An Implementation of Integrated Visualization & Endpoint Modelling

By

Michael W. Schmidt

A thesis presented to the Honors College of Middle Tennessee State University in partial fulfillment of the requirements for graduation from the University Honors College.

Fall 2018
An Implementation of Integrated Visualization & Endpoint Modelling

Michael W. Schmidt

APPROVED:

_____________________________________
Dr. Ryan Otter
Dept. of Biology

_____________________________________
Dr. Phillip Phillips
University Honors College Associate Dean
Abstract

This creative work investigated Integrated Visualization and Endpoint Modeling with the intent to construct a software prototype. Integrated Visual Endpoint Modeling (IVEM) is a retrospective analysis tool that aids in assessing environmental damage. Specifically, IVEM is used to assess particular regions of a contaminated site using field collected data, but data are not directly comparable to one another. At present, a software implementation of IVEM did not exist. This paper details one approach to software implementation of IVEM utilizing C++, PHP, and web technologies to deliver a proof of concept. The works presented here resulted in a functional IVEM web application.
## List of Figures, Appendices, Symbols and Equations

**Figure 1**  A high level diagram illustrating the IVEM implementation process flow.

**Figure 2**  An IVEM result from processed data.

**Appendix A**  Sample data utilized for program development and testing.

**Appendix B**  File format specification found on the hosted the IVEM implementation. This is the data format the application expects to receive.

**Appendix C**  Source code for the IVEM implementation.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\pi$</td>
<td>Greek little Pi. A numerical constant which is approximately 3.14159</td>
</tr>
<tr>
<td>$e$</td>
<td>Euler’s number. A numerical constant which is approximately 2.71828</td>
</tr>
<tr>
<td>$\theta$</td>
<td>Greek Theta. A variable typically used to represent an angle in radians. A circle has $2\pi$ radians.</td>
</tr>
<tr>
<td>$\sum_{i=1}^{n} i$</td>
<td>Summation. The current expression is the sum of $1 + 2 + 3 + \ldots + n$</td>
</tr>
<tr>
<td>$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$</td>
<td>Arithmetic Mean.</td>
</tr>
<tr>
<td>$\sigma = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n-1}}$</td>
<td>Standard Deviation.</td>
</tr>
<tr>
<td>$\sigma_{\bar{x}} = \frac{\sigma}{\sqrt{n}}$</td>
<td>Standard Error of the mean.</td>
</tr>
<tr>
<td>$Area = \frac{ab \sin(C)}{2}$</td>
<td>Area of a triangle, Side-angle-side method. (Sides: $a$, $b$; Angle: $C$)</td>
</tr>
</tbody>
</table>
I. Introduction

Environmental disasters will never go away, but retrospective testing methods let us assess damage and provide insight for clean-ups. One chief issue common to both man-made and natural disasters are the resources we choose to allocate to these affected areas. Accurately evaluating affected areas involves numerous tests from several different disciplines, and carried out across different regions (Smith 2013). To challenge matters, collected data does not always agree with similar testing methods. It is often hard to reach a consensus among scientists as to which areas are the most affected since each test describes the scenario differently. The panel scientists report their findings to are not typically composed of scientists which is also problematic.

Integrated Visualization and Endpoint Modelling (IVEM) is a method of standardizing, structuring, and presenting data developed by Dr. Ryan Otter. His approach to processing and visualizing data overcomes the issues faced by scientist and regulatory bodies described above. Even though IVEM solves these problems, creating an IVE model is a time consuming and error prone task. Several statistical values such as arithmetic mean, standard deviation, standard error, and the minimum/maximum values are need for each and every test and region. Trigonometry is also used for generating plots. All of Dr. Otter’s IVE models are currently performed on a cumbersome Microsoft Excel spreadsheet.

Computers are able to solve a myriad of problems for their ability to retain large amounts of data and execute hundreds of billions of operations per second on modern hardware. Spreadsheet applications can be excellent tools for solving general
data-centric problems, but more complex problems may require a program tailored to a specific need. Computer science is an interdisciplinary field for solving computationally heavy problems in Mathematics, Biology, Physics, Chemistry, and other data driven fields.

A computer program to perform these burdensome tasks presents an excellent opportunity to apply computer science in the environmental toxicology field. IVE models require organizing data, creating statistics that describe the data, manipulation of that data, and generating plots. As of October 2015, no full implementation exists; only a complicated Microsoft Excel document which serves as a proof of concept. Creating a functional IVEM implementation applies computer science to accomplish a task in minutes rather than days. IVEM is an excellent retrospective analysis tool that should be made available beyond the currently time consuming process.

II. Objective

The aim of this project was to develop a functional program that implements IVEM and test this program’s results against data provided by Dr. Ryan Otter. In order to accomplish this task, several other smaller objectives first needed to be completed.

The program needs a method to store and handle user collected data appropriately, but in order to do so that data must first be organized. Therefore inputting, storing, and processing data needs to start with users providing well-formatted data. Providing guidelines on how to format data is a requirement in both large- (How to Choose a Data Format) and small- (Best practices for file formats) scale projects. These guidelines must provide a formal system on how this project can expect to receive data,
and the details the steps necessary for proper operation. On the application-side, data storage must consider the order and amount of data received. Adding/removing data are user-facing objectives the program should meet, but internally, statistics from this information will need to be calculated, and the data will also need to be sorted accordingly. Information will need to be passed both in and out of this internal arrangement too. This internally arranged structure chiefly needs to represent an area that underwent testing with methods to access and manipulate all of its relevant data.

Creating and calculating results from data is the end goal of this project, but without internal organization, the data won’t lead to a direct solution. It is not enough to have data arranged at a site level. The data must undergo standardization by using every recorded measurement at the test level. This is a fundamental requirement for an IVEM implementation. A “master” table that contains every recorded measurement will need to be made to calculate critical parameters. There also needs to be mechanisms to know when processed data is no longer accurate, and How? and When? these changes should occur or get updated. Finally, this table will have to be created or updated alongside each site data object on creation.

This project also aims to create an informative, yet easy to read graph. Providing a legend for the plotted regions, appropriate labels for each test, and sorted tabular results are an essential requirements for a general plot. However, unlike a general plot, IVEM expects and addresses zone ambiguity, so the plot will seek easy identification of these zones, the amount of uncertainty, and its overall significance. It is not an aim of the
project to adjust the size of the graph, but it will provide a reasonably sized output for simplicity. Third-party tools might be used to satisfy this requirement.

Finally, general principles related to application development are important too. Development for cross-platform compatibility is a requisite not only graphical output formats, but benefits users on different platforms. A successful implementation also makes provision for expansion and experimental features, so development will also seek code modularity where it can be used most fittingly. Furthermore, in the application’s creation opportunities for object oriented paradigms are hoped to be explored. One way to accomplish this goal is to keep “hard-coded” portions to a minimum. Another aim will explore C++ inheritance semantics. The project is intended to be a single use program and will not require anything outside of providing formatted data.

III. Methods & Materials

The project was approached and designed as a web service with three distinct processing layers. The first layer consists of the core IVEM implementation as a C++ back-end program. This program is the “last man” in the process chain and performs all the calculations to generate the final result. The second layer is an interface script. This interface script was written in PHP and acts as an interface layer. On one side, the script executes the back-end program passing a numerical parameter on how to proceed. On the other side, the script provides the graphical interface to the user. The third layer, a web browser, allows the user to send data files to the application via the PHP script, receive the results, and obscures the underlying complexity of the program. See the diagram below.
Figure 1. The user formats and provides data files via a web browser. The data files are received by a server that supplied the interface. The interface executes a back-end program to perform the calculations and create a report. When the program terminates, the script renders the report to the user.

Using web technologies avoided many platform-specific issues and greatly simplified the project. In this paradigm, the user is expected to use an HTML 5 conforming web browser rather than a specific device. Building IVEM this way did add an additional requirement, but a trivial one that enabled the application to be truly platform agnostic. It also ensured a uniform experience. Furthermore, this requirement affords IVEM’s to be accessible on mobile devices (i.e. field tablets) which was not initially considered.

PHP is a scripting language for creating dynamic web content and an excellent candidate for managing the web interface for IVEM. Websites such as Facebook use PHP to fetch and present information on a user-by-user basis (MacKay, et. al. 2017). For this project, PHP responsibilities were rather elementary: It handled user files, ran the backend program, and renders a page with the results. As pointed out by MacKay (et al., 2017), PHP’s syntax has the familiarity of C++ which reduced the time needed to
produce a working interface. Other technologies such as Scalable Vector Graphics (SVG) and Bootstrap were respectively used for the graphical plot and interface styling in conjunction with the PHP script.

All the code needed to facilitate the service was executed on a server. The PHP script allows a user to upload a file via the web browser and manages a counter to keep track of how many files it receives. Each file the user uploads represents an area of interest and the associated test data as stated in the guidelines. After receiving a file, each file is renamed consecutively starting at 0 (ex. “0.dat”, then “1.dat” … and so on). When the script receives the command to process the data, it executes the C++ back-end program with a numeric parameter. For example, let’s say ‘3’ was passed as a parameter. This implies that there were three regions that were investigated and three files were uploaded by the user. The backend program now executes while the script waits for the program to complete. Because the script renamed all the user files consecutively, the backend program can expect the files 0.dat, 1.dat, and 2.dat to exist (no more than “3.dat”). The back-end program executes under this assumption, processes each file, and creates a result file before it terminates. Once the PHP script receives the termination signal from the back-end program, it checks for the existence of the result file and finishes the request by sending the results. A reset function deletes the user files, result file, and sets the file counter back to zero.

The backend program performs all the mathematical operations to produce a result. Those operations must be performed on data it retains in quickly accessible memory and not the files themselves; the program parse these files into an object which
represents a dataset. The eponymously named object, a Dataset object, retains this data and provides an interface to control the data indirectly for operations such as adding, removing, organizing, querying, and manipulating those values as well as generating statistics about the data. A Site object retains Dataset objects, and Site objects interface with each other for instructions on how to manipulate their respective datasets. The underlying container to hold data in a Dataset is the STL’s list class, and the Dataset object directly operates on this list. The STL is a well vetted, open source, ISO specified collection of tools that developers should use for best practices. The STL list was selected for its flexibility in data storage.

As mentioned earlier, each file submitted represents a sampled region and its test data. The backend program’s internal representation of these sampled regions are called SampledSite objects and are created using C++’s inheritance system (Site is the parent class). As each file is parsed, the back-end program instantiates these SampledSite objects. Each object has its own properties that only pertain to the region. When the first SampledSite object is created, another special object called a MasterSite is also created using the inheritance system. As additional sites are added, their data is also merged into the MasterSite’s datasets. The MasterSite object therefore contains all data from all tests conducted at every sampled region. Once all data is read in, the data is sorted and information is needed for all SampledSite objects to normalize their data. The MasterSite object provides information on how to do so. Each SampledSite object standardizes its datasets using the
MasterSite's information. Finally, with the data in a standardized form, results can be calculated and plotted.

The C++ back-end program needs to output a Scalable Vector Graphics image as part of its duties. Scalable Vector Graphics or SVG is a technology for developing high-quality images using vectors, lines, and shapes. Most, if not all, of SVG’s shapes rely on numerical parameters. SVG’s use a Cartesian plane coordinate system to render these shapes, but the representation of this plot is better suited in a polar coordinate system. For example, if $n$ tests were conducted across different regions, then the angle for any endpoint would be some multiple of $\frac{2\pi}{n}$ on a polar system. This computes a $\theta$ value. The magnitude then becomes the test’s mean or error values.

The resulting values can be connected to form a polygon. The resulting area inside that polygon represents the degree of damage at that region. Each polygon generated shares a common origin from which it was derived from: the origin in the polar system. Using this point, the angle formed from the two neighboring endpoints ($\theta$ from above), and the magnitude values as distances, the polygon can be split into triangles. The area for these triangular segments can be computed using the trigonometric side-angle-side formula:

$$Area = \frac{Test_1 \cdot Test_2 \sin(\theta)}{2}$$

If we let the $E$ be the standardized values for $k$ tests, and let $n$ represent the number of endpoints or tests, then algorithmically calculating the area of the polygon is just the sum of these triangles:

$$A = \sum_{k=2}^{n} \frac{E_{k-1} \cdot E_k \sin\left(\frac{\theta}{n}\right)}{2}$$
This equation fails to calculate the area for a triangle between the $n^{th}$ and $1^{st}$ endpoint. The missing “slice” can be calculated and added to compute the complete area of the polygon:

$$Area = A + \frac{E_1 \cdot E_2 \cdot \sin(\frac{\pi}{2})}{2}$$

Using this approach, the area of all polygons can be computed iteratively. This process was applied for the mean as well as the upper and lower bound standard error polygons. Even though this method is an effective way to determine the area of a polygon, no actual polygon exists yet; the polygon is still a virtual abstraction of the data at this point.

