Automated Processing of USGS Daily Discharge Data: Downloading, Metadata Retrieving, and Trend Analysis

Ruoyu Zhang, Yibo Wang, Hyunglok Kim, Keyu Chen, and Yusheng Jiang

Group 2

1 Introduction

The stream discharge is one of the most valuable hydrologic data that help people to learn how water in streams behaves differently among various regions and how the availability of water changes through time. The USGS keeps updating streamflow at thousands of gages in the U.S. and provides the data for everyone to use, However, the current practice to obtain the data is not user-friendly. For example, users who study the floods in the past century at a gage will have to use the complex USGS website interface and repeatedly type start and end dates to download the data they need. If there are hundreds of floods to be studied, the total processes of downloading the data would be extremely tedious and time consuming.

In this project, we want to use Python to simplify this complex approach by web-scraping streamflow data automatically from the USGS website with as few as possible inputs so that users can easily download the data and switch the study period or gage automatically. Our deliverable is a Python package that users can simply request the data for any gage and time period with only the USGS gage ID as the required input. Once an object is created for the gage, we developed several methods for users to get basic metadata and statistics (i.e., min, median, max, mean) of streamflow at the gage, and users can also implement trend analysis of streamflow through Mann-Kendall Test (Mann, 1945). Using the package, we generated a look-up table for people to check the metadata and the trends/changes of water resources for gages recording streamflow over 10 years in the Chesapeake Bay watershed.

2 Data Description

The distribution of water resources is projected to shift due to climate change. Trenberth[1] (2011) has shown that we are likely to experience more frequent floods

because the likelihood of occurrence of extreme precipitation events would increase significantly. Urban areas are the most vulnerable regions to severe flood events. In a natural and forested environment, precipitation is stored in soil and then slowly released into the streams; In urban areas, most land is covered by impervious surfaces that prohibit the passage of water to store in soil, so most of water travels to the streams within a much shorter period of time than that in a forested environment. Such practices cause the water level to increase rapidly in urban areas and threaten people's lives and properties. At the same time, the probability of occurrence of prolonged droughts also increases due to climate change. Without proper water management strategies, we probably will not have sufficient water supply to meet the consumption of populated urban areas. Therefore, dense populations in urban areas will be under greater risk of experiencing hydrologic hazards, both floods and droughts, in the future due to climate change.

Before we start to predict the future, we need to know the water availability and behaviors in the past. The United States Geological Surveys (USGS) has started to monitor streamflow in major rivers in the United States before the 1900's and is continuing to expand the monitor network in recent decades and set new gages from small tributaries to other main streams among the United States. This provides us the opportunity to explore and learn water behaviors from the data itself.

In this project, we collected daily streamflow time-series for all gages in the Chesapeake Bay. The USGS has built 927 gages in the Bay, of which 625 have daily discharge records. Gages started to measure discharge at different dates, and some of them are no longer active or maintained by the USGS. We removed the no data records (USGS uses -999999 for no data value for discharge) from the raw data and performed analysis using cleaned data.

2.1 Experimental Design

We firstly developed two classes to obtain and store gage metadata and discharge time-series. The parent class is "USGS_Gage()", and the child class is "USGS_Data_Retrever()" (Figure 1). In the parent class, the metadata of a gage is obtained from the USGS website and saved to the object's constructor attributes. The constructor stores the USGS gage ID (the only required input parameter), variables measured in the gage in vars_info, and the geospatial information of the gage in geo_info. The vars_info and geo_info are empty unless the two methods are implemented. The first method, "getVarsMetaData()", web-scrapes all variables and their measurement periods at the gage from the USGS website. The other method, "getGeoMetaData()", obtains the county and state where the gage is located.

The child class, "USGS_Data_Retriever()" is more powerful than the parent class, which contains more methods to perform data downloading and plotting, statistic

summarizing, and trend analyzing for the streamflow data at the gage. Users have the flexibility to let the program download data for the full period (autoDates=True) or they can define their own desired time frame (pass dates to start and end dates and autoDates=False). They can also choose the unit of discharge data, either cubic foot or meter per second (cfs or cms) by setting input parameter “metric” to be True or False.

