**CR2C Data System Description:**

CR2C’s data systems manage two main streams of data: 1. Low-volume laboratory and field monitoring data that are stored on a google spreadsheet as entered by the facility’s operators; 2. High-volume Human Machine Interface (HMI) data from the facility’s sensors stored automatically on a remote server. These data are cleaned and analyzed by the cr2c-monitoring scripts, stored in the “MonitoringProcedures/Data” directory on the Codiga Center’s box sync folder, and used to perform validation and modeling exercises. Each data stream and process is described below (a schematic of the system is shown in **Figure 1**). For more detailed information on each script, please see the Documentation file in the repository.

**Figure 1:** CR2C Data Systems Schematic



Lab Data:

Raw laboratory data are entered manually into the “CR2CMonitoringData” google spreadsheet. The “process\_lab\_data.py” script reads these data from the google spreadsheet using the “google sheets” API, verifies that dates, stages and values have been entered correctly and computes the resulting values (for example, raw COD data gives a dilution factor and a reading from the Hach spectrometer and the code computes the resulting COD value (in mg/L)). Currently, the “process\_lab\_data.py” script does not use the “get\_gsheet\_data.py” script as a dependency for reading in data from the gsheet file, but this change is planned for the near future.

Cleaned lab data are inserted/updated into their respective table on the “cr2c\_lab\_data.db” sql “db” file on Codiga’s Box folder. The lab data structure is shown in **Figure 2**. The “db” file contains a table for each parameter type (eg. “COD\_data” table in the “db” file is a table containing processed COD lab data). Each table has a column indicating the date and time of sample collection (“Date\_Time”), the treatment stage of the collected sample (“Stage”), the value type (“Type”, where appropriate, eg. for TSS/VSS, COD, VFA), and the value of the laboratory parameter itself (“Value”).

**Figure 2:** Lab Data Structure Schematic



Field Data:

Data from monitoring forms that are filled out on site are synced to the “CR2CMonitoringProcedures” spreadsheet through the google forms app. In the same way as the laboratory data, these data can be downloaded from the google spreadsheet using the “google sheets” API. Storing these data on a separate sql database file and writing an sqlite3 wrapper to allow flexible querying of the file to retrieve individual answers is a planned addition to the system.

The structure of the field data as they are currently stored in the “CR2CMonitoringData” is shown in **Figure 3**. Somewhat analogously to a spreadsheet, each tab is like a table in an sql “db” file, which stores the answers to a given google form. Each variable name in the table for a given google form is the name of the question in that form (such as the “CR2C Daily Operations Log” form) and each row is an answer to that question. The time stamp of each form submission is automatically stored by google forms. The “get\_field\_data.py” script that will be added to the repository will retrieve answers to any subset of questions on the google form.

**Figure 3:** Field Data Structure Schematic



HMI Data:

Data from the facility’s Human Machine Interface (HMI) are logged as meter values change appreciably over a ~30s period (if no change is observed, a new value is not logged). These data are stored in a server close to the facility and can be queried using the eDNA program on the server and transferred onto a local machine through the Virtual Private Network (VPN). Before they can be used to run further analyses, the data have to be cleaned to remove erroneous values. The “hmi\_data\_agg.py” is currently the data processing script that performs these cleaning tasks, which consist of removing values above or below a certain limit (the limits are set differently for different types of parameters). The main purpose of the “hmi\_data\_agg.py” script, however, is to calculate total and/or average parameter values for a given time period and range using linear interpolation.

Cleaned and averaged/totaled data for any element id that are output by the “hmi\_data\_agg.py” script are automatically uploaded to an SQL database on the Codiga Center’s Box Sync folder. The structure of these data is displayed in **Figure 4**. Because we anticipate processing large volumes of HMI data for many of the facility’s different meters, the HMI data are partitioned into separate sql “db” files by year (corresponding to the dates of the HMI data queried, NOT the day that the script is run). Each “cr2c\_hmi\_agg\_data\_[year].db” file contains a table whose name corresponds the type of aggregated data requested in the input arguments to the “hmi\_data\_agg.py” script. For example, if I tell the script to give me 1 hour flow averages for the FT202 flowmeter for a raw HMI data file with data for July-August 2017, the results will be inserted or updated to a table called “FT202\_1hour\_AVERAGES” on the “cr2c\_hmi\_agg\_data\_2017.db” file.

**Figure 4:** HMI Aggregated Data Structure Schematic



To run any of these scripts on your machine, you have to have Box Sync installed and Codiga’s “CR2COperations” folder synced on it. Codiga’s Box folder contains the client secret that gives the script permission to read data from the “CR2CMonitoringData” google spreadsheet as well as the “cr2c\_lab\_data.db” file, so without Box sync installed, the program won’t work!

Visualization and Validation

Cleaned and processed laboratory, field, and HMI data are used to produce visualizations and summary statistics. This is accomplished with the “get\_lab\_plots.py”, “get\_wide\_tables.py”, and “get\_reactor\_plots.py” scripts which first get their data by querying each sql file (using the “get\_lab\_data.py”, “get\_field\_data.py”, and “get\_hmi\_agg\_data.py” sqlite3 wrappers). Some of these data sources are combined to run scripts that need more than one data source to validate meter (or laboratory) readings. For example, the “get\_mass\_balance.py” script computes a COD mass balance using laboratory COD data and average flowrates through the reactors from the HMI.