Instead of using a 3rd party library to provide plotting, a less practical method was selected as an exercise. Text holding variables (called “strings”) were used to create an actual SVG image from scratch. SVG follows a format that’s similar to how HTML marks-up web pages; rather than defining a document pixel by pixel, an element or “tag” is used. Field/value pairs are used to describe the tag’s attributes. For this project Line and Path were the predominately used tags. A possible line tag could look like this:

```
<Line x1="1" y1="2" x2="10" y2="20" stroke="black" />
```

The fields x1 and y1 define the starting (x, y) coordinate for the line, and fields x2 and y2 defining the ending (x, y) coordinate. Needless to say, stroke defines the color of the line. Other field/value pairs exist and can be used to provide additional attributes. In this example, a black line runs from the point (1, 2) to (10, 20) on the canvas of the SVG. It uses the top-left corner as the origin in a Cartesian system.
A simple way to adapt IVEM’s data is to convert it to the Cartesian plane coordinate system. These conversions are fairly easy and the back-end program can do these conversions on the fly. The conversions are:

\[ x = r \cdot \cos (t \cdot \theta), \quad y = r \cdot \sin (t \cdot \theta) \]

Using this approach, an \((x, y)\) coordinate can be created from the statistical result \(r\) and a number, say \(t\) for the test number, to get the amount for the angle needed. The implementation present here used partially defined strings as templates and exploited the numerical fields in both \texttt{line} and \texttt{path} tags to create the shapes needed. For example, the line for an endpoint needs to span out using two pieces of information, an angle and a magnitude from some reference point and direction. Since these rotation and magnitude values were readily from the area calculations, \(r\) and \(t \cdot \theta\), this data was reused. The \((x, y)\) coordinate was calculated using the method described above to plot the shapes onto an SVG canvas. \texttt{line} tags were used to plot endpoints and the \texttt{path} tag was used for the legend, mean, and error bounds. A similar method was used on \texttt{text} tags for labeling. Tabular data followed the same approach, but used a table and the area results directly.

**IV. Results**

A case study using fictitious data was prepared and used to validate the IVEM implementation. Fabricated data was compiled by Dr. Ryan Otter as datasets that one might observe in the field (see Appendix A for the data used). This specifically included four contaminated zones and a fifth pre-disaster reference zone. Each zone included five different test sets:
• 7-ethoxy-resorufin O-deethylation (EROD), a liver enzyme commonly used as a pollutant marker found in pigs and fish (Zamaratskaia, Vladimir 2009). This test is measured in pmol/mg/min.

• A ground sediment test that measures the concentration of a contaminant in mg per kg of soil.

• A comet assay test that “...has applications in testing novel chemicals for genotoxicity, monitoring environmental contamination with genotoxins, human biomonitoring and molecular epidemiology...” (Collins 2004). This test measures DNA damage as a percentage.

• A gene test that measures the expression of a given gene in response to the environment (Choi and Sang 2007). The test is measured as a simple count of occurrences.

• A concentration test that includes a direct mg/kg concentration of the of a substance in fish.

Each test measured a completely different feature one might wish to observe in a contaminated region. It is important to note that while the sediment and fish test share the same unit of measurement (mg/kg), they were designed to simulate measuring different compounds with different concentration ranges. The rest of the tests have different units of measurements, but all tests cannot be directly compared against each other.

Most datasets contained ten data points per region. The sediment test included twenty data points with the “South Corner” including an additional two points. The
South Corner also included an additional two points in its gene test. The purpose of these additional data tested IVEM against uneven sampling in both test-to-test and region-to-region scenarios. For example, the sediment test contains twenty points where other tests contained ten (test-to-test: The amount of data in each test differs), and the South Corner contained two additional data points in both gene and sediment samples while others sites did not (region-to-region: The amount of data in a region differs from the others). Notice that South Corner specifically contained an additional two data in the sediment test as well which caused both region-to-region and test-to-test amounts to differ simultaneously. Finally, some data were purposely manipulated to be artificially high or low to increase variation which might result from equipment errors, bad data collection, or by sheer chance.

During development, the arithmetic mean, standard deviation, and standard error from the mean were calculated using the back-end program and checked against precomputed values for each region. With each calculation of raw data measuring up correctly, the data and descriptive statistics were standardized. If everything performed accordingly in the implementation, the program should predict normalized values for both data and statistics. Using each dataset as the final testing case, the implementation did predict accurate results. Furthermore, the program correctly predicted the area of each polygon, including the standard error boundaries from manual calculations. Previously, no working model provided a visual output so testing this project presented an opportunity to see a true to life example. Figure 2 is the result from the data contained in appendix A.
Figure 2. The graphical output that resulted from the data in Appendix A. Each zone was designed to be progressively worse than the last. Notice the large variance in The Pond: West Side’s fish test as a result from higher variation. The reference site (inter most site) barely registers a polygon due in the result due to background contaminations.
V. Discussion

The application’s functionality, interface, and graphical output are both intuitive and straightforward even for non-technical users. The resulting numerical data contains no unit of measurement, and can be thought of as a score where higher values signify worsening conditions. The numerical data is also sorted with respect to mean values and places the worst location at the top of the list. One does not need to know the statistics and math used to create the plot, but the result provides enough details to say that some amount of error can be expected both visually and numerically. Error boundaries are plotted with slightly darker colored lines surrounding the solid black line. This represents the mean measurement from the data with the standard error range filled-in between the colored boundary lines. The plot is designed such that someone can quickly gauge a test, set of tests, or a location as a whole and quickly determine its reliability despite the absence of numerical values. The color-filled error range is also semi transparent so overlapped regions are distinctly visible.

Each test ranged contained different value ranges and the best and worst test did not reach the outermost and innermost values on each endpoint. This was not expected, but after careful examination it was concluded this was an appropriate outcome. Each dataset was normalized using all data for a given test, but a dataset can only contain two extreme values because there can only be one maximum and minimum value in the entire collection. Even if the dataset contains a single entry, that entry is paradoxically both the minimum and the maximum value yet the test still contains a single extreme value.
Furthermore, if a test contains both the minimum and maximum value from the entire dataset, then it can only skew the mean and broaden the error boundary. Therefore plotted data will never reach values zero or one at the extremities of each endpoint. It was also not expected to see a sharp contrast between each site as well. Anecdotaly manipulating the data to overlap more and more still resulted with plots tending to “pull the zones apart.” This too was not expected, but again happens as data is normalized, the property mention above, and the result imposed onto each endpoint.

Moving away from a software prototype, other technologies such as the Python programming language would have dramatically simplified the development of IVEM into a more stable web application. An implementation in Python would merge both the web interface and IVEM program calculations into one cohesive application. Python has several hundred libraries (Python Package Index) that can add the functionality needed to replace the current prototype. For example, Python’s Flask library is “a micro-framework” that is better suited web framework than the current PHP script. It would combine both the web server, interface, and the app itself into the same code. SciPy and NumPy (“Sci” for science, “Num” for numbers, and “py” for Python) offers an extensive collection of science related functions ("SciPy.org" 2018), and array/matrix data structure with commonly used statistical calculations (“NumPy.” 2018). These libraries and methods were not known over the course of the projects development. However, the implementation did use C++’s Standard Template Library (STL) that follows best practices for code portability.
Future work to this implementation could remove some assumptions and limitations made at the model and code level. For example, the code made provisions for inverted data, but did not provide this functionality. In the case of inverted data, lower scores mark increasing damage (on the contrary, normal tests treat higher scores as increasing damage). Currently more testing and development needs to be conducted to remove this limitation. One assumption made is that all data scales linearly, but some tests such as the Richter Scale are non-linear. For example, a value of 3 is ten times the intensity of the value 2, and the value 2 is ten times the intensity of the value 1 in this non-linear scale. The impact of non-linear tests on IVEM is not known, and no work in this implementation exists for non-linear datasets. However, one could either scale the data before entering it, the application could correct it on the user’s behalf, or it may simply be the case IVEM can handle non-linear datasets.

Appendix B shows the current file format with the letter “A” after each test name. This “A” designation is currently a fixed parameter and symbolizes that the dataset is in an ascending order (linear and non-inverted), however other letters might symbolize different data formats:

- **I**: Data is inverted and should be treated as such.
- **T**: TAB is the delimiter for each datum.
- **CSV**: The data follows a comma separated value format.
- **L<n>**: The data is in a logarithmic scale. An optional value $n$ inside “<” and “>” could specify a custom base.
- **L<e>**: Data follows the natural logarithmic scale. Same as LN or $\log_e$
At the beginning of this article, an IVEM implementation was explored using common programming tools to develop a proof of concept. This article has shown IVEM implemented as a web application with a graphical interface which is easy to understand. Likewise the output is easy to understand as well. This was a critical part of the model because ultimately IVEM is targeted toward non-scientific entities where a simple but expressive result is needed. To date no implementation of IVEM existed, but this project offers a complete and working example that one might expect using curated but realistic data. This study did not implement some functionality, but instead presented IVEM as a usable, multi-platform, and a highly accessible application. From this prototype, future work should consider programming environments more suitable for web development and frameworks better suited for plotting graphs. A more reliable application would consider Python as a framework to solidify this modeling tool. In order for this project to be taken more seriously, real data could be used and compared with opinions from the scientific community with graphs presented to policy makers.
Appendix A

Raw data for each datasets that produced Figure 2 and associated arithmetic means, standard error, and standard deviation for each test. Standard deviation is based on the sample and not the standard deviation of the population.

**The Pond: West Side**

<table>
<thead>
<tr>
<th>Fish (mg/kg)</th>
<th>Gene (count)</th>
<th>Comet (percent)</th>
<th>EROD (pmol/mg/min)</th>
<th>Sediment (mg/kg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>409</td>
<td>42</td>
<td>269</td>
<td>75</td>
</tr>
<tr>
<td>34</td>
<td>450</td>
<td>57</td>
<td>300</td>
<td>120</td>
</tr>
<tr>
<td>26</td>
<td>400</td>
<td>50</td>
<td>290</td>
<td>80</td>
</tr>
<tr>
<td>18</td>
<td>424</td>
<td>47</td>
<td>280</td>
<td>75</td>
</tr>
<tr>
<td>17</td>
<td>438</td>
<td>46</td>
<td>270</td>
<td>110</td>
</tr>
<tr>
<td>18</td>
<td>449</td>
<td>43</td>
<td>275</td>
<td>101</td>
</tr>
<tr>
<td>29</td>
<td>418</td>
<td>51</td>
<td>285</td>
<td>89</td>
</tr>
<tr>
<td>22</td>
<td>444</td>
<td>49</td>
<td>282</td>
<td>76</td>
</tr>
<tr>
<td>14</td>
<td>431</td>
<td>47</td>
<td>276</td>
<td>156</td>
</tr>
<tr>
<td>20</td>
<td>419</td>
<td>53</td>
<td>292</td>
<td>135</td>
</tr>
</tbody>
</table>

**Fish** 21.200  **Gene** 428.200  **Comet** 48.500  **EROD** 281.900  **Sediment** 103.150

**The Pond: East Side**

<table>
<thead>
<tr>
<th>Fish (mg/kg)</th>
<th>Gene (count)</th>
<th>Comet (percent)</th>
<th>EROD (pmol/mg/min)</th>
<th>Sediment (mg/kg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>350</td>
<td>35</td>
<td>220</td>
<td>67</td>
</tr>
<tr>
<td>12</td>
<td>300</td>
<td>30</td>
<td>215</td>
<td>62</td>
</tr>
<tr>
<td>13</td>
<td>412</td>
<td>34</td>
<td>210</td>
<td>56</td>
</tr>
<tr>
<td>14</td>
<td>318</td>
<td>33</td>
<td>202</td>
<td>59</td>
</tr>
<tr>
<td>9</td>
<td>300</td>
<td>41</td>
<td>200</td>
<td>58</td>
</tr>
<tr>
<td>8</td>
<td>312</td>
<td>30</td>
<td>215</td>
<td>51</td>
</tr>
<tr>
<td>8</td>
<td>385</td>
<td>31</td>
<td>245</td>
<td>47</td>
</tr>
<tr>
<td>13</td>
<td>390</td>
<td>32</td>
<td>235</td>
<td>60</td>
</tr>
<tr>
<td>10</td>
<td>350</td>
<td>33</td>
<td>222</td>
<td>70</td>
</tr>
<tr>
<td>11</td>
<td>325</td>
<td>32</td>
<td>226</td>
<td>39</td>
</tr>
</tbody>
</table>