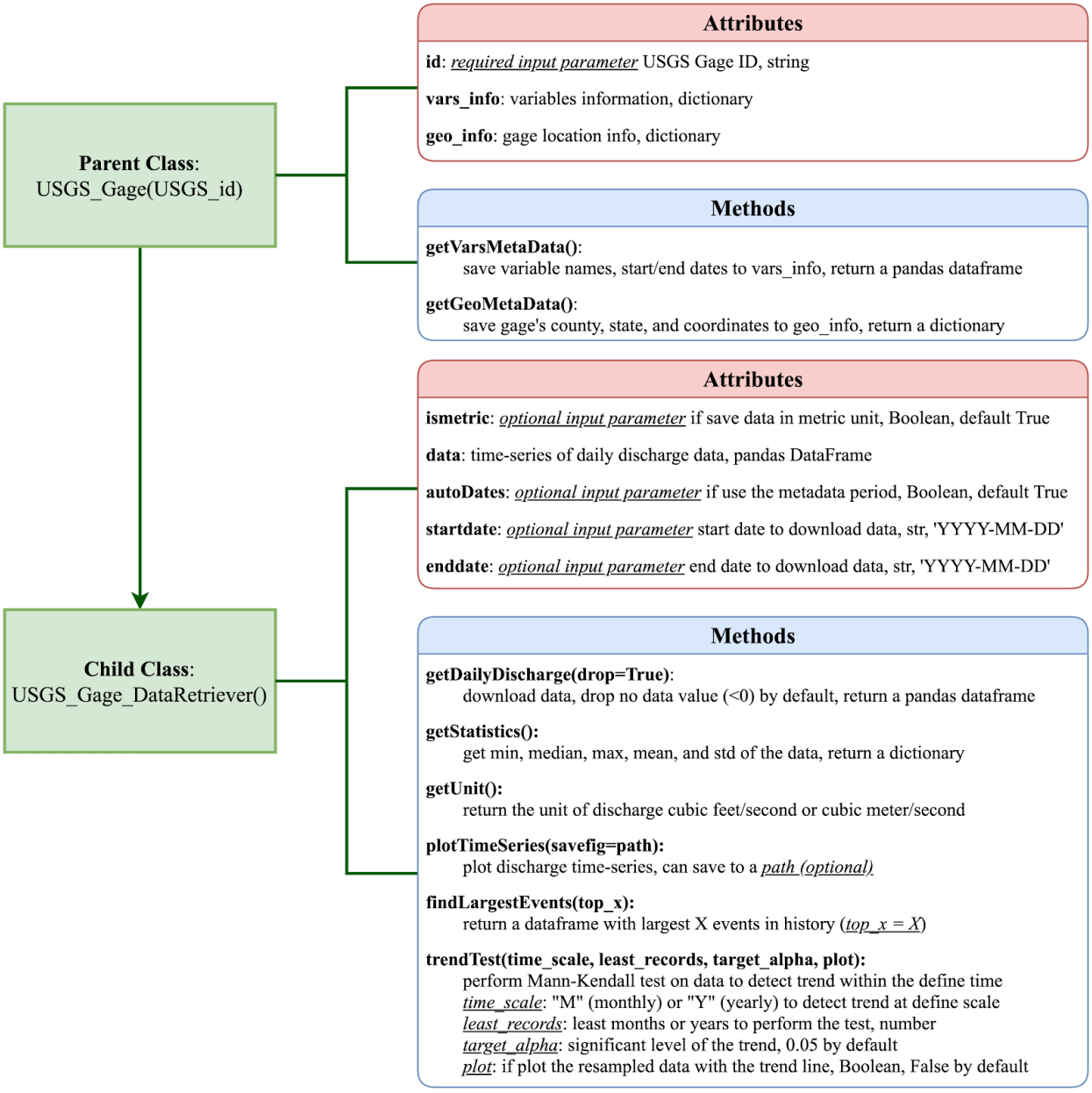

Figure 1. Structure of the USGS Data Retriever program.

Our data analysis uses the four main methods in the child class:

1. getDailyDischarge(drop=True)

This method downloads the discharge data for the user defined time period. The method returns a pandas data frame, and saves the frame in the object's data attribute. Discharge smaller than 0 is removed from the final data, unless the drop option is set to False.

2. getStatistics()
   This method obtains the minimum, median, maximum, mean, and standard deviation of discharges within the user-defined period. The results are saved in a dictionary, with keys Min, Median, Max, Mean, and Standard Deviation, in the object's stat attribute.
3. findLargestEvents(top_x):
   This method finds the largest X events in the discharge record at the gage (X is the number defined by the user). The returned results is a pandas data frame, including both dates and discharge quantities.
4. trendTest(time_scale, least_records, target_alpha=0.05, plot=False)
   This method enables users to test if there is a trend of discharge at monthly or yearly scale. Users need to define the time scale (either "M" for month or "Y" for year) and least months or years to perform the test in parameter "least_records". The trend significant level is set to be 0.05 by default. If users want to visualize the trend line, they can turn "plot" to True.
   The slope and p-value of the trend line will be returned. If data has less than defined months or years, null value will be returned.

Looping through all gages in the Chesapeake Bay, we created an object for each, downloaded the discharge data, performed the trend test, and saved the geospatial metadata, start and end dates of discharge, flow statistics, and slope and p-value from the trend test in a pandas data frame. We saved this data frame as a csv file as our look-table for each gage.

