Homework 6: Data Visualization Framework

In this assignment you will work as a team to design and implement an extensible data visualization and analysis framework, consisting of an interactive GUI tool and underlying interfaces and implementations.

In this assignment you will practice to (a) Design and implement a black-box framework with a common plugin interface for data from a wide variety of sources and a common plugin interface to perform a wide variety of data visualizations. (b) Perform domain analysis to determine appropriate data sources, visualizations, and data representations for your chosen domain. (c) Coordinate software design and development within a team and between teams. (d) Demonstrate proficiency with others' code by using open-source, third-party libraries and developing plugins for another team's framework.

Framework overview and architecture

The framework will provide a web-based frontend for data analysis and visualization. A good framework provides a lot of utility with reusable functionality and also makes it easy to analyze more data and provide more visualizations.

When you are done, you will have a framework to play with data and analyses from a field that interests you, from politics to juggling, from biodiversity to baseball, from social media to traditional news. Be creative when you design, implement, and use your framework---surprise us! This homework enables a far broader variety of solutions than the previous ones.

Your framework will support two types of plugins:

- **Data plugins** that extract data from some source and make it available for processing. Possible sources include (but are not limited to) local files, web pages, web APIs, mathematical formulae, or sensors. Data could be tabular data, graphs, events, locations, text, images, or any domain-specific notation. Many popular websites provide APIs to access data (e.g., Wikipedia, Twitter, GitHub), many sites provide live feeds (e.g., RSS, weather stations), many sites provide data for analysis (e.g., Your World in Data, WPDRC). For many data sources libraries exist to make accessing the data easy (e.g., GitHub for Java/JavaScript)
- **Display plugins** that visualize data provided by the data plugins, possibly after some processing. Possible visualizations include (but are not limited to) maps, pie charts, bar charts, histograms, statistical abstracts, X-Y plots, scatter plots, bubble charts, or choropleths. Many libraries exist and are easy to embed in HTML frontends, even without much HTML/JavaScript knowledge, such as ECharts-Java, Plotly, C3, Chart.js, or Kartograph.

Between those plugins, the framework can do some data processing. For example, interesting frameworks may

- merge data from multiple sources
- enrich data with additional information (e.g., use machine-learning APIs to estimate the sentiment of text or the objects in images)
- aggregate data (e.g., aggregate time series data by month)

It is a design decision for the framework designer of how much processing is done by the framework or by the plugins. For example, grouping time series data by month could be done in a data plugin, in the framework, or in the visualization plugin. Work done in the framework provides more reuse value to plugin authors, but also possibly restricts what kind of plugins are possible.

Conceptually this will look like this:

```
┌─────────────┐                                       ┌───────────────┐
│ Data Plugin ├──►┌────────────────────┬──►┌───────────────┐
└─────────────┘   │                    │   │ Visualization │
```

```
┌─────────────┐                         ┌───────────────┐
│ Data Plugin ├──►┌────────────────────┬──►│ Visualization │
└─────────────┘   │                    │   └───────────────┘
                  │ Framework:         │
┌─────────────┐   │                    │   ┌───────────────┐
│ Data Plugin ├──►│ UI, abstractions,  ├──►│ Visualization │
└─────────────┘   │ data processing,   │   └───────────────┘
                  │ lifecycle          │
┌─────────────┐   │                    │   ┌───────────────┐
│ Data Plugin ├──►└────────────────────┴──►│ Visualization │
└─────────────┘                            └───────────────┘
```

Making the framework interesting. Remember the "maximizing reuse, minimizes use" phrase from the framework lecture. The more generic a framework, the less interesting it becomes for reuse. A good framework will be domain specific and will provide a lot of power within that domain. A good framework focused on a single issue is better than a generic one that could visualize all kinds of data. For example, generic tabular data typically only allows for little processing and generic visualizations, but time series data or geo data would facilitate more interesting processing and visualizations that could work out of the box. An interesting framework will do some processing on the data between data and visualization plugins and not just pass all data along directly. Appendix A gives an example of a more interesting framework.