**Fish** 10.800  **Gene** 336.200  **Comet** 33.100  **EROD** 219.000  **Sediment** 58.800
### The Pond: North Corner

<table>
<thead>
<tr>
<th>Fish (mg/kg)</th>
<th>Gene (count)</th>
<th>Comet (percent)</th>
<th>EROD (pmol/mg/min)</th>
<th>Sediment (mg/kg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>200</td>
<td>18</td>
<td>175</td>
<td>50</td>
</tr>
<tr>
<td>8</td>
<td>210</td>
<td>15</td>
<td>150</td>
<td>34</td>
</tr>
<tr>
<td>10</td>
<td>193</td>
<td>13</td>
<td>160</td>
<td>67</td>
</tr>
<tr>
<td>11</td>
<td>180</td>
<td>20</td>
<td>165</td>
<td>34</td>
</tr>
<tr>
<td>7</td>
<td>170</td>
<td>20</td>
<td>164</td>
<td>36</td>
</tr>
<tr>
<td>6</td>
<td>112</td>
<td>19</td>
<td>163</td>
<td>39</td>
</tr>
<tr>
<td>7</td>
<td>188</td>
<td>18</td>
<td>161</td>
<td>45</td>
</tr>
<tr>
<td>8</td>
<td>170</td>
<td>23</td>
<td>171</td>
<td>42</td>
</tr>
<tr>
<td>9</td>
<td>200</td>
<td>26</td>
<td>180</td>
<td>44</td>
</tr>
<tr>
<td>10</td>
<td>180</td>
<td>28</td>
<td>172</td>
<td>42</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mean</th>
<th>Fish</th>
<th>Gene</th>
<th>Comet</th>
<th>EROD</th>
<th>Sediment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8.500</td>
<td>180.300</td>
<td>20.000</td>
<td>166.100</td>
<td>44.850</td>
</tr>
<tr>
<td>Std. Dev.</td>
<td>1.581</td>
<td>27.398</td>
<td>4.619</td>
<td>8.621</td>
<td>9.987</td>
</tr>
<tr>
<td>Std. Err.</td>
<td>0.500</td>
<td>8.664</td>
<td>1.461</td>
<td>2.726</td>
<td>2.233</td>
</tr>
</tbody>
</table>

### The Pond: South Corner

<table>
<thead>
<tr>
<th>Fish (mg/kg)</th>
<th>Gene (count)</th>
<th>Comet (percent)</th>
<th>EROD (pmol/mg/min)</th>
<th>Sediment (mg/kg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>100</td>
<td>9</td>
<td>75</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>90</td>
<td>8</td>
<td>72</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>80</td>
<td>7</td>
<td>67</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>70</td>
<td>11</td>
<td>56</td>
<td>17</td>
</tr>
<tr>
<td>2</td>
<td>60</td>
<td>9</td>
<td>54</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>75</td>
<td>7</td>
<td>58</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>85</td>
<td>12</td>
<td>45</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>89</td>
<td>6</td>
<td>80</td>
<td>15</td>
</tr>
<tr>
<td>1</td>
<td>85</td>
<td>7</td>
<td>90</td>
<td>20</td>
</tr>
<tr>
<td>1</td>
<td>85</td>
<td>8</td>
<td>45</td>
<td>1</td>
</tr>
<tr>
<td>75</td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mean</th>
<th>Fish</th>
<th>Gene</th>
<th>Comet</th>
<th>EROD</th>
<th>Sediment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2.400</td>
<td>80.750</td>
<td>8.400</td>
<td>64.200</td>
<td>8.600</td>
</tr>
<tr>
<td>Std. Dev.</td>
<td>1.350</td>
<td>10.550</td>
<td>1.897</td>
<td>15.083</td>
<td>5.205</td>
</tr>
<tr>
<td>Std. Err.</td>
<td>0.427</td>
<td>3.045</td>
<td>0.600</td>
<td>4.770</td>
<td>1.164</td>
</tr>
<tr>
<td>Fish (mg/kg)</td>
<td>Gene (count)</td>
<td>Comet (percent)</td>
<td>EROD (pmol/mg/min)</td>
<td>Sediment (mg/kg)</td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>--------------</td>
<td>-----------------</td>
<td>-------------------</td>
<td>-----------------</td>
<td></td>
</tr>
<tr>
<td>0.67</td>
<td>12</td>
<td>1</td>
<td>17</td>
<td>0.1</td>
<td>0.3</td>
</tr>
<tr>
<td>0.3</td>
<td>16</td>
<td>2</td>
<td>15</td>
<td>0.3</td>
<td>0.8</td>
</tr>
<tr>
<td>0.8</td>
<td>20</td>
<td>4</td>
<td>22</td>
<td>0.5</td>
<td>0.2</td>
</tr>
<tr>
<td>0.2</td>
<td>9</td>
<td>3</td>
<td>21</td>
<td>0.7</td>
<td>0.9</td>
</tr>
<tr>
<td>0.9</td>
<td>15</td>
<td>2</td>
<td>12</td>
<td>0.4</td>
<td>0.5</td>
</tr>
<tr>
<td>0.5</td>
<td>22</td>
<td>1</td>
<td>18</td>
<td>0.8</td>
<td>0.4</td>
</tr>
<tr>
<td>0.4</td>
<td>29</td>
<td>4</td>
<td>10</td>
<td>0.2</td>
<td>0.5</td>
</tr>
<tr>
<td>0.5</td>
<td>17</td>
<td>5</td>
<td>9</td>
<td>1.0</td>
<td>0.3</td>
</tr>
<tr>
<td>0.3</td>
<td>16</td>
<td>3</td>
<td>9</td>
<td>0.45</td>
<td>0.8</td>
</tr>
<tr>
<td>0.5</td>
<td>13</td>
<td>2</td>
<td>5</td>
<td>0.67</td>
<td>0.9</td>
</tr>
</tbody>
</table>

The Pond: Reference

<table>
<thead>
<tr>
<th>Fish</th>
<th>Gene</th>
<th>Comet</th>
<th>EROD</th>
<th>Sediment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>0.507</td>
<td>16.900</td>
<td>2.700</td>
<td>13.800</td>
</tr>
<tr>
<td>Std. Dev.</td>
<td>0.226</td>
<td>5.666</td>
<td>1.337</td>
<td>5.673</td>
</tr>
<tr>
<td>Std. Err.</td>
<td>0.071</td>
<td>1.792</td>
<td>0.423</td>
<td>1.794</td>
</tr>
</tbody>
</table>
Appendix B

Integrated Visualization & Endpoint Modeling File Specification

The Integrated Visualization & Endpoint Modeling (IVEM) uses plain text files for processing data. Text files with encodings (such as .DOC, .RTF, .ODT) will cause an error in processing and produce incorrect results or cause the program to crash. You can create input files on any operating system from commonly used programs including:

- **Windows machines:** Notepad, Notepad++.
- **macOS:** TextEdit may be used if you remove encode by pressing SHIFT + COMMAND + T or selecting Format > Make Plain Text from the global menu bar.
- **Cross Platform:** The Atom editor is recommend on any operating system if the file is saved with a ‘.txt’ extension.

When creating input files, the first line represents the name or a region, site, or area of interest and must be unique. Each file represents a site or region with test data following afterwards. Test data must have a name, followed by capital letter ‘A’ on the next line. The unit of measurement follows the next line. The name will helps the user identify the test on the plot, the ‘A’ is reserved for future use, and the unit of measurement may be:

- **Single unit:** Specified directly such as Percent, Count, ‘%’, or alpha.
- **Compound units:** No more than three units using a forward slash (‘/’ as a delimiter. For example mg/kg, or pmol/mg/kg.

The order in which test data is listed in the file does not matter – the application will resolve tests automatically. The letter ‘A’ must be placed at the end of the unit of measurement for all dataset if they are to be compared. All test data enter below unit of measurement is delimited by a new line. Data may integers or rational numbers. The word ‘END’ is placed at the end of the data to signal the end of the dataset. The line after END must be blank before specifying a new test. Here is an example for testing two location using two test:

<table>
<thead>
<tr>
<th>Filename: &quot;ID 0123-4a.txt&quot;</th>
<th>Filename: &quot;123abc.dat&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cannonville Lake, South</td>
<td>Cannonville Lake, East</td>
</tr>
<tr>
<td>EROD</td>
<td>Fish Contamination</td>
</tr>
<tr>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>pmol/mg/kg</td>
<td>Percent</td>
</tr>
<tr>
<td>13.2</td>
<td>5.2</td>
</tr>
<tr>
<td>12.3</td>
<td>4.2</td>
</tr>
<tr>
<td>. . .</td>
<td>. . .</td>
</tr>
<tr>
<td>15.2</td>
<td>1.3</td>
</tr>
<tr>
<td>15</td>
<td>4.5</td>
</tr>
<tr>
<td>END</td>
<td>END</td>
</tr>
</tbody>
</table>

Fish Contamination | EROD |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>Percent</td>
<td>pmol/mg/kg</td>
</tr>
<tr>
<td>3</td>
<td>23.2</td>
</tr>
<tr>
<td>4.2</td>
<td>14.2</td>
</tr>
<tr>
<td>. . .</td>
<td>. . .</td>
</tr>
<tr>
<td>3.8</td>
<td>4</td>
</tr>
<tr>
<td>12.333</td>
<td>17.8</td>
</tr>
<tr>
<td>END</td>
<td>END</td>
</tr>
</tbody>
</table>

The file name and extension have no restrictions. You must supply three or more tests in order to
Appendix C
// This code snippet demonstrates a simple dataset summary function in C++

#include "dataset.h"
#include "dataset-summary.h"

#include <boost/function.hpp>

DatasetSummary: DatasetSummary(Dataset * dataset) {
    m_deser = dataset ? dataset.get() : nullptr;
    d_json();
}

DatasetSummary: DatasetSummary(Dataset summary) {
    #include "dataset.h"
    #include "dataset-summary.h"

    m_deser = nullptr;
    d_json();
    m_deser = summary ? summary.get() : nullptr;
}

DatasetSummary: DatasetSummary() {
    m_deser = nullptr;
    d_json();
}

END >>>>>>>
#F39C12
#D354D0
#C032BD
#8E44AD
#2980B9
#27A6E0

colors.dat >>>>>>>
t-Int{
    } (dataset)
} (dataset)

Tествационнaя: Тествансмарь(dset, Sапмаследстйe стиe)
{
    w-range = dataset::range;
    w-tail = dataset::tail;
    w-max = dataset::max();
    w-geq = dataset::geq);
    w-median = dataset::median();
    w-q1 = dataset::q1();
    w-mn = dataset::mn()w;  
    w-sterr = dataset::std-err();
    w-stdev-s = dataset::stdev-s();
    w-stdev-p = dataset::stdev-p();
    w-mean = dataset::mean();
    w-n = dataset::size();
    w-inverted = dataset::inverted();
    w-extend = dataset::extend();
    w-name = dataset::name();
    void DatasetSummary::d-int{()
```cpp
#include <iostream>
#include <string>

#ifdef DATASETRAIT
#ifdef DATASETRAIT_H
#define dataset summary

END >>>

    w-n-norm=dataset::normalize;
    w-standartized=dataset::standartized;
    w-n-twobound=dataset::n-twobound;
    w-n-mean=dataset::n-mean;
    w-n-upper-bound=dataset::n-upper-bound;
)

void TestsummarY::tjut() {

    w-m_sitet=dataset::site
    t-jut();
}

} DataItemValues(dataset) {

TestsummarY::TestsummarY(dataset, SampedSite, Site)

    w-siters=site (site) ? 

} DataItemValues
```
class DatasetSummary
{
public:
    DatasetSummary() ;
};

using boost::function;
using std::string;

typedef DatasetSummary * DatasetSummaryPtr;

typedef std::shared_ptr< Dataset > DatasetPtr;

class Site;

#include "siteexample.h"
#include "dataset.h"

#include <boost/function.hpp>
```cpp
function<
double(dataset)> mean;

function<
double(dataset)> w-mean;

function<
string(dataset)> w-notifyed;

function<
bool(dataset)> w-tested;

function<
string(dataset)> w-testunit;

function<
string(dataset)> w-testname;

// Dataset hooks

protected:

dataset* w-dself;

{::0 : return (w-dself ? w-range(w-dself)) const

double range() const

{::0 : return (w-dself ? w-targ(w-dself)) const

double targ() const

{::0 : return (w-dself ? w-max(w-dself)) const

double max() const

{::0 : return (w-dself ? w-min(w-dself)) const

double min() const

{::0 : return (w-dself ? w-stddev(w-dself)) const

double stddev() const

{::0 : return (w-dself ? w-mean(w-dself)) const

double mean() const
```
```java
{ : -1 : return (m-stteref) + w-stteref } const
{ : ' ' : return (m-stteref) + w-stteref } const

TestSummary(Dataset, SampleDStats):?
TestSummary(Dataset, SampleDStats):?

public:
   public:
       class TestSummary
       class TestSummary

   extend the DatasetSummary class to include site specific data. //
       extend the DatasetSummary class to include site specific data. //

   {
   }
   {

   void d-ullt():
   void d-ullt():

   private:
   private:

   function<double(Dataset*) - m-range;
   function<double(Dataset*) - m-range;

   function<double(Dataset*) - m-max;
   function<double(Dataset*) - m-max;

   function<double(Dataset*) - m-min;
   function<double(Dataset*) - m-min;

   function<double(Dataset*) - m-median;
   function<double(Dataset*) - m-median;

   function<double(Dataset*) - m-picker:
   function<double(Dataset*) - m-picker:
```

function(w-score, m-score, w-mean, m-mean, w-stdDev, m-stdDev) {  
  return (w-score - m-score) / (w-stdDev - m-stdDev);  
}

protected:

{  
  w-score:
  {  
    if (w-score < m-score) {  
      return w-score;  
    }  
  }
  {  
    if (w-score > m-score) {  
      return m-score;  
    }
  }  
  {  
    return w-score;  
  }  
}
// Constructors & Destructor

// Implementation

using std::abs;
using std::setprecision;
using std::setw;
using std::cout;
using std::endl;
using std::cout;
using directives

#ifndef TYPES_H
#define TYPES_H

#include "dataset.h"

#endif

#include "stream"  
#include "stringstream"
#include "math"
#include "iomanip"
include <string>
#include <iostream>
#include <cmath>
#include <iomanip>
#include <sstream>

#include <sstream>
setprecision(precision); //

#include <sstream>
setprecision(precision);

#include "dataset.cpp"

END

#ifndef DATATSUMMARIES_H
#define DATATSUMMARIES_H

*/
*
{

*/
```java
@override
data set = to-copy.data; 
size = to-copy.size;
 inversión = to-copy.inversion;

if (dataset:dataset:inverse) {
  if (dataset(type: SELF)) {
    state-stats();
    init-stats();
  }
}
```
node::value = d;
} for (double d : data)
    node::norm_value = 0;

w-norm_type = SELF;
w-inverted = inverted;

int starts();

unit_setter(unit);
name_setter(name);

Dataset::node(node);
    vector<double>::const_iterator it;
} Dataset::Dataset(vector<double> data, string name, string unit, bool inverted)

{  
    
    w-norm_type = to-copy, w-norm_type;
w-inverted = to-copy, w-inverted;

    state_starts();
    init_starts();

    unit_setter(to-copy, unit);
    w-name = to-copy, w-name;
ret-String + '\n' + ret-String + '\n' + ret-String + '\n' + ret-String + '\n' + ret-String + '\n' + ret-String + '\n' + ret-String + '\n' + ret-String + '\n' + ret-String

if (m-units < 2)
{
    ret-String + m-units[2];
    ret-String + m-units[1];
    ret-String + m-units[0];
}

string ret-String = m-units[0];

} dataset::unit() const

Public Observers

{
    }

dataset::~dataset()

{
    
    rank();
    normalize();
    sortdata();
    state-stats();
    w-Size = m-data.size();

    
    m-data.push-back(node);
cout << endl;
cout << setprecision(6) >> std::endl;

} // ST Dev:

} // setw(7) >> mean()

} // setw(12) >> Mean:

} // ST Dev:

} // setw(14)

} // setw(6)

} // setw(6)

} // setw(6)

} // setw(6)

} // setw(6)

} // setw(6)

} (data: m-data:

for (node data : m-data:

cout "" Value 

} // end:

} // end:

if (m-normtype == EXTERNAL)

} // end:

void dataset::print()

return ret-String

} // end:
```cpp
{ return m_basist;

    { state_basist = False;
      dataset::baset();
      } (state_basist)
    } (dataset::baset())

{ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~` ~` ~` ~` ~` ~` ~` ~` ~` ~` ~` ~`
      cout >> "\nN. U. ERR: "
      cout >> setw(12) >> "Range: "
      cout >> setw(12) >> "Range: "
      cout >> setw(12) >> "Range: "
      cout >> setw(12) >> "Range: "

{ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~` ~` ~` ~` ~` ~` ~` ~` ~` ~` ~` ~`
      cout >> setprecision(6) >> dataset::norm-mean();
      cout >> setprecision(6) >> dataset::norm-mean();
      cout >> setprecision(6) >> dataset::norm-mean();
      cout >> setprecision(6) >> dataset::norm-mean();
      cout >> setprecision(6) >> dataset::norm-mean();

{ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~` ~` ~` ~` ~` ~` ~` ~` ~` ~` ~` ~`
      cout >> setprecision(6) >> dataset::std-error();
      cout >> setprecision(6) >> dataset::std-error();
      cout >> setprecision(6) >> dataset::std-error();
      cout >> setprecision(6) >> dataset::std-error();
      cout >> setprecision(6) >> dataset::std-error();
```
{ if (state_q1) {
  if (state_q1) {
    double dataset::q1() const
  }
}

if (state_q1) {
  double dataset::q1() const
}

if (state_q1) {
  return abs(q5() - q1());
}

double dataset::q4() const

{ return abs(q5() - q1());
}

double dataset::q4() const

{ return abs(q5() - q1());
}

double dataset::q4() const

{ return abs(q5() - q1());
}

}
if (m-inverted) {
    return maximize(m-inverted);
}

double Dataset::maximize() const
{
    if (m-inverted) {
        return maximize(m-inverted);
    } else if (state-max) {
        return maximize(m-inverted);
    } else if (state-max) {
        return maximize(m-inverted);
    } else if (state-9) {
        return maximize(m-inverted);
    } else {
        return m-9;
    }
}
if (state-
max
(s))
{
double state-
max
(s); if (state-
max
(s))
{
retu
rn state-
max
(s);
}
return w-
max
;
}
double Dataset::mean() const {
    return mean() + std::error();
}

double Dataset::mean() const {
    return mean() - std::error();
    if (m_inverted)
}

double Dataset::up::mean() const {
    return m-sum;
    if (state-sum)
}

double Dataset::std::error() const {
    return m-sum;
    if (state-sum)
}

double Dataset::std::error() const {
    return state-error = false;
    if (state-error)
}
double dataset::norm-93() {
} //

double dataset::norm-94() {
} //

return abs(value) / w-normbasis;

double value = median() - w-normval;
} ()

double dataset::norm-95() {
} () //

return abs(value) / w-normbasis;

double value = median() - w-normval;
} () //

double dataset::norm-96() {
} () //

return abs(value) / w-normbasis;

double value = mean() - w-normval;
} () //

double dataset::norm-97() {
} () //

return mean() - dataset::std-error() !

else

return mean() + dataset::std-error() !

if (w-inverse)
void normalize();
    m_normtype = ref-value;
    m_normval = ref-value;
    } 

public:

    void Standardize(double ref-value, double ref-basis);

    void Normalize();
    m_normtype = SELF;

    } ;

    double abs(double valued) {
        return abs(value - m_normval);
    }

    double lowerBound() {
        return abs(value - m_normval);
    }

    double upperBound() {
        return abs(value - m_normval);
    }
rank():
state-stats():
    w-size = w-data.size();
    {{
        const Datatnode & a, const Dataatonode & b)
            return a.value > b.value;
    }
    w-data.sort({})
this-w-data.push-back(*iter):
    for (; iter = -merge.w-data.end(); iter++)
        [lower-bound<merge]:
            const_iterator iter = -merge.w-data.begin();
        } (const merge:const Dataatonode & merge)
    void dataset::merge(const Dataatonode & merge)
    { norm-heaper();
    state-stats();
    rank();
    w-size++;
    () (sortdata)
    w-data.push-back(tns);
    { TODO: Insert in place to avoid whole set sorting: O(N) vs O(n log n)
        this.norm-value = 0;
        this.value = to-insert;
        Dataatonode tns;
    } (const insert:double to-insert)
    {
w-uintcnt++;
} (., ., == reader ==
    continue;
(. , == reader ==
    reader = uint.at(1);
} (for (size t i = 0; i < uint.size() i++;

    w-uintcnt = 1;
    char reader;
} void dataset::unit-setter(string unit)
{
    iter-rank = rank;
    for (iter = w-data.end(); iter++ rank++;
         rank = 1;
    iter = iter->begin();
    } ()
} void dataset::rank()
{
    // NI
} void dataset::remove(unit pos)
{
    norm-hepler();
```c
void dataset::calc_stats()
{
    if (m_size == 0)
    {
        c_stderr();
        c_stderr();
        c_stddev();
        c_stddev();
        c_g();
        c_g();
        c_median();
        c_mean();
        c_sum();
        c_baseStats();
    }
    void dataset::calc_stats();
}

void dataset::calc_stats()
{
    w multiplier - 1 + Reader;
    break;
    if (w multiplier > 3)
    {
        continue;
    }
```
void Dataset::stddev-s() const
{
    [m, e] n = s m / s m;
    e l s e m = n d:
    i r e r t r a t e \_ t e r a t o r i t e r = w - d a t a . b e g i n()
    } 470
    } 478
    d e t e [ ] n u m s :
    s q r t ( t e m p s u m ) :
    ( )\_ t e m p s u m = w - s i z e - 1
    [ t ] n u m s = [ i t e r = 0; i t e r < w - s i z e; i t e r ++ ]
    ( )\_ t e m p s u m += n u m s [ i t e r ] * ( w - m e a n ) ;
    n u m s [ i t e r ] = p o w ( t e m p s u m ) ;
    f o r ( i t e r = 0; i t e r < w - s i z e; i t e r ++ )
    ( )\_ c - m e a n ()
    d o u b l e t e m p s u m = 0;
    d o u b l e n u m s = n e w d o u b l e [ w - s i z e ]
    \{ w - m e a n = s t d : n u m e r t c \_ t m t t s \_ d o u b l e \_ t n a t e : n a n ;
    } 468
    e s e w - m e a n = s u m ( ) / ( w - s i z e ) ;
def dataset():
    data = []
    dataset = []
    for i in range(size):
        if i % 2 == 0:
            dataset.append(data[i])
        else:
            dataset.append(data[i])
    return dataset

def main():
    data = [1, 2, 3, 4, 5]
    dataset = dataset()
    for i in range(len(dataset)):
        print(dataset[i])

if __name__ == '__main__':
    main()
for (int k = 0; k < n; k++)
    
    m=q1 = temp-q1 + (iter-value) * .75;
    
    if (plus-one-form)
        
        n = (m-size - 3) / 4;
        
        if (n <= m-size)
            
            plus-one-form = true;
            
            n = n - 1;
            
            if (n <= m-size - 4)
                
                if (m-size % 2)
state_q1 = false;
{
    m_q1 = (temp_q1 /= 2);
    temp_q1 += iter->value;
    iter++;
    temp_q1 = iter->value;
    iter++;
    for (int k = 0; k < midpoint; k++)
    {
        m_q1 += iter->value;
    }
    iter++;
    for (int k = 0; k < midpoint; k++)
    {
        m_q1 += iter->value;
    }
    m_q1 = (temp_q1 /= 2);
}
else { // odd n on lower half
    even length //
    if (midpoint % 2 == 0)
    {
        m_q1 = temp_q1 + (iter->value) * .25;
    }
    iter++;
    temp_q1 = (iter->value) * .75;
}
```java
for (int k = 0; k < n; k++)
    iter++; // Appy weights

iter++; //
for (int k = 0; k < n; k++)
    iter++;

if (plus-one-form)
    n = (m-size - 1) / 4 + 3 form //
else { // swapped
    plus-one-form = true;
    n = (m-size - 3) / 4 + 3 form // of the form 4n+1\(^2\) (then mod 4 == true)
} // odd length total set //
```
\[
\begin{align*}
\theta &= \text{w-summary-p} \\
\theta &= \text{w-summary-s} \\
\theta &= \text{w-summary} \\
\theta &= \text{w-summary} \\
\theta &= \text{w-g3} \\
\theta &= \text{w-4l} \\
\theta &= \text{w-wmedian} \\
\theta &= \text{w-mean} \\
\theta &= \text{w-tq4} \\
\theta &= \text{w-basis} \\
\theta &= \text{w-normal stats} \\
\theta &= \text{w-norm} \\
\theta &= \text{w-normal stats} \\
\end{align*}
\]
```java
{ // 662
    this.value = new int[data.sort().length()] // 663
    const Database & lhs // 665
    w-value > // 667
    return this.value; // 669
} else // 669
    w-value > // 672
    return this.value; // 674
    // m-inverted()
} () // 675
void Dataset::sortData() // 676
{
    state-sum = true; // 678
    state-stderr = true; // 679
    state-stdev-s = true; // 680
    state-stdev-p = true; // 681
    state-99 = true; // 682
    state-91 = true; // 683
    state-mn = true; // 684
    state-max = true; // 685
    state-median = true; // 686
    state-mean = true; // 687
    state-basis = true; // 688
} void Dataset::state-stalls() // 691
{ // 691
    w-sum = 0; // 692
    w-stderr = 0; // 693
    // 694
} // 694
```
void Dataset::c-basis()
{
    // Code snippet
    w-basis = Dataset::range();
    }
}


d::norm_value = d::value - w-normbasis;
    for (Variable d : w-data)
    }

if (m_basisspace) {
    m_basisspace = Dataset::rangen();
} else {
    c-basis();
}

else {
    if (m_inverse) {
        m_inverse = inverse();
    }
}

if (m_normmax) {
    c-max();
}

if (m_normmax) {
    Dataset::norm-helper();
}
}
```cpp
using std::string;
using std::vector;
using std::list;
using std::shared_ptr;
using Statements

#include "types.h"
#include <vector>
#include <list>
#include <memory>
#include <string>
#include <internal/processor/directives>

#include <internal/processor/directives>

#define DATASET

#include <internal/dataset.h>

END