3 Methodology

3.1. Trend analysis

Monotonic trends in a target USGS data were calculated using the nonparametric Mann-Kendall test[2] (MKT). We tested monthly and yearly time-scale trends of the discharge of interest. The null hypothesis (H0) of the test assumed that there is no significant trend in the examined time series. First off, the statistical significance of the trends was tested by computing test statistics (S) from all the subsequent data values:

$$S = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \operatorname{sgn}(x_j - x_i)$$

where,

$$\operatorname{sgn}(x_j - x_i) = \begin{cases} +1, (x_j - x_i) > 0 \\ 0, (x_j - x_i) = 0 \\ -1, (x_j - x_i) < 0 \end{cases}$$

$$\operatorname{var}(S) = \frac{1}{18}\left[n(n-1)(2n+5) - \sum_{p=1}^{q} t_p(t_p - 1)(2t_p) + 5\right]$$

$$Z = \begin{cases} \frac{S-1}{\sqrt{\operatorname{var}(S)}}, \text{if S} > 0 \\ 0 \qquad\quad , \text{if S} = 0 \\ \frac{S+1}{\sqrt{\operatorname{var}(S)}}, \text{if S} < 0 \end{cases}$$

The statistic Z, when compared with the significant level, alpha, decides the presence or absence of a statistically significant trend. In this study, we set the significant level of 0.05 and we only considered the stations where the observation period is longer than 10 years. Second, besides finding the statistical significance of trend using MKT, the slope of the trend and confidence interval of the slope were estimated by a simple nonparametric procedure proposed by Sen[3] (1968). Sen's slope (also called Theil slope) estimator is the median of the slopes, Q(i,j), calculated from each pair of xi and xj as follows:

$$Q_{i,j} = \frac{x_i - x_j}{t_i - t_j}, t_i > \mathrm{t_j}$$

Where x_i and x_j are the observed discharges at time t_i and t_j. If there are n values of x in the time series, we get as many as n(n 1)/2 slope estimates for Qi,j (ti > tj).

We used a Python script for trend test of Man-Kendall and Sen's slope from following libraries:

- https://docs.scipy.org/doc/scipy-0.15.1/reference/generated/scipy.stats.mstats.theilslopes.html
- https://up-rs-esp.github.io/mkt/_modules/mkt.html

4 Beyond the original specifications

4.1 Web scraper

Given the scope of our project (hundreds of gages in the Chesapeake Bay area), it was almost impossible to manually download the data. Therefore, we developed **two web scrapers** to download the flow discharge data as well as their metadata. First, in order to automatically download data from hundreds of gages, we used the "urllib.request" package to visit their individual websites with gage ID being the only difference in the url address. The return was a JSON file for each gage, and we were able to locate the discharge data within nested dictionaries. Second, in order to enable user-interaction queries, we had to download metadata, namely county names and IDs, variables names, IDs and their start and end date, and longitude and latitude of the gages. However, this information couldn't be readily extracted from the JSON files, so we used the "beautifulsoup" package to scrape the information directly from the websites.

Once an object was created for the gage, we generated a look-up table for people to check the metadata and the trends/changes of water resources for gages recording streamflow over 10 years in the Chesapeake Bay watershed.

4.2 SCRUM activity

We were initially struggling with the project management because it was difficult for the five of us to communicate effectively. Therefore, we decided to adopt the **SCRUM** strategy for our project planning. We first brainstormed several user stories that we wanted our program to do, and then we split the tasks and finished the coding within 2 sprints (2 by 2 days). We found SCRUM quite useful because everyone knew clearly his/her tasks and the deadlines. We used the Trello website to split tasks both in coding

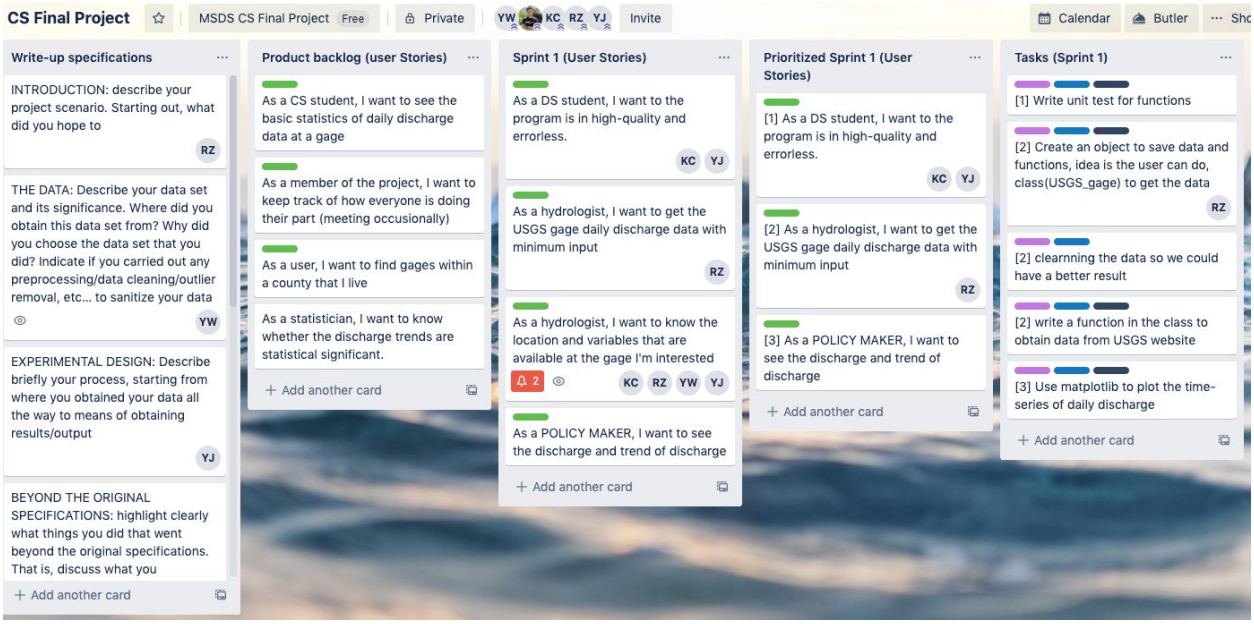

Figure 2. Trello dashboard that our group used to track the progress of this final project.

and in this write-up. This method significantly increased our efficiency and reduced anxiety since we could easily monitor how the entire project proceeded.