Teamwork, milestones and late days

The assignment has some somewhat unusual mechanics with multiple milestones and involves teams. Hence, please read the following instructions carefully as they differ in several ways from prior assignments.

Teams: Work in teams of 2 or 3 students. You can form teams yourself. If you are looking for teammates, consider the pinned Piazza post for finding team members. You will use a shared Git repository to coordinate and turn in your work. Coordinate so that one team member creates the repository with this GitHub Classroom link and have everybody else then join this project. Be careful about signup, because GitHub makes it difficult to fix mistakes (send a private post on Piazza for help).

Milestones: This homework has three separate milestone deadlines and a team presentation during recitation. Do note that the effort required per milestone is not equal and that Milestone B will likely take longer than Milestone A or Milestone C.

- **Milestone A:** You will design your framework and submit design documents on Apr 11.
- **Design presentation (optional, recommended):** Instead of recitation, on Apr 13, you can present your work for feedback at the time slot you sign up for.
- **Milestone B:** Your framework and sample plugins implementation, along with design documents and documentation is due Apr 18.
- **Milestone C:** Write plugins for one of the "selected frameworks," due Apr 29 (Friday, last day of class!). Authors of selected frameworks have two more days until May 1 to hand in their report.

Late days: Each team gets 2 *team late days* that can be used on any milestone. You can use up to two team late days per milestone, as usual. In addition, in a team of *N* members you can convert *N* leftover individual late days of any team members into one *team late day* (i.e., you can pool your remaining individual late days). For example, if you have a team with two members where one member has 1 late day left and the other member has 4 late days left, your team can use an extra 2 *team late days* (rounding down from 5/2).

You may use late days for Milestone B, but if you want your framework to be considered as "selected framework" you need to submit your Milestone B by 9am of Apr 20.

You cannot use late days for the design presentation.

Framework requirements

You have much freedom to be creative in your framework design and implementation.

At minimum, your framework must achieve the following design goals:

- Your framework should support an analysis of your choice that is generic enough to allow multiple data sources and multiple visualizations but also specific enough to provide value through reuse and shared functionality.
- Your framework must support (a) data plugins and (b) display plugins for a domain of your choosing. A plugin developer should be able to implement additional data sources and visualizations with only a small amount of work and benefit from the reuse of the entire framework. Implementing a new plugin must not require changes to the framework code.
- Your plugin interfaces must be general and interchangeable. Plugin interfaces should hide the details of their implementations and permit visualizations to be applied to data sets regardless of their source. You will have at least two separate plugin interfaces, one for each kind of plugin. If necessary, you may create multiple data source interfaces to expose appropriate details for incompatible data types. Visualization plugins will typically return HTML that will be embedded in the web-based GUI provided by the framework in one or more locations.
- Your web-based GUI must allow the user to create or select data sets from multiple data sources and to create multiple visualizations of those data sets.
- Your framework must support a mechanism to initialize and parameterize data sets and visualizations. For example, a data set that draws its data from a CSV file would need to know the path of the file and some description of the relevant columns. A data set that draws its data from a web page would need to know the web page's URL.

Please read the grading rubric below carefully for requirements we consider during grading. Key points are:

- You need to implement a web-based user interface for your framework. The user interface can be used to select which data to collect or how to visualize it. The web-based user interface will integrate the output of visualization plugins.

- You must be able to compile and run the framework (with all library dependencies) from the command line using Maven or npm. A `README.md` file needs to describe how to compile and start the framework.

- The framework must be able to load plugins without requiring changes to the framework code (you can use the same plugin loading mechanisms as in Recitation 10). It will interact with the plugins to receive data or produce visualizations. The plugins do not communicate directly with each other.

- You need to provide sample plugins to illustrate the functionality and flexibility of your framework. The plugins should be different from each other to illustrate the range of possibilities. At least one third-party library must be used for a visualization plugin.