```

Returns mean + std error
            double up-mean-err() const;
            AmendmentObservers //

Returns unit of dataset
            string unit() const;
            int td() const {return m-td;}

Returns name of dataset
            string name() const {return m-name;}
            bool standardized() const {return (m-support == EXTERNAL);}
            bool inverted() const {return m-inverted;}

Returns num of elements in list
            int size() const {return m-size;}
            bool empty() const {return (m-size == 0);} //
            bool strictObservers // Strict Observers
                {!dataset(itr+td, name, name, string unit, bool descending = true);}

dataset(const dataset& copy) : dataset(*copy)
    {//
        dataset(const dataset& copy)
            dataset();
            ~dataset();

    public:
        class Dataset
    

typedef dataset* datasetPtr;
typedef std::shared_ptr<dataset> datasetPtr;
struct DatasetNode;
    class Dataset;

Forward definitions and typedefs //


```c
{ :: do return abs(norm-g3) - norm-g1) { :: do return norm-g3) :
  do return norm-median() :
  do return norm-mean() :
  { :: do return n-tower-bound() :
    do return n-upper-bound() :

do return range() { :: do return absmxwnum() - mxwnum - max - min

do return basst-unit: Range / 100

Modifies start values //

{ :: do return order 09 - 01
  do return highest value //
  do return third quartile //
  do return median //
  do return first quartile //
  do return smallest value //

Returns std error of the mean //

Returns std dev of a population //

Returns mean - std error //

Returns mean //

//

void print() :

Observer Mutators //

double std-error() const;

double std-dev() const;

double std-dev(p) const;

double mean() const;

double mean-error() const;
```
mutable bool state-9;
mutable bool state-8;
mutable bool state-7;

private:

mutable bool state-stdev-s;
mutable bool state-stdev-p;
mutable bool state-mean;

State Data Flags //
void calc-stats() {;//
void standardize(doubke ret-value, double ret-basis);;
void normalize();;

void remove(int position);//
void merge(shared_ptr<Sort>& merger);;
{; 
void merge(const dataset &merger);;
{;

void insert(const dataset &insert) {;

void insert(double to-insert);;

Set the unit of the dataset //
void unit-setter(String unit) {;
{;

void unit-setter(int -td) (td = -td;
{;

void unit-setter(String name) {;
{;

void name-setter(String name) {;
{;

Set the name of the dataset //

Mutators //

double sum() const;
}
<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>773</td>
<td>double w-tfr; mutable double w-g3; mutable double w-median; mutable double w-q1; mutable double w-stderr; mutable double w-stdev-s; mutable double w-stdev-p; mutable double w-mean; mutable double w-mtrim; mutable double w-mtrimn; mutable double w-mtrimmax; mutable double w-mwbasists; double w-nmbasists; double w-normval; double w-basists; unsigned int w-tid; int w-unincrement; int w-size; bool w-inverted; mutable bool state-sum; mutable bool state-basists; mutable bool state-max; mutable bool state-g3; mutable bool state-g4; mutable bool state-g5; mutable bool state-g6; mutable bool state-g7; mutable bool state-g8; mutable bool state-g9; mutable bool state-g10</td>
</tr>
</tbody>
</table>
F|ag Resetter //

# The dataset //

Stores compound units (mg/L/hr) // The name of type of test conducted // 796

string m-units[] = ["mg/L"];

mutable double m-sum; // 794

std::unordered_map<std::string, double> m-data; // 798

NORM = m-normYpe; // 797

uint stats(); // 813

state stats(); // 812

sum stats(); // 811

void stats(); // 810

void rank(); // 608

void c-max(); // 808

void c-g(); // 798

void c-median(); // 808

void c-gt(); // 798

void c-mn(); // 808

void c-stder(); // 808

void c-stdev(); // 802

void c-stdev(); // 804

void c-stdev(); // 800

void c-stdev(); // 799

Helper Methods //

// 798

ListDataset m-data; // 798
include "box-whisker.h"
#include "plotter.h"
#include "integrator.h"
#include "stemmaster.h"
#include "summary.h"
#include "table.h"
#include "dataset.h"
#include "dataset-summary.h"
#include "types.h"

user_defined_directives

#include <time.h>
#include <string>
#include <cassert>
#include <fstream>
#include <iostream>
#include <vector>
#include <iostream>
#include <preprocessor_directives

def main.cpp

END

endif

{ void sortdata() {
  void norm(data());
}
true_configure();
Integrator_interface();
    begin = clock();
    
vector<string> paths;

double time-spent;
    clock-t begin, end;
}() // int main

};
\[
\text{if (row &lt; w-rows || row == 0)}
\]

\[
\text{\{}
\]

\[
\text{row-table[set-row(\text{row, const IST-string} &amp; entries)}
\]

\[
\text{\{}
\]

\[
\text{w-table} = [\text{0}]\text{ headings:}
\]

\[
\text{\{}
\]

\[
\text{\} (row-table: set-header(\text{const IST-string} &amp; \text{headings})
\]

\[
\text{\} (row-table: ~HTMLTable)
\]

\[
\text{\} (HTMLTable)
\]
```cpp
#include <list>
#include <vector>
#include <string>
*

#include "HTML-TABLE-H"
#include "HTML-TABLE.H"
void HTMLTable::strng_entry(string &strng, entry &entry)
{
    m-table[row] = entries;
    else
        return;
```
private:
{
    out += \"</tr>\n\";
    out += temp;
    
    \'(sod = temp.substr(0, sod_num)
        \'(\');\n    } (sodu::string str = i; sod)
        \'(\');\n    sod = temp.substr(0, sod_num)
    \'(\');\n    out += \"</class=num-result\nout.clear();

    std::string temp = to-string(num)
    \'(\');
} (template <typename T, std::string entry, string & out>)

void set-row(const row, const list<string> & entries);
void set-header(const list<string> & headings);

~HTMLTable();
}
HTMLTable(int rows, int cols, string id,t)
{HTMLTable();
} ()

public:
class HTMLTable
{private:
using std::cout;
using namespace std; //

#include "sttemplate.h"
#include "sttemplate.h"
#include "porter.h"
#include "tagger.h"
#include "tagger.h"
user defined directives //

#include <command>
#include <match.h>

sys directives //

#define USE_MATH_DEFINES

int main() {  //
    string m_buffer;
    vector<string> m_table;
    int m_cols;
    int m_rows;
    bool m_headerset;
```cpp
m-master = s-MasterSitePtr(new MasterSite(##site-iter));

sites-iter = m-sites.begin();
} Constructor MASTER Reference. //
{
    sampled-site, reset()
    {
        m-sitecount++;

        m-sites.push-back(sampled-site);

        m-sitesnames.push-back(sampled-site-name);
    } if (find-site(sampled-site-name) == -1)
sampled-site = s-SampleSitePtr(new SampleSite(path));
} for (const string path : paths)
    state-conf = true;
    m-sitecount = 0;

shared_ptr<SampleSite> sampled-site;
vector<SampleSitePtr> const-iterator sites-iter;
} } Intergrator::Integartor(const vector<string> paths)
master Iterator from those sites. //
holds All the sites, the datasets of those sites, and creates a
Take a vector of File Paths, create the Intergrator object that

using std::setprecision;
using std::strw;
using std::sni;
using std::endl;
```
} (site-name) = site-name();
} (const s-SamplerMethod site : m-sites)
} const void Integrator::print-site(string site-name) const
{
    site->print-tests();
    for (const s-SamplerMethod site : m-sites)
    } const void Integrator::print-sites() const
{
    m-master.reset();
    m-sites.clear();
} (const Integrator::~Integrator());
{
    assign-tips();
    m-usable-tips.reserve(m-master->test-count());
    
    m-master->add-data(sites->iter());
    while (sites->iter = m-sites.end())
        sites->iter++;
}
double radius = refer->minRadius();

dataSetRef.refer = m-master->getTest(name->setter);

for(const DataSetPtr test : m-master->getMasterSets())
    data_set::iterator site-iter = m-sites.begin();
}

void data_set::iterator site-iter, string name->setter())
    data_set::iterator site-iter = m-sites.end();

{ std::data_set::iterator site-name, test-name();
    for (const DataSetPtr test : m-master->getMasterSets())
        std::data_set::iterator site-name;
}

// Mutators

{ return m-master->test-count();

    configure();
    if (state-config)
        int iterator::get-testcount()()

    { {
        break;
        site-print-tests();
    }
```cpp
void Intersector::normalize(string site-name) {

    vector<SampleSiteIterator>::iterator iter = site-iter;
}

void Intersector::normalize(string site-name, string name-setter) {

    vector<SampleSiteIterator>::iterator iter = site-iter;
    if ((*site-iter)->name() == site-name) {
        while (site-iter != m_sites.end()) {
            if (*site-iter == m_sites.begin()) {
                site-iter++;
                break;
            }
            (*site-iter)->name-name.set();
            site-iter++;
            break;
        }
    }
}

void Intersector::normalize(string site-name, string name-setter, refer name-setter, r-value r-basis) {

    double r-basis = refer-r-basis();
}
```


```c
{
    state-conf = true;
    
    m-stlecount++;
    OR...
    // you can handle this dynamically...

    1000: keep a list of names and only restandardize those sites.
    THAIS A BIG PROBLEM!
    if sites are standardized, they aren't now!
    m-master->add-delta(state);:
    m-stlenames.push-back(state->name();)
    m-stlenames.push-back(state;)
}

1010 (intergrator: add-stle(s-Smithedstle) site)
}

1050 void Intergrator::add-stle(new-Smithed)
{
    s-Smithedstle new-Smithed(path)
    void Intergrator::add-stle(string path)
    }

1090
c{1091 (intergrator->normalizater();
    (intergrator->normalizater();)
    if (name == ()
    for (iter = m-sites.begin(); iter != m-sites.end(); iter++)
```
void Integrator::configure()
{
    state-conf = true;
    
    m-statecount = m-statecount-1;
    m-states[name].reset();
}

void Integrator::remove-state(string name)
{
configure()
    if (state-conf)
} ()
void Interpreter::show-results()
{
    state-conf = false;
    calculate-site(*test)
    for (iter = m-teres-test.begin(); iter != m-teres-test.end(); iter++)
        site-iter++;
        {
            m-teres-test.push-back(*site-iter)-get-payloads
            ();
        } (*state-iter)-generate-payloads()
    } (0 < (state-iter)-standardize-count())
    {
        (*state-iter)-standardize(test-id, r-val, r-basis);
        m-master-step-reference(test-id, r-val, r-basis);
    } (test-id <(*ref-iter)-ref-test-id)
    ref-iter *= (ref-iter)-m-master-references.begin();
    ref-iter += m-master-references.end();
    while (site-iter = m-teres.end)
    {
        determine-coverage();
    }
{
    test++;
    {
        *(test-<int>enumerator(t, name))
        *(test-<int>enumerator(t))
    }
    for (int i = m-states.end(); i != m-states.end(); ++i) //
        get name based on td
    }
    for (int t = 0; t < test-count; t++)
        string name;
        int test-id = 0;
        int test-count = m-master->test-count();
        vector<vector<int>> m-states;
        void Integrator::assign-tds()
            //
            {
                w-protter = nullptr;
                delete w-protter;
                m-protter->prot-results();
                r(w-master)();
                m-master->master-size();
                m-theta,
                r(w-results, w-master);
                m-protter = new protter(w-results);
void Integrator::set_theta(int points)
{
    m_theta = (2 * M_PI) / points;
    m_sin_theta = sin(m_theta);
    
    ResultsPtr site_result;
    site_name = iter->site_name();
    site_id = str - iter->site_id();
    lag_u = str - iter->upper_error();
    lag_l = str - iter->lower_error();
    
    if (site_result)
    {
        site_results.end();
        site_results.begin();
        