4.3 User-interaction queries

In order to visualize the change of discharge data, we created interactive maps that show the monthly and yearly changes of discharge data for each gage in Jupyter Notebook by Plotly. The users will have a clear understanding of the trends for each gage by visually inspecting the map, and can zoom in to a specific area of interest. If the users are interested in a specific gage or an area and want to have access to the raw data, they can click on the gages and easily query the data by either the county ID, gage ID or the coordinate that is clearly shown on the map (see detailed explanation in the result section 5.3).

5 Results

5.1. Final code and Visualization of Maps

The Python package is uploaded in Github. The results of data analysis and queries are plotted in maps and can be checked in our Jupyter Notebook.

5.2. Discharge Trend at USGS Gages

We plotted the trend analysis result for the USGS gage at James River near Big Island, VA (ID is 02024752). This gage is detected significant increasing trends (Figure 3) on both monthly and yearly discharge data. We used interactive maps in Jupyter Notebook to show the trends for all gages in the Chesapeake Bay watershed (more in section 5.3).

5.3. Query Results and Visualization

5.3.1. Find gages have more than 10-year records of discharge within the Chesapeake Bay

After obtaining metadata for all USGS gages in the Chesapeake Bay watershed, we noticed there were some gages not monitoring the discharge data or the records of discharge were less than 10 years. Therefore, our first query is pretty simple: to find gages that have more than 10-year records. This information can be obtained from the trend line slope, as we did not fit trend lines to gages that 1) have no discharge observations or 2) have less than 10-year data. The trend slopes for these gages would be empty.

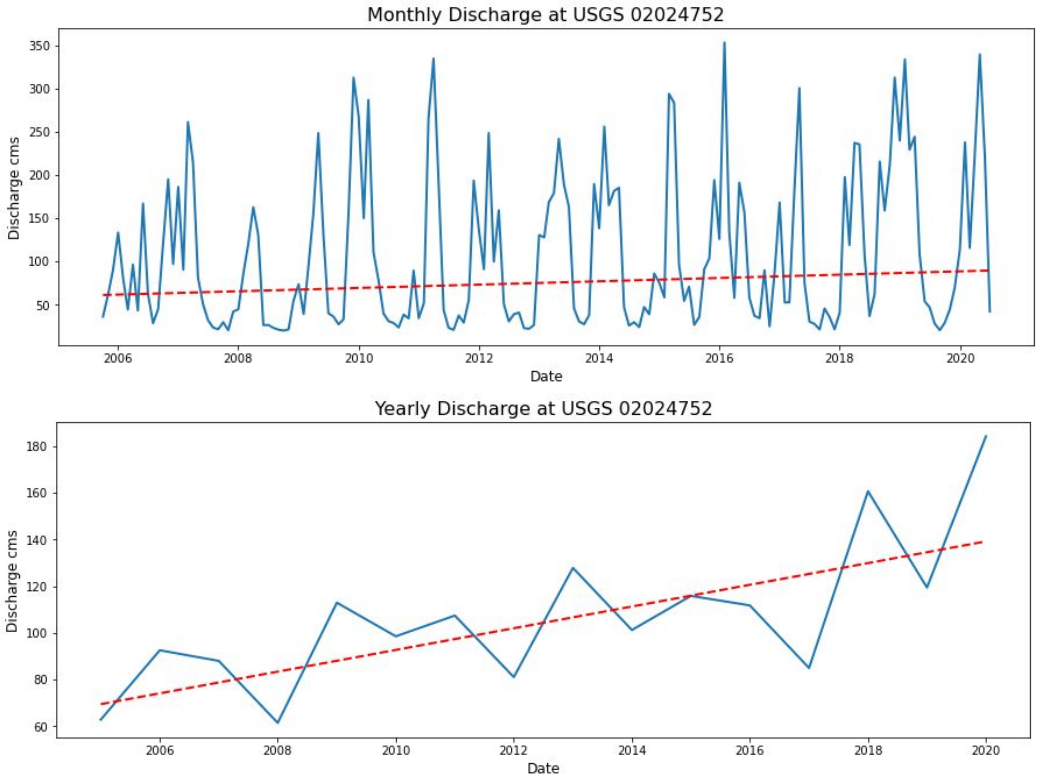

Figure 3. The (up) monthly and (bottom) yearly Time-series of discharge data at USGS gage 02024752, James River near Big Island, VA, with the fitted trend lines. Significant increasing trends are detected at both monthly and yearly scales.

