

```
hdfd_HI_datafields = arcpy.ListFields(intersect_vector)
print('fields listed and haz_index field added')

deletefields = ['FID_ha_0_ticlipppolyclip', 'Input_FID, pointid, grid_code',
'FID_ha_0_ticlipppolyclip_1', 'Input_FID_1', 'pointid_1', 'grid_code_1', 'ha_0_grid_1',
'FID_ic__TNinchpolyclip',
        'Input_FID_12', 'pointid_12', 'grid_code_12', 'FID_sn__TNinchpolyclip',
'Input_FID_12_13', 'pointid_12_13', 'grid_code_12_13', 'FID_temp_max_TNfehrpolyclip',
'Input_FID_12_13_14',
        'pointid_12_13_14', 'grid_code_12_13_14', 'FID_temp_min_TNfehrpolyclip',
'Input_FID_12_13_14_15', 'pointid_12_13_14_15', 'grid_code_12_13_14_15',
'FID_to_0_ticlipppolyclip',
        'Input_FID_12_13_14_15_16', 'pointid_12_13_14_15_16',
'grid_code_12_13_14_15_16', 'FID_wg__TNmphpolyclip', 'Input_FID_12_13_14_15_16_17',
'pointid_12_13_14_15_16_17',
        'grid_code_12_13_14_15_16_17', 'Input_FID', 'pointid', 'grid_code']


arcpy.management.DeleteField(intersect_vector, deletefields)
print('fields deleted')

hazardindexfield = 'Haz_Index'
```

```
#SQL statements for selecting hazard index data
lowhitemp = '"temp_max_grid" BETWEEN 100 AND 104.999'
midhitemp = '"temp_max_grid" BETWEEN 105 AND 114.999'
highhitemp = '"temp_max_grid" >= 115'

lowlowtemp = '"temp_min_grid" BETWEEN 0 AND -9.999'
midlowtemp = '"temp_min_grid" BETWEEN -10 AND -19.999'
highlowtemp = '"temp_min_grid" <= -20'

lowwgust = '"wg__grid" BETWEEN 40 AND 49.999'
midwgust = '"wg__grid" BETWEEN 50 AND 59.999'
highwgust = '"wg__grid" >= 60'

#lowrainfall = '"ra__grid" BETWEEN 1 and 2'
#midrainfall = '"ra__grid" BETWEEN 3 and 4'
#highrainfall = '"ra__grid" >= 5'

lowsnowfall = '"sn__grid" BETWEEN 2 AND 3.999'
midsnowfall = '"sn__grid" BETWEEN 4 AND 7.999'
highsnowfall = '"sn__grid" >= 8'

lowiceaccum = '"ic__grid" BETWEEN 0.01 AND 0.099'
```

```
midiceaccum = '"ic__grid" BETWEEN 0.1 AND 0.249'

highiceaccum = '"ic__grid" >= 0.25'


tornado = 'NOT "to_0_grid" = 0'

hail = 'NOT "ha_0_grid" = 0'


Nada = 'haz_index IS NULL'



#Start selection and attribute generation

lowlist = [lowhitemp, lowlowtemp, lowwgust, lowsnowfall, lowiceaccum]

midlist = [midhitemp, midlowtemp, midwgust, midsnowfall, midiceaccum]

highlist = [highhitemp, highhitemp, highwgust, highsnowfall, highiceaccum, tornado, hail]


arcpy.MakeFeatureLayer_management(intersect_vector, 'hazardindex')


arcpy.SelectLayerByAttribute_management('hazardindex', 'NEW_SELECTION', Nada)

arcpy.management.CalculateField('hazardindex', 'haz_index', '"no risk"', 'PYTHON3')

print('none calculated for haz_index')


for low in lowlist:

    arcpy.SelectLayerByAttribute_management('hazardindex', 'NEW_SELECTION', low)

    arcpy.management.CalculateField('hazardindex', 'haz_index', '"low"', 'PYTHON3')
```

```
    print(low + ' calculated for haz_index')
for mid in midlist:
    arcpy.SelectLayerByAttribute_management('hazardindex', 'NEW_SELECTION', mid)
    arcpy.management.CalculateField('hazardindex', 'haz_index', '"medium"', 'PYTHON3')
    print(mid + ' calculated for haz_index')
for high in highlist:
    arcpy.SelectLayerByAttribute_management('hazardindex', 'NEW_SELECTION', high)
    arcpy.management.CalculateField('hazardindex', 'haz_index', '"high"', 'PYTHON3')
    print(high + ' calculated for haz_index')

arcpy.Delete_management("hazardindex")
print('"your a nerd" written by my 12yr old nephew')
```

VITA

MICHAEL SHOOP

Education:	Public Schools, Kingsport, TN
	B.S. Geography, East Tennessee State University, Johnson City, TN, 2001
	M.S. Geosciences, Geospatial Analysis Concentration, East Tennessee State University, Johnson City, TN 2019
Professional Experience:	Lead GIS Analyst, Bechtel, Infrastructure, Frederick, MD April 2015-March 2017
	Lead GIS Analyst, Bechtel Western Australia pty ltd, Perth, WA January 2012-April 2015
	GIS Analyst, Bechtel, Oil Gas & Chemicals, Houston, TX November 2008-January 2012
	GIS Technician, Harris County Flood Control District, Houston, TX July 2008-October 2008
	GIS Analyst, EnSoCo Inc., Houston, TX September 2006-May 2008
Presentations:	East TNGIC Conference, Kingsport, TN 2018
	NETGIS meeting, Kingsport, TN 2019
	TNGIC conference, Chattanooga, TN 2019
	AASC conference, Santa Rosa, CA 2019