- As others may implement more plugins, good API design and documentation are essential. Describe in your `README.md` file how to extend the framework with additional plugins and write clear API documentation for all relevant interfaces or methods (including plugin APIs and all relevant data structures used for input/output).

- We expect some testing of your framework, including some automated unit testing. Your tests need to include some tests that use stubs (or mocks -- we will cover these in class soon) to test code with missing pieces or to test for correct error handling. This can simulate the behavior of some software component (such as a plugin) for the purpose of testing another software component (such as your framework). We recommend, but do not require, that you test your plugin implementations. You do not need to automate tests for GUIs.

- Optional (for bonus points): Your framework can do some data processing using a machine-learning API (e.g., object detection, sentiment analysis, network analysis), as machine learning is a natural and common way to augment frameworks. If you do this, add a section "Data processing" to your `README.md` file, where you briefly describe what you are doing.

If you feel like a challenge, there are many ways you can exceed these minimal requirements. For example, you could also provide persistence of your data sources and visualizations, allowing users to continue a visualization when they restart the tool.

Milestone A: Plan and design your framework

For this milestone you will design your framework (identifying its common and variable parts and producing a preliminary API), and also plan how you will work as a team.

Design. For this milestone you will describe some of your key design decisions with text and diagrams. It is likely a good idea to already experiment with implementations.

Submit a file `design.md` with the following sections:

- **Domain:** Describe your framework domain, with examples of possible plugins your framework could support. This could be about half a page, similar to the description in Appendix A.
- **Generality vs specificity:** Describe your decisions about the generality and specificity of your framework (i.e., domain engineering): your key abstractions, the reusable functionality your framework provides, and the potential flexibility of plugins. This might be about 1 page.
- **Project structure:** Describe your overall project structure: the organization of the framework and plugins into packages or projects, the location of plugin interfaces and key data structures, and how plugins are loaded. This will help us navigate your project later and will likely require less than half a page.
- **Plugin interfaces:** Describe your plugin interfaces (code snippets or API documentation), including key methods and the data structures exchanged between plugins and the framework. Feel free to include or link to diagrams. This may require multiple pages, depending on the size of your APIs.

Work plan. Submit a separate `plan.md` file that describes how you will divide work among your team members, and describe any internal deadlines you will meet to ensure a high quality product for Milestone

B. Your team responsibilities may overlap, but one person should be principally responsible for each artifact of your project.

We suggest, but do not enforce, that you update the plan document as you continue working on the project.

Design presentation (optional, recommended)

We strongly recommend (and incentivize with 10 bonus points) that you present your design to your classmastes and TAs during a recitation session on Nov 17.

To sign up for a presentation slot, add yourself to this spreadsheet. You can present in any recitation, independent of your section. Note that presentation slots are limited and can be claimed in the spreadsheet first come first serve. We will update locations for the presentations in the spreadsheet on the evening before.

In at most *8 minutes* total, cover the key points of your design in Milestone A, particularly domain, generality, and plugin interfaces.

Create *6 or fewer slides* for your presentation and upload them as `presentation.pdf` to your team's repository before the presentation. Include UML diagrams and/or code as necessary to clearly illustrate your framework design and how it could be used.

Your primary audience is your peers and your goal is to illustrate how you achieve reuse in a domain.

Unless arranged upfront otherwise, you will present from your own laptop. We will have HDMI and USB-C connectors.

Your entire team must give the presentation at your presentation time, and *all team members should actively participate* in the presentation.

Milestone B: Framework and sample plugin implementation

Implement, document, and test your framework, and write sample plugins to demonstrate how the framework can be extended and used. A two-person team must implement two data plugins and two visualization plugins. A three-person team must implement three data plugins and three visualization plugins.

Read the description above and the grading rubric below for minimal requirements. Please pay particular attention to make it easy for others to start and extend your framework. A good readme and good documentation will be useful both for selecting frameworks, for grading, and for other teams who may implement plugins.