Figure 3 shows a sample time series at USGS gage 02024752 with a trend line. If the Menn-Kandell test indicates a significant trend, the median slope calculated from Sen's slope method is shown with a red line. It is clear that the monthly and yearly trends are different; thus, it is essential to consider different time scales when we test trends in the data.

In Pandas, our query is performed as: **sub_data = df[~df['slope'].isnull()]**. We found 432 gages that had been observing discharges more than 10 years within the Chesapeake Bay watershed.

5.3.2. Find number of qualified long-term gages in each state

As geospatial metadata of each gage was saved in the table, we could query to find the number of qualified gages in each state within the Chesapeake Bay watershed. Here, we used the Pandas function "pivot_table" to generate a summary table:

State	DE	MD	NY	PA	VA	WV
GageID	2	73	45	126	164	22

Among 6 states within the Chesapeake Bay watershed, Virginia has the most USGS gages (164 out of 432, 38.0%) that monitor the discharge more than 10 years. Pennsylvania has 126 (29.2%) long-term gages, the second largest number within the Chesapeake Bay. Other states (Maryland, New York, West Virginia, and Delaware) have 133 (30.8%) gages in total that monitor the discharges.

5.3.3. Classify Gages with Different Trends on Monthly Discharge

The slope of the trend line is an indicator of whether the monthly discharge is stable, increasing or decreasing. The greater the absolute value of slope, the more significant the discharge changes over time. We grouped the daily discharge data by month and calculated the slope value using Mann-Kendall trend test[1]. We then classified slopes into five groups: 1) significant decrease (slope <= -0.15), 2) decrease (-0.15 < slope <= -0.05), 3) no change (-0.05 <= slope < 0.05), 4) increase (0.05 <= slope < 0.15), and 5) significant increase (slope >= 0.15).

As the table saved coordinates (latitude and longitude) of each gage, we made a map (Figure 4) to show the distribution of gages within the Chesapeake Bay watershed.

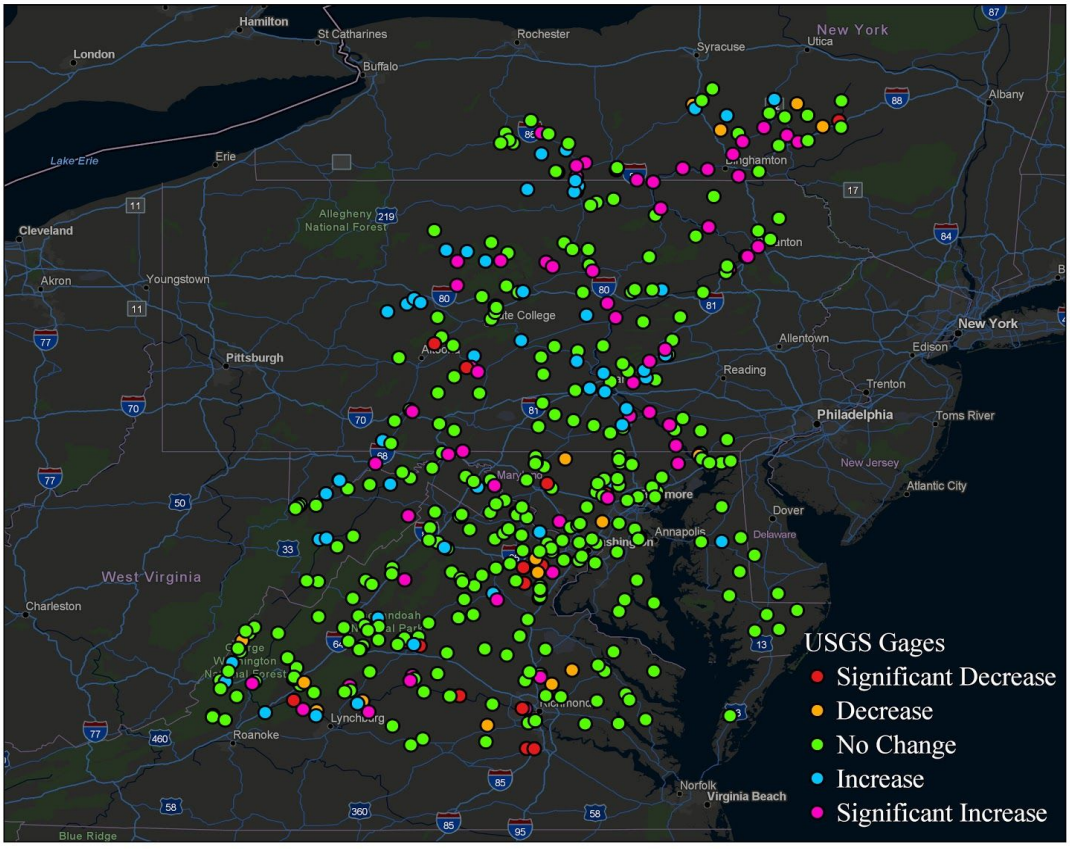

Figure 4. Monthly trends of discharge for gages having more than 10-year observations in the Chesapeake Bay.