        Int site_id;
        string site_name;
        
        vector<Results> site_results;
        
        ResultsPtr site_result;
        site_name = iter->site_name();
        site_id = str - iter->site_id();
        lag_u = str - iter->upper_error();
        lag_l = str - iter->lower_error();
    }
}
```

1180 {  
1181     state-result = null;  
1182     delete state-result;  
1183     w-results.push-back(state-result);  
1184 }  
1185
1186 state-result = new Results(state-name, state-id, u-area, w-area, t-area);  
1187 // That 0 needs to be fixed  
1188
1189 t-area += 0.5 * t-ag-t * str-m * w-m sine-theta;  
1190 w-area += 0.5 * t-ag-w * str-m * w-m sine-theta;  
1191 u-area += 0.5 * t-ag-u * str-n * w-m sine-theta;  
1192
1193 }  
1194 t-error = t-error + t-m = mean(t-error);  
1195 t-ag-t = t-ag-t;  
1196 t-ag-w = t-ag-w;  
1197 t-ag-u = t-ag-u;  
1198  
1199 while (t-error > test-results.end) {  
1200     partial area calculation for the polygon  
1201     }  
```
```cpp
tf (^id == *iter)
for (iter = w-unstable-tids.begin(); iter != w-unstable-tids.end(); iter++)
    vector<null>::const_iterator iter;
}

bool Iterator::isUnstable(tdf const) const
{
    set-theta(w-unstable-tds.size);
    {
        state-iter++;
        {
            state-iter = df-iter;
            if (state-iter = w-unstable-tds.erase(state-iter))
                return true;
            while (df-iter = w-unstable-tds.begin())
                while (state-iter = w-unstable-tds.end())
                    state-iter++;
        }
        return false;
    }
}

master-iter++;

master-iter = w-unstable-tds.push_back(master.iter);

while (master-iter = w-unstable-tds.begin())
    while (master-iter = w-unstable-tds.end())
        master-iter++;
}

vector<Sample> &getMaster() const
{
    return m-master->getMaster();
}

vector<null>::const_iterator begin() = w-unstable-tds.begin();
vector<null>::const_iterator end() = w-unstable-tds.end();

void Iterator::determineCoverage(){};
```
```cpp
#define INTTEGRATOR_H
#define INTTEGRATOR_H

END >>>>>

/* {

} */

void Integrator::maximize_accuracy()
{
    return -1;
    ++
    ++
    return 1;
    if (*iter == name)
        ++
        while (iter != m_names.end())
            ++
            List<string>::const_iterator iter = m_names.begin();
}

int Integrator::find_status(const name) const
{
    return false;
    return true;
```
class Integrator

typed Plotter * PlotterPtr;

typed shared_ptr<Integrator> _IntegratorPtr;

typed Integrator * IntegratorPtr;

class Plotter;
class Integrator;

using std::shared_ptr;
using std::string;
using std::set;
using Statements //

#include "types.h"
#include "sttemple.h"
#include "sttemaster.h"
#include "site.h"

User Defined Directives //

#include <set>
#include <algorithm>
#include <memory>
#include <iostream>
#include <math.h>

System Directives //

#define USE_MATH_DEFINES
private:
  void configure() {
}

void remove_site(string name) {
}
void add_site(string path) {
}
void normalize(string name) {
}
void standardize(string name, string name-setter) {
  Mутаторs
}

int get-testcount() {
  return m-testcount;
}
int show-results() {
}

void print-master() {
  w-master->print-tests();
}
void print-sites() {
  const observers //
  ~тестратор()
}

тестратор(const vector<string> paths) {
}

тестратор & тестратор (const std::vector<string>& paths) {
}

public:

void determine-coverage() {
    // use all (the most) sites
}

void calculate-site(vector<test-results> & test-results);
void set-theta(test-points);
void assign ids to sites and tests
    // helper methods
}

PlotterPlotter vector<
vector<
    vector<
        vector<
            vector<
                vector<
                    vector<
                        vector<
                            vector<
                                vector<
                                    vector<
                                        vector<
                                            vector<
                                                vector<
                                                    vector<
                                                        vector<
                                                            vector<
                                                                vector<
                                                                    vector<
                                                                        vector<
                                                                            vector<
                                                                                vector<
                                                                                    vector<
                                                                                        vector<
                                                                                            vector<
                                                                                                vector<
                                                                                                    vector<
                                                                                                        vector<
                                                                                                            vector<
                                                                                                                vector<
                                                                                                                    vector<
                                                                                                                        vector<
                                                                                                                            vector<
                                                                                                                                vector<
                                                                                                                                    vector<
                                                                                                                                        vector<
                                                                                                                                            vector<
                                                                                                                                                vector<
                                                                                                                                                    vector<
                                                                                                                                                        vector<
                                                                                                                        vector<
                                                            vector<
                                                        vector<
                                                    vector<
                                                vector<
                                            vector<
                                        vector<
                                    vector<
                                vector<
                            vector<
                        vector<
                    vector<
                vector<
            vector<
        vector<
    vector<

double w-site-theta;
double w-theta;
tot w-site-count;

bool state-cont;

using std::sort;
using std::iostream;
using std::fstream;
using std::to_string;
using std::endl;
using std::cerr;

#include "types.h"
#include "svg.h"
#include "box-getsker.h"
#include "integrateor.h"
#include "pointer.h"

#include <windows.h>

#include "algorithm"
#include "serialize"
#include "stream"
#include "string"

/* defines FILENAME_MAX */
defines stdio.h

#include "pointer.cpp"

END >>>>>
#endif

if (!find-site(string name) const { // search the vector of site names, 1 if not found

...
vector<vector<int> > testerResults = tester.returns();
const char * out-filename = out-filename.c_str();
}
void Pointer::plotResults(string out-filename)
{
	plot-coords();

this->mean() ;

m-results[] = tester-returns();

m-endpoints = endpoints-cnt;
}

m-origin-y(300) ,

m-origin-x(300) ,
const MasterStreighten master
in endpoints-cnt,
double theta,
results* results,
plotter(vector<vector<int> > testerResults, * tesres-ad]list,
draw-site(buffer, *color, site);
  color = m-colors.begin();
  if (color == m-colors.end())
    }
  for (const vector<results> & site : *m-end-list)
    SVG::header(buffer, 600, 600);
    START OF SVG

    END OF RESULT TABLE //
    buffer.clear();
    out-fffie >> buffer;
    result-table-row(buffer);
    result-table-row(buffer);
    result-table-row(buffer);
    START OF RESULTS TABLE

    buffer + "<div class=wrapper>

    h1=header(buffer);
    
    
    return
    err >> "Could not create " >> out-fffie-range >> ". " >> end;
    }
    if (out-fffie.good())
      out-fffie.open(out-fffie-name, CSR);
      ofstream out-fffie;
      string buffer = string();
    
    vector<STRING> const-fffieator = m-colors.begin();


void pointer::html::header(string &html)
{
    static string html = "<!DOCTYPE HTML>
";
    Create necessary HTML specs
    for (*cooler, cooler::site[0].site-name());
}

// Helpers: Output Formatting

// NULL, NULL, SW-SHOWNORMAL:

// HELODESCRIPTION(NULL, "open", out->filename-cst,

out->file.close();
buffer->clear();
out->file >> buffer;
END OF HTML

// END OF SVG
buffer->clear();
out->file >> buffer;
buffer += "</div>"

SGV::footer(buffer);
draw-axiss(buffer);

colors++;
draw-legend(buffer, *cooler, site[0].site-name());
```cpp
        ) { (.*), )
        pos = temp.
        return;
    } )
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
    return;
```
void PrintTable<str_t>(result_table &result) {
    "<tr>
    " +âm
    " </tr>
    "
}

"<tr>
    " +âm
    " </tr>
    "
{
    "<tr>
    " +âm
    " </tr>
    "
    results++;
    "<tr>
    " +âm
    " </tr>
    "
    for (int i = 0; i < result.size(); i++)
    {
        if (result[i].start == result[i].end)
        {
            pos = 3;
            pos = str[result[i].start] = pos;
        }
        else if (result[i].start < result[i].end)
        {
            pos = str[result[i].start] + str[result[i].end] - 1;
            pos = pos;
            pos = pos;
        }
        else
        {
            pos = pos;
            pos = pos;
            pos = pos;
        }
    }
"<tr>
    " +âm
    " </tr>
    "
}
<html>
<body>
  <div id="content">
    <pre>
      SVG: text(buffer, table-iter-&gt;test-name(), end-x, end-y, "\n\n")

      proc(m-ORIGIN-X, m-ORIGIN-Y, m-radius + 15, theta, end-x, end-y) {
        for (int i = 0; i < m-endpoints; i++)
          theta = 0;

        {
          theta = m-theta;
          stroke-width=
        }

      proc(m-ORIGIN-X, m-ORIGIN-Y, m-radius, theta, end-x, end-y) {
        for (int i = 0; i < m-endpoints; i++)
      }

      vector<testresults>::const_iterator table-iter = tests.begin();
      vector<testresults> & tests = m-adp-1st[0][0];

      double theta = 0;
      double end-y;
      double end-x;

      void printf::draw-axes(string & buffer) {
        Helpers::Macro SVG generator methods
      }

      "<html>
      
      "<head>
      "
      "<body>
      "
      "</body>
      "</html>
    
    </pre>
  </div>
</body>
</html>
extra + " stroke-width=1.0"
extra + color:
extra + fill-opacity=0.45, stroke=
extra = "fill"
extra + color:
extra = "fill"

string extra;

double mean-x2, mean-y2;
double mean-x1, mean-y1;

double end-y;
double end-x;

double theta = offset * m-theta;
vector<double> ys;
vector<double> xs;

} const TestResults & next
  
const TestResults & curr,
  
int offset,
  
const string & color,
  
void Pointer::draw-section(string & buffer,

  
} 

theta += m-theta;
  
theta += ++t;
draw-mean(buffer, mean-x1, mean-y1, mean-x2, mean-y2);

SVG: path(buffer, x, y, extra);

ys.push-back(end-y);
xs.push-back(end-x);
proc(m-origin-x, m-origin-y, curr-upper-error() * 2.5, theta, end-x, end-y); // get fourth corner

ys.push-back(end-y);
xs.push-back(end-x);
proc(m-origin-x, m-origin-y, next: theta + m-theta, end-x, end-y); // get third corner

ys.push-back(end-y);
xs.push-back(end-x);
proc(m-origin-x, m-origin-y, next: upper-error() * 2.5, theta + m-theta, end-x, end-y); // get second corner

ys.push-back(end-y);
xs.push-back(end-x);
proc(m-origin-x, m-origin-y, curr-upper-error() * 2.5, theta, end-x, end-y); // get first corner

proc(m-origin-x, m-origin-y, next: mean() * 2.5, theta + m-theta, mean-x2, mean-y2);
static int placeholder = m_results.size();

} // string SITE-NAME
const string & color,
void Patter::drawLegend(string & buffer,

{ draw-SECTION(buffer, color, offset + 1, *curr, start);
  
  if (offset + 1 < test-count) 
    { offset = test-count; size();
      
      test-results[0][tag] = Tag;
      test-results[1][tag] = test-iter;
      test-results[2][tag] = test-iter;
      vector<TestResults>::const-iterator test-iter = test-results.begin();
    }
  
  const TestResults & curr = test-results[0][tag];
  const TestResults & tag = test-iter;
  const TestResults & start = *test-iter;
  vector<TestResults>::const-iterator test-iter = test-results.begin();

} // real-F2n optimized
string color = "colors.dat";

const string color = "colors.dat";

void pointer::import_colors()
{
    // Helpers: General pointer methods

    Path path = new Path();
    path.moveTo(0, 0);
    path.lineTo(100, 0);
    path.lineTo(100, 100);
    path.lineTo(0, 100);
    path.closePath();

    Fill fill = new Fill();
    fill.setColor("#F00");
    fill.setOpacity(0.5);

    Stroke stroke = new Stroke();
    stroke.setColor("#00F");
    stroke.setLineWidth(2);

    Graphics graphics = new Graphics();
    graphics.setFill(fill);
    graphics.setStroke(stroke);
    graphics.drawPath(path);
}

void pointer::draw-mean(int x, int y)
{
    // Calculate mean of x and y coordinates
    double x1 = (x1 + x) / 2;
    double y1 = (y1 + y) / 2;

    // Draw mean point
    draw point(x1, y1);
}

int main()
{
    int x1 = 50;
    int y1 = 50;
    int x2 = 150;
    int y2 = 150;

    // Draw line from (x1, y1) to (x2, y2)
    draw line(x1, y1, x2, y2);

    // Draw rectangle from (x1, y1) to (x2, y2)
    draw rectangle(x1, y1, x2, y2);