We strongly recommend to run a static analysis tool or linter like Spotbugs, ts-standard or similar.

Finally, remember that we cannot consider all late submissions as candidates for "selected framework" for Milestone C (see details above). If your team's framework is selected, then you will receive 10 points extra credit and you will not need to write any plugins for Milestone C.

Milestone C: Plugin implementations for another framework

Soon after the Milestone B deadline we will select several framework implementations for other teams to plug into for Milestone C. If we select your framework, your team must support your framework for others (and you will receive some bonus points). If we do not select your framework, you must implement plugins for one of the selected frameworks.

Supporting a selected framework. If your framework is selected you must provide technical support for the teams building plugins for it. This includes responding to questions promptly on GitHub, fixing bugs in your framework (if necessary), and addressing misunderstandings teams have about your framework. Overall, your goal is to keep the other teams happy and ease their task of implementing plugins for your framework.

Please note that none of your framework code can be changed directly by the developing teams. This has two implications: (1) your framework must allow dynamic loading of plugins (outlined in the previous section), and (2) you must fix bugs reported by other teams as soon as possible and update your framework accordingly.

As you support your framework you should keep all communication between you and other teams public, using GitHub for all technical support. This is because (1) you will be graded on the quality of technical support you provide, and (2) all development teams can benefit from seeing the problems raised by others, as well as their eventual resolution. You should not provide support beyond what is legal or socially acceptable, or that would violate the course cheating policy. If in doubt, ask the course staff.

At the conclusion of this assignment, you must also submit a short report (at most 1 page) of your experience providing support for your framework. Submit this report in as `experience_report.md` to your repository (you can do this until 2 days past the Milestone 2 deadline)

Writing plugins for another team's framework. If your framework was not selected, your team must develop plugins for another team's framework. Two-person teams must develop two data plugins and one visualization plugin. Three-person teams must develop three data plugins and two visualization plugins.

At least one of your plugins must use an extra third-party library; you may use any library you want to aid in data analysis or visualization. Your plugins for Milestone C cannot use the same date source or visualization as the sample plugins for the framework you are plugging into, but you may otherwise base your Milestone C plugins on your team's own plugins from Milestone B. You are permitted to use the same libraries that you used in Milestone B.

In your `README.md` file provide pointers to the plugins you implemented and instructions on how to start the framework with your plugins.

You will have access to the GitHub repositories of the selected frameworks and their documentation. The teams of selected frameworks will provide instructions of how to add dependencies on the framework for your own projects. You will use *GitHub issues* to communicate with the members of the selected framework teams, and you may submit *pull requests* on GitHub. Please also note that you may not change any framework code directly; your plugins must work for the official version of the framework you are plugging into. If you think you have found a bug in the framework, report the problem or submit a patch (e.g. a pull request) to the framework team's for them to address.

At the conclusion of this assignment, you must also provide feedback (at most 1 page) on the quality of technical support from the framework-providing team. Submit this report in as `experience_report.md` to your repository together with your implementation.

Submitting your work

For deadlines, see milestone descriptions above. As usual, always commit and push all solutions to your team's GitHub repository.

Avoid committing passwords or other credentials to your GitHub repository. If credentials are needed to use some plugins, provide instructions on how to acquire an API key instead.

For this assignment, you will submit work for each milestone to Gradescope rather than Canvas. Prepare a submission as a 1-page PDF that has the names and Andrew IDs of all team members as well as a link to the latest commit of your team repository in the usual format: `https://github.com/CMU-17-214/<reponame>/commit/<commitid>`. Add all team members as part of the Gradescope submission.

For Milestone A, submit `design.md` and `plan.md` to your team's repository.

For the design presentation, submit `presentation.pdf` to your repository (no Canvas or Gradescope submissions needed).

For Milestone B, have a `README.md` file and all framework and sample-plugin code.

For Milestone C, have a `README.md` file and all plugin code.