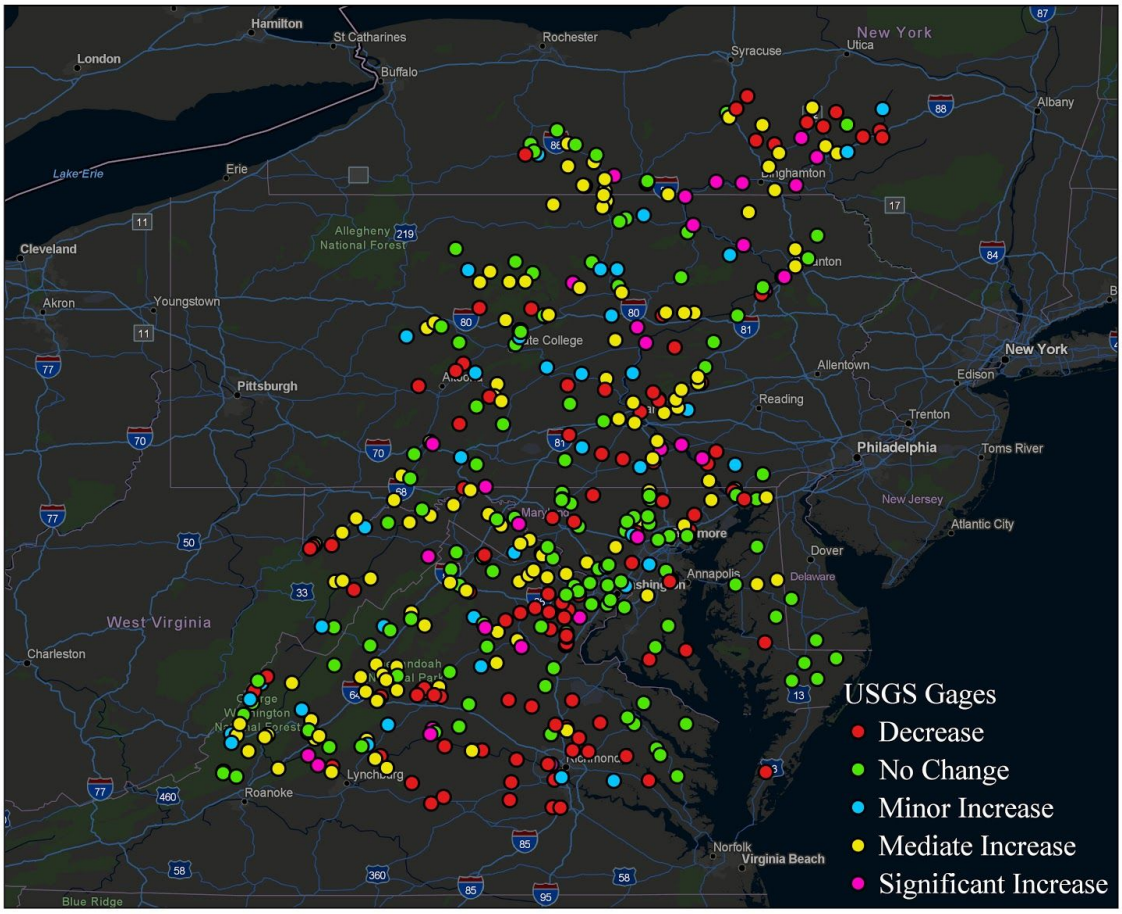

Figure 5. Yearly trends for gages that have more than 10-year observations in the Chesapeake Bay Watershed.

For each class, 15 gages had decreasing significant decrease trend, 17 had decrease trend, 286 had no change, 57 had increase trend, and 57 had significant increase trend. We did not find a spatial pattern of clustering of increasing or decreasing trends in the monthly interval.

5.3.4. Classify Gages with Different Trends on Yearly Discharge

We also performed the trend analysis on yearly mean discharge data. The trend of yearly data should be more sensitive than the monthly data, as the random events, such as hurricanes that significantly elevated the discharge were averaged. Therefore, we used different classification breaks to classify the yearly slopes. For yearly data, we classified slopes into 5 categories: 1) decrease (slope < 0), 2) no change (0 < slope <= 0.5), 3) minor increase (0.5 <= slope < 5), 4) mediate increase (5 <= slope < 10), and 5) significant increase (slope > 10). For each class at yearly scale, 108 gages had