    // Draw text at (x1, y1)
    draw text("Hello, World!", x1, y1);

    return 0;
}
System // 1572

// USE-MATH-DEFINES

// USE-MATH-DEFINES

// USE-MATH-DEFINES

#define PLOTTER-H
#define PLOTTER-H

END >>>>>
{
    double & end-y)
    double angle, double radius, double start-y, double start-x, double end-x,

void PLOTTER::proc(double start-x, double start-y, double end-x, double angle, double radius)
{
    {
        colors-ffee >> color;
        m-colors.push_back(color);
    }
    white (colors-ffee)
    colors-ffee.open(color, c-str());
}
const MasterStepPerMaster();
        int test-cut,
        double theta,
        List<Results> * show-results,
        Plotter(vector<Results> * results,vector<
        :{}
        ) (0) (0, ORIGIN-X,0, ORIGIN-Y);
    public:
    class Plotter
    {
    using std::sin;
    using std::cos;
    using std::vector;
    using Statements;
    
    typedef Plotter* Plotter;
    
    class Plotter;
    
    typedefs //
    
    #include "svg.h"
    #include "interactor.h"
    #include "stepsmaster.h"
    #include "types.h"
    #other directives //
    
    #include "math.h"
    #include <vector>
    #include <string>
void draw-section(string &output,
    void draw-axes(output);

HEPERS: SVG, plotter, methods //

void htmz-footer(string &-htmz);
    void result-table-ftr(string &-htmz);
    void result-table-rows(string &-htmz);
    void result-table-header(string &-htmz);
    void htmz-header(string &-htmz);

HEPERS: Output Formatting //

vector<strin> -colors;
    list<results> *-results;
    vector<vector<testresults>> *-ad]-list;
    double m-radius;
    double m-theata;

const double m-ORIGIN-Y;
const double m-ORIGIN-X;

int m-endpoints;

private:
    void plot-results(string out=filename="output.HTMz");

    ~plotter();
}
END >>>>>>
#endif

double x_end,
double y_end,
double angle,
double radius,
double center,
void proc(double center, 
void import_colors();

HEPVERS: General Plotter Methods

void draw_mean(string & output, 
string sttene-name),
const string & color, 
void draw_legend(string & buffer, 
const vector<testresult> & tests),
const string & color, 
void draw_string(string & output, 
const testresults & next),
const string & color,
void draw_string & color,
void draw_string & color,
Inherited Methods

{  
  return temp & temp薮(size - 1);
  }  

if (site::get-n(string test-name) {

  using std::sort;
  using std::shared_ptr;
  using std::test-team;
  using std::vector;
  using std::string;
  using std::map;
  using std::map;
  using std::cout;
  using std::cout;

  #include "cycle.h"
  #include "types.h"
  #include "user-defined directives //

  #include <algorithm>
  #include <memory>
  #include <string>
  #include <assert>
  #include <iostream>
  #include <iostream>
  #include <iostream>
  #include <iostream>
  #include <iostream>
  #include <iostream>

  system directives //
```cpp
vector<dataset_ptr>::iterator iter;
} (void State: merge-test(dataset_ptr add-data) {
  *(iter += merge(temp)) = temp->name();
  for (iter = m-tests.begin(); iter != m-tests.end(); ++iter)
    vector<dataset_ptr>::iterator iter =
     thread-ptr<dataset> temp[new dataset(add-data)];
} (void State: merge-test(const dataset add-data) {
  sort-tests():
    m-tests.push-back(new-set);
    dataseleter new-set = new dataset(add-data);
} (void State: add-test(dataset add-data) {
  temp = nullptr;
  delete temp;
  sort-tests():
    m-tests.push-back(temp);
    dataseleter temp = new dataset(add-data);
} (void State: add-test(const dataset add-data) {
```
uses bubble sort

swapLead = false;
}
}
int start = 0;
bool swapped = true;
}
void Site::sort_tests()
{
    return false;
    return true;
    if (dataset->name() == dataset->name())
        for (dataset->begin() ; iter != dataset->end() ; ++iter)
            const_iterator iter;
        vector<dataset_ptr> const &dataset =
            Site::collections(dataset = const &dataset)
        } const

    return datset;
    if (dataset->name() == test_name())
        for (dataset_ptr dataset : m_tests)
            Site::get_test(String test_name()) const
        }

    if (iter == m_tests.begin() ; iter != m_tests.end() ; ++iter)
        if (dataset->name() == adj.data->name())
            if (dataset->name() == adj.data->name())
                for (iter = m_tests.begin() ; iter != m_tests.end() ; ++iter)
using statements
#include "ypes.h"
#include "dataset.h"
user defined directives

#include <memory>
#include <list>
#include <vector>
#include <fstream>
#include <stream>
system directives

#define SITECORE_H
#define SITECORE_H
#define SITE_CORE_H
#define SITE_CORE_H

END

{ }

{ }

swapped = true;
swap([l + t] + tests, w - tests, [t] + l + tests)

for (size_t t = 0; t < w - tests.size; t++)
start++;
void add-Test(Dataset &dataset, add-data);

void add-Test(const Dataset &add-data);

operations //

datasets //

vector<Dataset &dataset, w-descriptions;

To Remove //

vector<Dataset &dataset, w-tests;

protected:

int get-n(string test-name);

virtual void describe() = 0;

virtual void print-tests() = 0;

class Site //

public:

{}()

Site();

~Site();

using std::shared_ptr<T SITE-SITE>

using std::shared_ptr<Dataset> dataset;

using std::vector;

class Site;

T36 //

T35

T34

T33

T32

T31

T30

T29

T28

T27

T26

T25

T24

T23

T22

T21

T20

T19

T18

T17

T16

T15

T14

T13

T12

T11

T10

T9

T8

T7

T6

T5

T4

T3

T2

T1

T0
using statements

#include "streammaster.h"
#include "types.h"
#include "USER Defined Directives //

#include <memory>
#include <string>
#include <cassert>
#include <fstream>
#include <iostream>
#include <iomanip>
#include <fstream>
#include "system directives //

streammaster.cpp

END >>>>>
#endif

void sort_tests();

bool collision(int& test, string& test_name) const;

barasetter get_test(string test_name) const;

search //

virtual void test_remove_test(string test_name) = 0;

void merge_to_an_exisiting_dataset, by shared_ptr //

void merge_to_an_exisiting_dataset, by reference //

void merge_test(const barasetter& add-data) ;

};
desc = (order == \"A\") ? false : true;

InetSocketAddress order;

InetSocketAddress name-setter()

InetSocketAddress open(path, c-ctrl())

void desc)

vector<double> data;

FileStream InetSocketAddress;

String s-reader;

String test-unit;

String name-setter;

char order;

bool desc;

} (MasterState::MasterState(String path) : state-desc(strtolower(true))

{id-tests()

sort-tests()

} copy-iterator?
for (string path : paths) {
    MasterSite::MasterSite(vector<string> paths) : state-decriptor(true) {
        // Sort all the datasets names for this
        sort(all the datasets names with same name)
        // don't add datasets with same name
        data.clear();
    // Clear END
    got all data
    // printing data read
    unit
    //
    getSiteInput(reader, s-reader);
    data.push-back(reader)
    while (s-reader.size() > 50)
    getSiteInput(reader, s-reader);
    //}
}
vector<dataset::iterator>;

} void MasterSite::print_tests()
  // observers

{ w-descriptions::clear();
  m-tests::clear();
  delete result;
  for (verboseMasterSite Result : w-descriptions)
    { test = multipler;
      delete test;
    } for (dataset test : m-tests)
  } MasterSite::MasterSite()

{ id-tests();
  sort-tests();

  { temp::reset();
    this->MasterSite::add-dataset(temp);
    shared_ptr<shared_dataset> temp(new shared_dataset(path));
  }
{ report = new TestDecorator("MASTER",
  "push = new TestDecorator("MASTER",
  for (Dataloader test : tests)
  VerboseErsatzTest.push();
  )()
  void MasterSite::describe()
  {

    for (Test test = test->maxmean();
        test->median();
        test->mean();
        test->meanerror();
        test->up_meanerror();
        test->size();
        test->tp();
        test->unit();
        test->name();
        test->begin();
        for (Test* test = test->maxmean();
            test->median();
            test->mean();
            test->meanerror();
            test->up_meanerror();
            test->size();
            test->tp();
            test->unit();
            test->name();
            test->begin();
            " Master "
            echo " END "
        )
    )
    count += 4;
break;
    ():() = bases
    ():() = val
} (i = test-td) (j = test+td)
for (i = m-tests.begin(); j = m-tests.end(); j++)
vector<dataset> ::iterator ier;

void master::std-rec(fact test-f, double & val, double & bases)
{

    for (vector<dataset>::iterator ier = test-f.begin();
    ier = test-f.end(); ier++)
} int master::test-td(fact test-f, int test-f) const
{
    result = multplet;
    return;
    -result = result;
    if (result-td(test-f) == test-f)
    for (verbose::result, verbose::verbose::result as verbose::result)
} void master::descibe(fact test-f, verbose::verbose::result as verbose::result)
void MasterSite::remove-test(string name) { }

// Mutators
{
  
return *iter;

  if (iter->name() == name)
    for (iter = m-tests.begin(); iter != m-tests.end(); iter++)
      vector<MasterSite>::const_iterator iter;

} } MasterSite::get-test(string name) const

} } { (*iter*) = basis
    (*iter*) = val

} if (iter->name() == ds-name)
  for (iter = m-tests.begin(); iter != m-tests.end(); iter++)
    vector<MasterSite>::const_iterator iter;

void MasterSite::std-rec() { std::string ds-name, double & val, double & basis

} {

}
void MasterSite::add-data(SampleSite Amrage)
{
    void MasterSite::add-data(tests::Merge::tests BEGIN()::
        sort-tests():
        
        // check if master site has no data, create the first set
        void MasterSite::add-data(SampleSite Amrage)
        
        // check if master site has no data, create the first set
        } (this = tests::Merge::tests BEGIN();
    }
}

}
:td-tests();
:sort-tests();
{

}\ter++;
\collision(\ter) \if \merge-test(\ter) : add-test(\ter);
}\collision(\ter) \if \merge-test(\ter) : add-test(\ter);
}
}
while (\ter = merge-tests.end())

\ter++;
\this-w-tests.push-back(\ter);
if (\this-w-tests.size() == 0)

while (\ter = merge-tests.end())

\ter++;
\this-w-tests.push-back(\ter);
if (\this-w-tests.size() == 0)
```
define MASTERSTATE
#define MASTERSTATE_H


```
public:
    } // public site

typedef shared_ptr<MasterSite> s-MasterSite;
typedef MasterSite* MasterSitePtr;
typedef shared_ptr<IMasterSite> s-IMasterSite;
typedef IMasterSite* IMasterSitePtr;
    // typedefs

class MasterSite;
class SampleSite;
    // Forward declarations

    using std::shared_ptr;
    using std::vector;
    using std::placeholders;
    using std::placeholders::_1;
    #include "types.h"
    #include "stlsemaphore.h"
    #include "site.h"
    #include "dataset.h"
    #include "stream.h"
    #include <memory>
    #include <vector>
    #include <iostream>
    #include <system DIRECTIVES //

    // User defined DIRECTIVES //
```cpp
// Add or merge file path.
void add_data(string path);
// Add or merge site's site into this site
void add_data(site& merged);
// Add or merge into this site
void add_data(sitesitesite merged);
// Remove test(string site name);
void remove_test(string site name);

// Mutators

// Construct from a vector of paths;
MasterSite(vector<string> paths);
// Construct from a single site data file
MasterSite(string path);
// Copy from a sampled site
MasterSite(const SampledSite& copy-in);
MasterSite(const MasterSite& copy-in);
```
using std::vector;
using std::endl;
using std::cout;
using statements //

#include <iostream>
#include <string>
#include <assert>
#include <fstream>
#include <iostream>
#include <sstream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
#include <iostream>
using std::vector;
using std::shared_ptr;
using std::string;

using std::istream;

set the item.

set the item.

set the item.

set the item.

set the item.

set the item.

set the item.

set the item.

set the item.

set the item.

set the item.

set the item.
```java
sort-tests():
InputTest.close();
Clear stream //
{

data.clear();
Prepare for next iteration //

w-tests.push-back(dataset);
if (collition(dataset))

desc();
test-unit, test-name,

dataset = new dataset(data,
Create a dataset //

getLine(InputTest, s-reader);

} // Clear "END"
data acquired //

Prime the stream //

getLine(InputTest, s-reader);
getline(InputTest, test-unit());
getline(InputTest, test-unit());

desc = (order == ",") ? false : true;
```
void SampleEnv::test::present(test::td) const
{
    for (test::dataset& test : m-tests)
    {
        return
    }
}

void SampleEnv::descriptive()
{
    m-descriptive.push_back("push()
    