Reporting team problems

Teams do not always work conflict free and you only have limited time to get to know your team. Team members also have different abilities and backgrounds and contributions may be somewhat unequal.

We are not worried about unbalanced contributions if all team members agree on it, for example, to better make use of different expertise within a team. It is common that some team members may struggle with some task and rely more on help from others.

We however are very sensitive to *bad team citizenship*, which means that team members simply do not complete work that they agreed to. To provide evidence of bad team citizenship, you will need to have written documentation of agreements (preferably in the `plan.md` file). Not contributing at all to the assignment is an extreme form of bad team citizenship.

At the end of the assignment, you can use the team feedback form on Canvas to report bad team citizenship. We will use this information (in combination with looking at your `plan.md` file and commit logs) to adjust individual grades if needed. If you do not fill out this form, we assume everybody on your team made a fair attempt at contributing to the work and we will assign the same grade to all team members.

Evaluation

The assignment is worth 300 points with an opportunity for a total of 50 bonus points.

We expect to grade the assignment as follows:

Milestone 1: 90 points

- [ ] 10pt: A `design.md` file was committed with the four requested sections "Domain", "Generality vs specificity", "Project structure", and "Plugin interfaces"

- [ ] 10pt: The document contains a general description of the framework's domain and some example of possible data plugins and visualization plugins in section "Domain".
- [ ] 10pt: The purpose of the framework is clear. It is plausible that reuse with a framework is useful.
- [ ] 10pt: The document discusses the design decisions regarding the scope of the framework in "Generality vs specificity".
- [ ] 10pt: The design document suggests a reasonable scope that is not too narrow (i.e., allows multiple plugins) but also not too generic (i.e., has reuse value).
- [ ] 10pt: The document contains a description of the (planned) project structure and key data structures and the plugin loading mechanism in section "Project structure"
- [ ] 10pt: The document contains a description of the interfaces for both kinds of plugins and needed data structures in the section "Plugin interfaces". The description has sufficient documentation to understand roughly how the plugins are working.
- [ ] 10pt: The plugin interfaces and data structure interfaces seem plausible, allowing the kind of extensions described in the previous sections. It includes needed lifecycle methods. It does not indicate that the plugins need functionality that should be in the framework.
- [ ] 10pt: A `plan.md` with a concrete plan is submitted, identifying tasks and who's primarily responsible for each task

Design presentation: 10 bonus points

- [ ] 10pt: The team signs up for a presentation slot and presents the work. All team members take an active role in the presentation. The presentation did not run substantially over time. A `presentation.pdf` file is submitted.

Milestone 2: 150 points (40 bonus points possible)

- [ ] 50 points for the framework implementation:
  - [ ] 20pt: The framework compiles, can be started, and loads plugins. Builds are automated with maven or npm. The project can be compiled and started with maven or npm from the command line as documented in the `README.md` file.
  - [ ] 10pt: The framework allows for adding plugins without any changes to framework code; the framework is not coupled to any specific plugin implementations (during grading we will delete the plugins to see whether the framework still compiles).
  - [ ] 10pt: The framework provides a web-based GUI
  - [ ] 10pt: The framework receives data from data plugins and passes data to visualization plugins and shows the resulting visualization in the GUI
- [ ] 30 points for framework design:
  - [ ] 10pt: The framework retains control over the execution calls the plugins (inversion of control). The framework orchestrates all data exchange; plugins do not directly talk to each other. It is okay if the framework provides helper functions that the plugins call.
  - [ ] 10pt: The framework defines a format for data produced by data plugins and consumed by visualization plugins.
  - [ ] 10pt: The code reasonably follows practices for good API design as discussed in the lecture (we will look for information hiding, naming, and error handling).
- [ ] 30 points for sample plugins:
  - [ ] 15pt: For a team with *N* members, *N* data plugins are provided as samples. The plugins use different data sources and illustrate a range of different possible implementations. The plugins work.