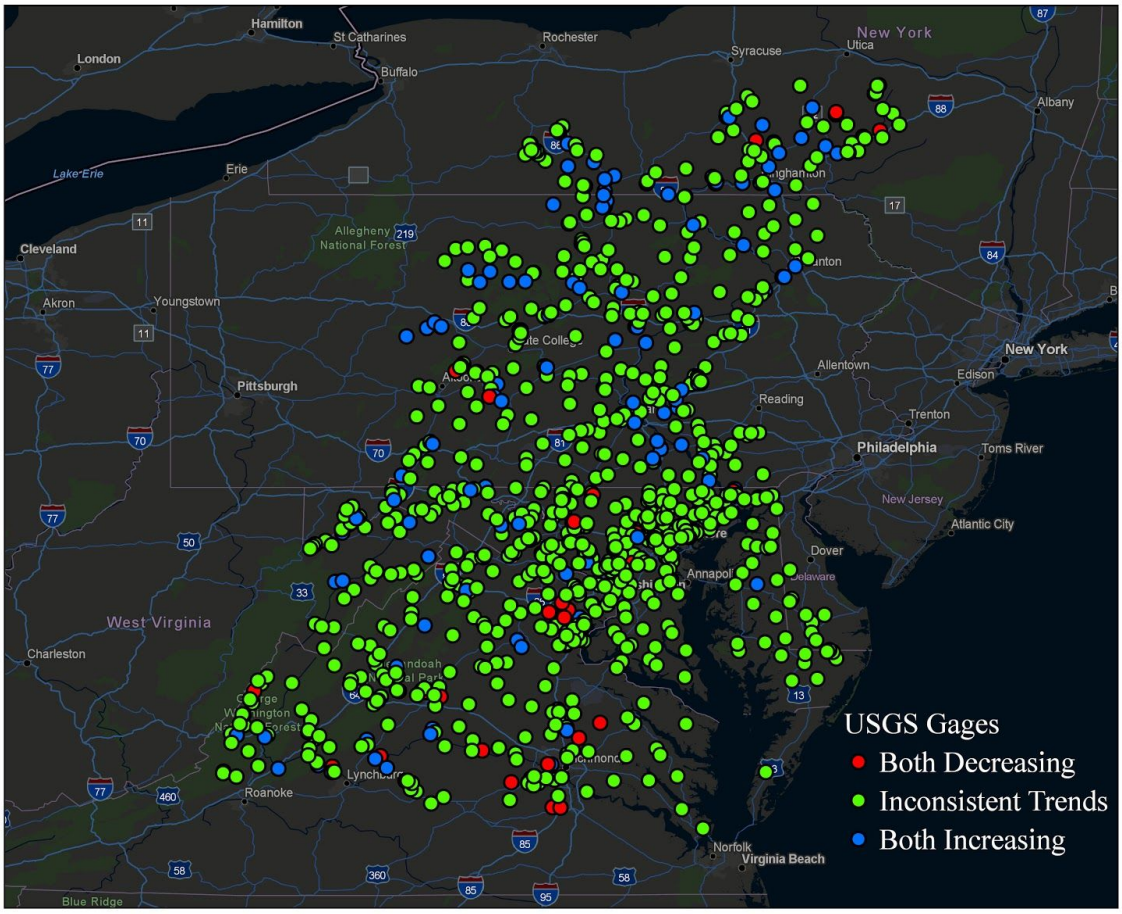

Figure 6. Consistency of monthly and yearly trends of gages within the Chesapeake Bay Watershed.

decreasing trend, 254 had no change, 51 had minor increasing trend, and 15 had mediate increasing trend, and 4 had significant increasing trend (Figure 4). We did not find a clustering of significant increase trend, but there seemed to have a clustering of mediate increasing trends in the center of Pennsylvania. Decreasing trends were found clustered in several gages in Northern Virginia and Richmond, VA.

5.3.5. Find Gages with the Same Trend on Monthly and Yearly Discharge

As we mentioned earlier, monthly trends can be masked by the outliers caused by extreme precipitation events. Therefore, we believed that the gages having the same sign of both trends would be more convincing to show the over trends of water resources change. Therefore, we applied a query to find both slopes were positive and negative, with the remaining gages inconsistent trends.

In this analysis, we found 110 gages (25.5%) in the Chesapeake Bay had

increasing trends and 29 gages (6.7%) had decreasing trends (Figure 6). There were 293 (67.8%) gages showing inconsistent trends on the change of discharge. We found the gages in North Virginia and Richmond areas showed consistent decreasing trends of water resources, and gages in Center and North Pennsylvania had clustered increasing trends. Most gages did not show consistent trends on both monthly and yearly trends.

5.4. Testing: describe what testing you did. Describe the unit tests

We performed a total of ten tests on our file USGS_FlowDAta_utils, we directly tested our child class USGS_DataRetriever_Test since it inherent all the attributes from the USGS_Gage class. The tests are described as below:

1. Test ‘test__init__id’
   We tested the __init__ method by first using assertEqual() function, to see if the ID number has a length of 8 digit.
2. Test ‘test__init__vars_info’
   We checked if the length of the variableID is equal to 5.
3. Test ‘test__init__vars_info2’
   We checked if the length of the variableID was equal to 5 and if the variable name was equal to ‘Discharge(Mean)’.
4. Test ‘test_getDailyDischarge’
   We tested the getDailyDischarge() method by using assertEqual() function to check if the data type of the data is float.
5. Test ‘test_getStatistics’
   We tested the type of the data retrieved from the getStatistics() by using assertEqual() function to see whether all of the variables are float type.

