Repository and Mining of Temporal Data
Milestone 2 Evaluation

Team Members:
Jessica Nguy jnguy2014@my.fit.edu
Siomara Nieves snieves2014@my.fit.edu

Faculty Sponsor and Client:
Philip Chan pkc@fit.edu

Meetings With the Client:
Oct 13; Oct 30

Progress of Current Milestone:

Task	Completion %	Jessica	Siomara	To-Do
CSV User Input catch cases	95%	0%	95%	Add more test cases
Q1	90%	80%	10%	Test, x-axis for figures 2 and 3, change to plotly
Website	100%	80%	20%	Completion deferred to later Milestone
Data Processing	N/A	N/A	N/A	Deferred to Milestone 3
Database setup for Meta-data, Meta-data inputs	80%	0%	80%	Switch Database to SQLite; Deferred to Milestone 3
Evaluation Document, Presentation	100%	60%	40%	None

Discussion of Each Accomplished Task:

Task 1: Python program accepts a single .csv file and checks for the values on top of the columns to know which one is the time and which one is the data. The file saves the time and data values into two lists. The lists are later converted into two separate arrays and then passed when called by Q1. Test cases for now include having blanks, having something written other than a float value, and having dots on the .csv file. Further research into handling different timestamps (Dates being in Year-Month-Day, Year/Month/Day, Hour:Minute:Second, etc) required.

Task 2: Programmed Q1, which consists of calling csv.py and taking the data as arrays then changing the timestamps as strings to datetime objects for plotting and to floats for calculations.

For the first visualization: the program will graph the target variable versus time. There is no additional calculations done with this graph, as the graph is used as a benchmark.

For the second visualization: the program calculates the change in the target variable and plots it against the time for when the change occurs. The program will display the second graph, which is the change in the target variable versus time. The z-score is reverse-calculated for where in the data the points (-3.0, -2.5, 0, 2.5, and 3.0) are located, and a dashed line is drawn across the y-axis in these points. The graph will change color when it is plotted between the dashed lines; between (- infinity, -3.0) and (3.0, infinity) the color of the line is red; (-3.0, -2.5) and (2.5, 3.0) is yellow, and (-2.5, 2.5) is green. Colors of the line do appear 'red' as it crosses over into the green zone, however that is to display the significant change between the variables for that specific timestamp.

For the third visualization: the program calculates the z-score of the initial values and plot it on a graph, which is the z-score of the target variable vs time. Between the thresholds of (- infinity, -3.0) and (3.0, infinity) the color of the line is red; (-3.0, -2.5) and (2.5, 3.0) is yellow, and (-2.5, 2.5) is green.

All graphs are plotted with matplotlib. There is an option to save the data, if the user wishes to do so. Matplotlib will be substituted for plotly for better use of datetime objects. Further enhancements to Q1.py is to change from using matplotlib to plotly, enhance the appearance of the graph, and code cleanup.

Task 3: Creation of the website has been deferred to a later Milestone. There are 5 pages currently. The website is hosted on personal machines, requires a command line request to begin hosting, and is not on a server. The website has three links on its homepage which direct to another page. Logins for users is set up, along with an admin page.

Task 4: Data processing was not used to create Q1, so the task was deferred to Milestone 3, when the group will work on Q2.

Task 5: Database setup is complete, however the group must switch from SQL to SQLite. The database will be further enhanced on Milestone 3. The current SQL database saves the metadata (tags, description, start/end timestamp, public/private) which will be further discussed in Q2.

Task 6: The Evaluation document is written using Google Docs. The presentation is created with Google Slides.

Discussion of Contribution for Each Task:

Jessica: Worked on setting up Django web application, which is hosted on personal computer. Created 4+ pages on the web application, one of which includes admin/user logins. Wrote the code for Q1.py and did necessary calculations and research into plotting the visualizations correctly. Trouble-shooted display problems with matplotlib and Q1.py Checked to ensure that integration between csv.py and Q1.py worked. Contributed to most of the Milestone 2 evaluation document.

Siomara: Wrote the read .csv program, which as of now only reads files locally, worked on integrating Q1.py with csvReader.py by creating a change of strings to datetime objects and floats. Researched into plotly visualization of data, researched into how to store user's files upload, researched into SQLite, wrote a dummy version of Q1.py using plotly. Contributed to most of the presentation document.

Plan for the Next Milestone:

Task	**Jessica**	**Siomara**
1.) Q2	Calculations for Pearson correlation, cross correlation, visualizations, writing main code	File storage, file conversion, visualizations
2.) Narrow Data	Research and create program to narrow data. Integration with database may be needed. Create NarrowData.py	Research how to implement a search for queries depending on the metadata provided by other users.
3.) Database Processing	Make sure that code can use information from the database	Linking database to Django, running test cases for queries and selections
4.) Meta-Data, Meta-Data	Create Input.py, be able to	Asking user for metadata

inputs	retrieve metadata and use it to run Narrow Data.py	input and storing into the database
5.) Improve existing code	Ensure that changes in .csv are carried over and do not break Q1.py and Q2.py	Creating more catch cases for errors in file, x-axis plotting for figures 2 and 3, change to plotly
6.) Evaluation Document, Presentation	Write evaluation document, Create presentation, put code onto GitHub repository.	Write evaluation document, Create presentation, put code onto GitHub repository.