    - [ ] 15pt: For a team with *N* members, *N* visualization plugins are provided as samples. The plugins illustrate a range of different possible implementations. The plugins work. At least one third-party library is used.
- [ ] 20 points for documentation
    - [ ] 10pt: The `README.md` describes the idea of the framework and how to start and extend it. The documentation matches the implementation.
    - [ ] 10pt: The plugin interfaces and all other classes/interfaces exposed to the plugins (e.g., exchanged data structures, input/output formats) are well documented in the code and/or readme. Documentation is sufficient to write plugins without having to understand the implementation internals of the framework.
- [ ] 5pt: Your code is generally readable and conforms to the expectation of the language.
- [ ] 5pt: You use reasonable commit messages and commits are reasonably cohesive.
- [ ] 10pt: At least some parts of the framework are tested with unit tests. Testing makes use of stubs/mocks in at least one test.
- [ ] 10 *bonus points* if your framework is selected for Milestone 3
- [ ] 20 *bonus points* if your framework does some data processing with a machine learning API (local library or web API).
- [ ] 5 *bonus points* if build and tests are automated with GitHub actions.
- [ ] 5 *bonus points* for using pull requests effectively using the Gitflow model (we will look for meaningful names, code reviews, and who merges the pull request)

Milestone 3 (Plugin teams): 60 points

- [ ] 10pt: The project with the plugins can be started with maven or npm from the command line as documented in the `README.md` file.
- [ ] 20pt: For a team with *N* members, *N* data plugins are provided. The plugins use different data sources. The plugins should be different from each other and different from the sample plugins. The plugins work.
- [ ] 20pt: For a team with *N* members, *N-1* visualization plugins are provided. The plugins should be different from each other and different from the sample plugins. The plugins work. At least one third-party library is used.
- [ ] 10pt: An `experience_report.md` file is submitted that provides feedback on the technical support provided by the framework team. The report makes a good faith attempt of reporting/critiquing the support.

Milestone 3 (Supporting teams): 60 points

- [ ] 40pt: The team was responsive to support requests and engages in support in a professional manner.
- [ ] 20pt: An `experience_report.md` file is submitted that reflects on the experience providing support. The report makes a good faith attempt of reporting/critiquing their own support experience.

Appendix A: Text sentiment analysis framework

Here we describe an example of a relatively simple but potentially interesting framework. You can follow this example, but we suggest that you pick your own idea. If you follow this example for your own submission, it is less likely that we will pick your framework as a "selected framework".

The idea is to perform sentiment analysis (how positive the tone of some text is) on different texts. The framework performs the sentiment analysis on text from different sources (provided by data plugins) and shows results in different ways (using visualization plugins). The framework performs the sentiment analysis itself, hence provides benefits for reuse. Further reuse benefits come from being able to reuse existing visualizations when just providing a new data plugin or vice versa.

There are many APIs for sentiment analysis, including offline APIs (e.g., CoreNLP) and web APIs (e.g., Google's Natural Language API).

Data plugins could provide a list of *text fragments with corresponding time stamps*. Data plugins could include:

- Twitter plugin that takes in a generic feed of Twitter messages, or a user's Twitter messages using their handle with the Twitter API
- YouTube plugin that can take in comments from different videos using the YouTube Data API
- News plugin that extracts data from various news articles and sources using the News API

The framework could provide a list of text fragments with time stamps and sentiment score to visualization plugins. Visualization plugins could include:

- Time-series graph displaying the sentiment over time
- Heat map displaying the magnitude for the most common sentiments and time of day
- Word cloud containing words from the text fragments with the highest and lowest sentiment scores (i.e. given that the data format is some list of text fragments with sentiment scores attached, this visualization could take the 10 text fragments with the highest sentiment scores and the 10 text fragments with the lowest sentiment scores, and create a word cloud containing the words from these text fragments)

Keep in mind that these are just some examples of plugins, and there are many possible other data and visualization plugins that could be created for this framework. In general, try to create a framework that can perform some kind of interesting transformation that allows for various unique plugins.