-push = new TestDescriptive(name)");
}
void SampleData::removeTests(string name) {
    
    } // SampleData::removeTests

    
    
    void SampleData::removeTests(string name) {
        
        } // SampleData::removeTests

    
    
    void SampleData::removeTests(string name) {
        
        } // SampleData::removeTests
vector<iterator>::iterator iter = w-tests.begin();
}
}

bool SampleDataset::standardize(Iterator test-1d, double ref-wa1, double ref-bas1s) {

{

return false;
{

}

return true;

w-stdcoun++;

}(*iter)-(standardize(ref-wm1, ref-bas1s));
}
}

if ((((*iter)-name()) == dataset-name && (*iter)-standardized())
{
    w-tests.end();
}

while (iter = w-tests.begin();

}

{ w-stdcoun--;

}(*iter)-normalizer() = (*iter)-name.setter();
}

for (iter = w-tests.begin(); iter = w-tests.end(); iter++)

vector<iterator>::iterator iter;
}
)

void SampleDataset::normalizer(string name.setter)
```cpp
    } // string test-unit:
    string name-setter:
    }() {
        string site-id = this->site-id();
        string site-name = this->name();
    }

    void summarizes():
    { // w-summarizes.push-back(dataset.get()):
        for (dataset<tree : w-tests>:
            w-summarizes.reserve(w-tests.size());
            vector<dataset>::iterator dataset-iter = w-tests.begin();
    }

    void generate-payloads():
    { // w-site=datasetidency()
        return false:
        { // w-site++
            while (true:
                w-site++
                w-site-td=standardize(r-val, r-base):
                } (wild*dataset::*td & test-td & (dataset::*standardize)())
            } (wild*dataset::*td & test-td & (dataset::*standardize)())
        } (*dataset::*td & test-td & (dataset::*standardize)())
    }
```
delete payload;

m-payloads.push_back(*payload);

});//
m,
we,
test-td,
test-unit,
name-setter,
test-td,
payload = new TestResults*name,

// store the payload
});//(())punten<()>()hlen<()>()mem-<()>()nter<()>()lower<()>()norm-<()>()ower<()>()punten<()>()unit<()>()test-<()>()td<()>()test-unit<()>()uD<()>()test-flat<()>()test-setter<()>()test-td<()>()test-unit<()>()test-<()>()next<()>()name<()>()name-setter<()>()name<()>()next<()>()create the payload data

} ((())-standardized) td<()>()test<()>()for (teer = w-tests.begin(); teer != w-tests.end(); ++teer)

vector<dataset>::const_iterator teer;

vector<dataset> payload;

double we, m, te;
```cpp
#include "dataset.h"
#include <vector>
#include <iostream>
#include <memory>
#include <iostream>
#include <iostream>
#include <memory> //

#include "SITESMP.h"
#include "SITESMP.h"

private sampleSet.h

END >>>>

return nullptr;

return

if (aIter) == name

for (auto iter = w_tests.begin(); iter != w_tests.end(); iter++)

vector<dataset> const_iterator iter;

dataSet Iterator: get-Test(string name) const

private

{ }

{ }

payload = nullptr;
```
class SampleDescSite {
    // Constructors & Destructor

    SampleDescSite();
    SampleDescSite(string path);
    ~SampleDescSite();
    
    // Master needs access to m-sets for merge
    iterator needs to use id-sets();

    private:
        typedef std::shared_ptr<SampleDescSite> sSampleDescSite;
        typedef SampleDescSite::iterator sMasterSite;
        typedef SampleDescSite::iterator sMasterSitePtr;
        typedef SampleDescSite::iterator sMasterSiteSet;

    public:
        class SampleDescSite {
            
            // Forward declaration

            using std::shared_ptr;
            using std::vector;
            using std::string;
            using std::set;
            
            #include "AbstractMaster.h"
            #include "Abstract.h"
            
            };
        
    
"
void generate-payloads();
{
    std::string name="test-td!
double m-win, double r-basis;

    normalize(tjs); normalize(test-id, m-win, double r-basis);
    // Returns true if dataset was normalized.
    bool standardize = true(test-id, m-win, double r-basis);

    normalize(string name-setter); normalize(name, double r-basis);
    // Normalize a dataset
    void normalize_all_datasets();

    // void remove-test("string name-setter");

    void test-enumerator(int test-id, string name-setter)
    {
        void test-enumerator(int test-id) { w-id = test-id; }
        void test-enumerator(int test-id) { site-name = site-name; }
        void set-name("string site-name")

        // Mutators

        vector<testresults> const * get-payloads();

        bool present(test-id)
        { return test-present(test-id); const
            return tests-stdcount();
            int test-count(); const return m-tests-stdcount();
            return site-name; }
        return site-id; const return m-id; const
        to Remove

        void destroy();
        void print-tests();
        observes //
        }


```cpp
/* WArtappers */
{
  font-size: 10px;
  font-family: 'Open Sans', sans-serif;
} /* text */

{  
  font-family: 'Open Sans', sans-serif;  
} /* body */

body styles >>>>

END >>>>
}

\#endtt

// Returns a shared pointer to a dataset, nullptr if not found
Dataset* getDataset(const std::string& name) const
{
  vector<DatasetSummary>* m_summaries;
  vector<TestResults>* m_payloads;
  string site-name;  
  int m_stcount;
  int m_id;
  bool state-descriptor;
  private:
}
End >>>>

```html
{  
  background-color: #d9e7f7;  
  text-align: center;  
}
```

```html
{  
  padding-right: 5px;  
  padding: 2px;  
  text-align: right;  
  font-size: 12px;  
  font-family: 'Droid Sans Mono';  
  border-top: 1px solid #555;  
  num-result
}
```

```html
{  
  padding-right: 5px;  
  padding: 2px;  
  font-weight: bold;  
  font-size: 12px;  
  name
}
```

```html
{  
  width: 300px;  
  col
}
```

```html
{  
  color: #fff;
}
```
void SVG::header(string &svg, int height, int width)
{
    svg += "<svg height=" + to_string(height) + " width=" + to_string(width) + " xmlns="http://www.w3.org/2000/svg"
";
    svg += "version=" + to_string(version) + " baseProfile="full" viewBox="0 0 " + to_string(width) + " " + to_string(height) + ""><rect width=" + to_string(width) + " height=" + to_string(height) + "x="0" y="0" rx="0" ry="0" fill="blue" stroke="black" stroke-width="2" /></svg>
";
    svg += "</svg>
}

void SVG::footer(string &svg)
{
    svg += "</" + to_string() + "svg";
}

using std::to_string;
using std::vector;
using using Statements

#include "svg.h"
// other directives

#include <vector>
#include <string>
#include <sys/protectives //
dpp >>>>>

2414
2413
2412
2411
2410
2409
2408
2407
2406
2405
2404
2403
2402
2401
2400
2399
2398
2397
2396


double y,

double x,

void SVG::rectangle(string &buffer,

{

   buffer += "\n";
   buffer += extra;
   buffer += " " " ";
   buffer += to_string(r);
   buffer += " "ly=
   buffer += to_string(y);
   buffer += to_string(x);
   buffer += "";
   buffer += to_string extrax;
   buffer += "";
   buffer += to_string extray;
} ( )

double r,

    double y,

double x,

    double x,

void SVG::point(string &buffer,

{

   buffer += "\n";
   buffer += extra;
   buffer += " Z";
   buffer += to_string extrax;
}
END >>>>
{
  buffer += text;
  buffer += table;
  buffer += extra;
  buffer += extra;
  buffer += extra;
  buffer += extra;
  buffer += extra;
  buffer += extra;
  string extra = "\"" + to_string(y) + "\"" + to_string(x) + "\"" + to_string(w) + "\"" + to_string(h) + ";\n  double y, x;
  double x, y;
  string table;
  void SVG::text(string &buffer, ...}
{
  buffer += text;
  buffer += extra;
  buffer += extra;
  buffer += extra;
  buffer += extra;
  buffer += extra;
  buffer += extra;
  string extra = "\"" + to_string(y) + "\"" + to_string(x) + "\"" + to_string(w) + "\"" + to_string(h) + ";\n  double y, x;
  double w, ...
```cpp
void path(string & this_path) {
  string extra = "";
  const vector<
double> & y,
  const vector<
double> & x,
  string path(string & this_path) {
    string extra = "";
    double end-y,
    double end-x,
    double str-y,
    double str-x,
    double end-
    int width,
    int height);
    int width,
    int height);
    void footer(string & svg) {
    void header(string & svg) {

} namespace SVG {

    using std::vector;
    using std::string;

    #include <vector>
    #include <string>

    // define SVG-H
    #define SVG-H
    #ifdef SVG-H
    
```
```
```cpp
#include "types.h"

// User Defined Directives

// System Directives

double pos-x, double pos-y;
```

w-unit \ = \ test-unit;
    w-test-name \ = \ name-setter;

} \) results(state-name, site-id, u, t, I)
    : double t,
    : double w,
    : double u,
    : test
int test-id,
    string test-unit,' test
    // state-name-setter,' test
int site-id,' test
    string results(state-name,' test

{ \)
    w-lw-err = I;
    w-mean = w;
    w-dp-err = u;

    w-state-id = site-id,' test
    w-state-name = state-name,' test

} \) results(state-name,' test


```cpp
// Forward declarations

using std::string;
#include <iostream>

#define TYPES_H
#define test_results

test_results

END

{ }

: results(site-name, test-td, test-name, test-unit, test-d, we, mean, te)
```

```csharp
private double[,] m;
private double u;
int site_id;
Results(string site_name,
{
    public:
    } class Results
    {
    {
    double value;
    double norm_value;
    int rank;
    } struct DataNode
    {
    enum NORM_T { SELF, EXTERNAL };
    typedef DatasetSummary* SummaryPtr;
    typedef Results* ResultSet;
    typedef TestResults* TestAdapter;
    typedef TestDescriptor* VerboseResultSet;
    typedef DataNode* DataAdapter;
    }
```
public:
    // Once constructed, unmodifiable.
    class TestResults : public Results {
        {
            double m-tw-err;
            double m-mean;
            double m-upp-err;
            {
                double lower-error() const { return m-tw-err;
                    {
                        double mean() const return m-mean;
                        {
                            double upper-error() const return m-upp-err;
                        }
                    }
                }
            }
        }
    }
public class TestDescriptor : public TestResults
{
    public

    double max; double q3; double med; double q1; double mn;

    TestDescriptor(const TestResults & results,
    
    public:

    

    };

    int m-site-id; int m-test-id; string m-unit;

    string m-test-name; string unit;

    

};

// Inherited members serve as single data points (distances).

    { } const test-id() const return m-test-id;
    { } const test-unit() const return m-unit;
    { } const test-name() const return m-test-name;

    observers

    double I;
    double m,

    };
<?php

if (isset($target_dir) && !file_exists($target_dir) && !mkdir($target_dir, 0777, true))
    if (move_uploaded_file($filename, $target_dir))
        $target_file = $target_dir . $filename . '.dat';

$target_dir = 'uploads/';
$iterator = new FilesystemIterator('uploads/', FilesystemIterator::SKIP_DOTS);

require 'include/header.php';

if (isset($display_errors) || !defined('E_ALL') && !(defined('E_STRICT') && defined('E_DEPRECATED')))
    error_reporting(E_ALL);

?>

<?php

<!DOCTYPE html>

<html>

<head>
    <title>Error: No file selected</title>
</head>

<body>

<h1>Error: No file selected</h1>

</body>

</html>
```
<?php

header("location: " . $SERVER[PHP_SELF]);

exec("rm -rf uploads/");

if (!isset($_POST, 'reset')) {
    exec("touch .out.");
}

isset($_POST, 'compute')
```
Bibliography & References


“9 Basic Shapes.” Basic Shapes – SVG 1.1 (Second Edition),

www.w3.org/TR/SVG/shapes.html.

“Area of a triangle - "side angle side" (SAS) method.” Area of a triangle

(Side-Angle-Side method) - Math Open Reference,


“Best Practices for File Formats.” Stanford Libraries,


Choi, Jung Kyoon, and Sang Cheol Kim. Genetics, Copyright © 2007 by the Genetics Society of America, Apr. 2007,

www.ncbi.nlm.nih.gov/pmc/articles/PMC1855137/.


“Flask home.” Welcome | Flask (A Python Microframework), flask.pocoo.org/.

“Is it a good practice to always scale/Normalize data for machine learning?” Cross

Validated,


Mark Otto, Jacob Thornton, and Bootstrap contributors. “Bootstrap.” · The most popular HTML, CSS, and JS library in the world., getbootstrap.com/.

“NumPy.” NumPy - NumPy, numpy.org/.