   We tested if we have the correct data by first access Min, Max, Median, Mean and Standard Deviation then combine it into the list then use assertEqual() function to test if all the number is equal to our expectation

6. Test ‘test_getStatistics_2’
   In addition to the unit test 5, We want to test if the length of the dictionary which we got from getStatistics() methods equals 5.
7. Test ‘test_getUnit’
   We tested the getUnit(), set the parameter “metric” to True, and test if our unit is ‘cms’.
8. Test ‘test_findLargestEvents’

We tested fundLargestEvents by using assertEqual() function to see if the output is the number we input. We also tested for a gage to test if our top 2 values of 'Flow(cms)' and 'Date' columns have the same expected values and dates.

9. Test 'test_getVarsMetaData'
   We tested GetVarsMetaData() by using assertEqual() to check if all the data types meet the corresponding formats of 'str, int, and datetime' requirement and their values meet the expected value from the website.
10. Test 'test_getGeoMetaData'
    We tested getGeoMetaData() by using assertEqual() to check if the data we are looking for is following the format of tuple, str, int, int and str and the values meet the expected ones. If the unit test passed, which means our web scripting process met our expectations for further analysis.
11. Test 'trendTest'
    We tested trendTest by using the assertEqual() to check the both value and format of the ten variables by using assertEqual().

We passed all the Unit Test, which proved that our code correctly downloaded the data and the methods within each class met our expectations.

6 Conclusions

We tested the ability of our program to download daily discharge data from all gages in the Chesapeake Bay. Excluding 283 gages that do not monitor discharges, we successfully downloaded the data for the rest 644 gages, obtained metadata, and performed trend analysis for discharge at both monthly and yearly intervals. In the design aspect, we believe the goal that users enter minimum input to use the program is achieved, because two classes in our program only require one parameter, the USGS gage ID, to build an object and execute methods. Our program is available on Github now (https://github.com/ruoyu93/USGS_DataRetriever).

The trend analysis was performed on 432 gages that have more than 10 years of records. At the monthly interval, we found 114 gages had increasing trends, 32 gages had decreasing trends, and 286 gages had no trend detected. At the yearly interval, 108 gages had increasing trends, 70 had decreasing trends, and 254 had no trend detected.There were 293 gages that had the same trend on both monthly and yearly intervals. Combining both trends together, 110 gages (25.5%) in the Chesapeake Bay had consistent increasing trends, 29 gages (6.7%) had consistent decreasing trends, and 293 (67.8%) gages had inconsistent trends on the change of discharge. The results are stored in this Jupyter Notebook.

This program can help not only hydrologists but also others that are interested in water resources to easily download, query, and analyze the discharge data at USGS gages. However, discharge is only one variable in hydrology, our program currently does not support downloading of other variables, such as water temperature, water height, pH level, etc. If we had more time, we would extend the program's ability to retrieve other variables as well. Another function that our present program is lacking is to retrieve 15-minute interval water data.The USGS is now improving the temporal resolution of water data with 15-minute intervals, which are crucial to predict flash floods. If we had integrated the program to handle both daily and sub-daily data, our program would be used by people studying and predicting floods in the future.

References

[1] Trenberth, K. E. (2011). Changes in precipitation with climate change. *Climate Research*, *47*(1-2), 123-138.

[2] Mann, H. B. (1945), Nonparametric tests against trend. Econometrica, J. Econometric Soc., 245–259, doi:10.2307/1907187.

[3] Sen, P. K. (1968), Robustness of some nonparametric procedures in linear models, Ann. Math. Stat., 1913–1922 http://www.jstor.org/stable/
223929.

Live presentation slides
https://docs.google.com/presentation/d/1Dx11Tb4M5XlF4_Tl-0kE3Tr63nOF7DC2VCHhGvS5s-Q/edit?usp=sharing

RUBRIC

Description	Possible Points	Earned Points	Comments
Introduction: Describe the project scenario (can be brief) [Roy]	5		Roy
An appropriate data set was used ● Explanation of data set/domain/where data was obtained ● Size (large enough)	10		Yibo
What data structure was the data stored in?	5		Yibo

Performed any data pre-processing? Such as (but not limited to): ● data cleaning ● outlier removal ● ..etc to sanitize the data	10		Roy
Data Analysis / Data Processing (queries) ● At least four (different)	20		Yusheng
Testing: Describe any test-driven development and/or unit tests	10		Keyu
Results displayed appropriately for each test	10		Keyu and Trello
Explanation of Results & Conclusions ● How these results can be used by others (relevance/significance)	20		Hyunglok Kim &

• Ways to improve / ways to expand / adding or removing functionality			
Presentation skills and video • Video presentation posted to the discussion board is no longer than 20 minutes and all group members presented aspects of the project • All group members spoke in the video and were present at the live session • The live session presentation was between 4-8 minutes • The presentation was of good quality, clear and easy to understand	10		
Total Points	100		

Extra Credit: [Max 3%: 1% per item, up to 2% for single item if substantial] ● Web-scraping to obtain data (instead of downloading) ● User-interaction to retrieve/display/analyze certain results -or- ● User-interaction to obtain some more data ● Advanced queries / manipulated the data in another way	Max 3%:1% per item, up to 2% for single item		
Final Grade			