Discussion of Each Planned Task:

Q2: Program will ask for user input for a target variable. Using that variable, the program will proceed to narrow the data to parse out .csv files that have no correlation with the target variable. Once the data in the repository has been narrowed down to 10 csv files, the program will calculate the Pearson correlation and cross correlation for the target variable and display several graphs.

The first visualization will consist of the target variable and two additional values. The variable and its lag time will be denoted on the line graph, with the x-axis being time and the y-axis being the rating of the values (with lag denoted). Scaling of the objects (i.e., Presidential ratings is scaled from 0 to 100, whereas unemployment percentages is from 0 to 10) will be worked on in a future milestone.

The second visualization is an in-depth version of the first graph. It will be a bar graph, with the x-axis being correlation values from 0 to 1.0, and the y-axis containing the target variable and two additional values (with lag indicated). The graph will consist of the absolute value of the correlation. The graph will be colored based on whether or not the original correlation was negative (in red), or positive (in green). An animation consisting of clicking on the bar graph will display a corresponding graph depicting mathematical reasons why that variable was selected to be displayed will be available, as well as a zoom-in option.

The third visualization will be of the cross correlation of the target variable. It will display when the correlation when it happened across timestamps. The x-axis will consist of the lags from 0 to 10 timestamps, while the y-axis is the correlation from -1.0 to 1.0. A dashed line will mark where 0.0 correlation is. If the granularity is too large (yearly data vs daily data), then the correlation will not be calculated, as there are not enough values to be able to calculate the cross-correlation.

Test Q2.py against known examples to ensure that the code works correctly. Additional display options, such as displaying occurrences of large granularity and inabilities to calculate cross-correlation and pearson correlation because of this will be expanded on in future milestones.

Narrow Data: Implement a program called NarrowData.py that takes the inputs from Input.py and searches through the database for values that match the tags.

Tag requirements include:

Time range: the time range for files within the database must occur several timestamps before the time range of the target variable and end at or near the end of the time range of the target variable.

Values: values within the database must match one or more items designated in Input.py as searching tags - if there is more than 10 results that return matching tags, then the results that have 2 or more matches will be returned. Additional combing of the searching tags vs data descriptions will be used to refine the search further.

Granularity: methods to handle different granularity sizes will be expanded upon in later milestones.

These values will be used in Q2.py to analyze the data for the correlation between matching values and the target variable. The public/private metadata will be added on in future milestones. Test the code against known results to ensure that the code is working correctly. Make sure that the code works with Q1.py, csv.py, Q2.py, and the database.

Database Processing: Link database to Django and correlate Input.py to the database. Convert current SQL database to SQLite. Test cases for queries for target variable comparison to make sure code works well when trying to retrieve the data. Be able to retrieve and store data in/from database.

Metadata, Metadata Inputs: Implement a program called Input.py that takes in user input for the target variable, a csv file input, and additional user-defined tags, including: time range, additional search values, and a short description of the uploaded csv file. Parsing the time range into readable floats requires additional research.

Improve Read of Existing Code: Current web application runs locally, link it to a web server with Django. Find a way to store all other .csv files on the web application and run test cases for evaluation of variables. Perform code cleanup to improve readability on Q1.py. Ensure that with the changes in csv.py that it will still work with Q1.py and Q2.py.

Evaluation Document, Presentation: Creation of an evaluation document to indicate what was completed in Milestone 3, future requirements in Milestone 4. Create detailed explanations of completed milestones and detailed plans of future milestones. Create a Presentation for Milestone 3. Upload completed and/or current documents to GitHub repository and update links on main page.

Sponsor Feedback:

Task 1: Handling different time formats, different data sets (not just SP 500), not only 2 columns but 2 rows as well.

Task 2: “zoom-in” most recent values, graph z-score of change of x vs time.

Task 3: M3 a) At least 10 variables and 3 plots

M3 b) different granularity, narrowing data based on timestamps

Task 4:

Task 5:

Task 6:

Sponsor Signature: _____ Date: _____

Sponsor Evaluation

- Sponsor: detach and return this page to Dr. Chan (HC 322)
- Score (0-10) for each member: circle a score (or circle two adjacent scores for .25 or write down a real number between 0 and 10)

Jessica	0	1	2	3	4	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10
Siomara	0	1	2	3	4	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10

Sponsor Signature: _____ Date: _____