Click to edit Master title style

QARTOD in Practice

Presented by Luke Campbell

Lessons Learned

Lessons Learned

- QARTOD is running in real-time for the Chesapeake Bay Interpretive Buoy System
  - Some degree of test coverage for all scientific parameters
  - Strong coverage for:
    - Currents
    - Temperature & Salinity
    - Dissolved Oxygen

Lessons Learned

- Configuring Test Constraints is not trivial:
  - Consistent Units between configuration and data
  - time-varying parameters
  - data model schema
    - How are deployments treated?
    - Maintenance vs recovered vs telemetered

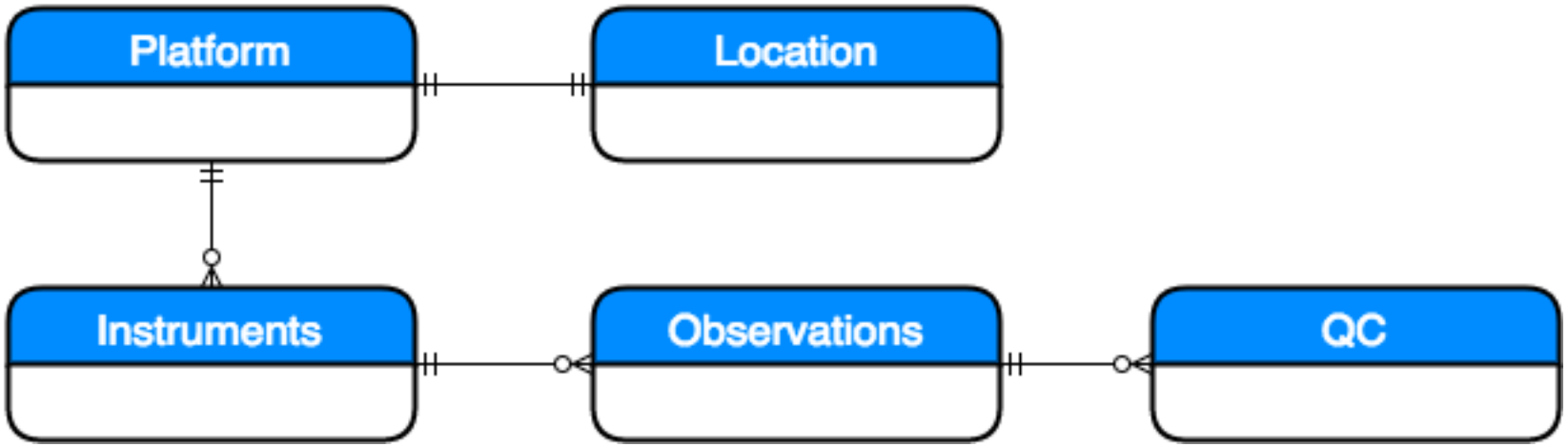
- How to deal with missing values
  - Types of missing data

The CBIBS Solution

IOOS | EYES ON THE OCEAN

The CBIBS Solution

- All of our observation data is stored in a PostGIS database.



The CBIBS Solution

- We run all QC tests on all recent observations every hour, or more frequent (schedule is dependent on which platform)

- We use google docs to configure QC parameters for all stations and parameters:
  - Station ID
  - Parameter Name
  - Units
  - configuration variable (min, max, rate of change, etc.)

The CBIBS Solution

- Test runs overlap

- We do infrequent manual historical QC runs if we get delayed data or make corrections to a process.
  - For example, we identified a bug in our processing of salinity, after regenerating all of the historical salinity values, we re-ran QC for all historical salinity observations

- Missing values are only identified in the cases for instrument failures
- All QC default to 2 for "Not Evaluated" on initialization

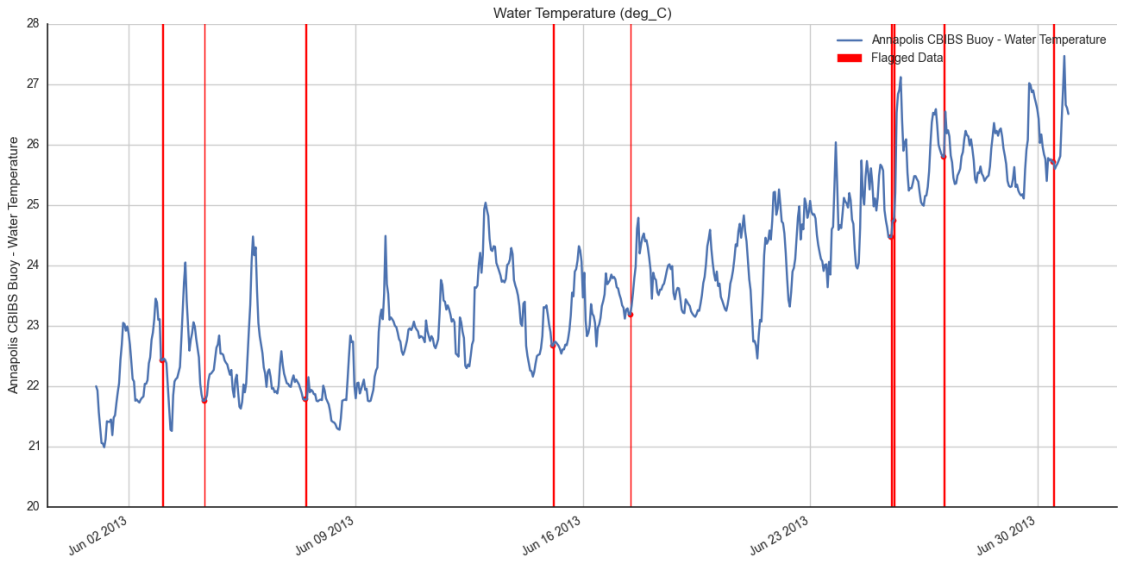
The CBIBS Solution

More on the DMAC side:

- We expose interfaces into the database for data access
  - THREDDS
  - Public API
  - API shows only data that are not marked as 3 or 4 (suspect or bad)

The CBIBS Solution

It works!



Time	Observation	qa_code	Flat Line Test
2013-06-01 23:50:00+00:00	198	1	1
2013-06-02 00:00:00+00:00	198	1	1
2013-06-02 00:10:00+00:00	198	1	1
2013-06-02 00:20:00+00:00	198	3	3

QARTOD and netCDF

QARTOD and netCDF

CF provides guidance for storing flags in netCDF files in §3.5.

The attributes `flag_values`, `flag_masks` and `flag_meanings` are intended to make variables that contain flag values self describing. Status codes and Boolean (binary) condition flags may be expressed with different combinations of `flag_values` and `flag_masks` attribute definitions.

QARTOD and netCDF

- Flags as encoded masks
- Flags as values

- Why we went with values

- Drawbacks
  - Adds several variables to every dataset
- Pros
    - Clear
    - Self-describing
    - Doesn't require additional programming to use
    - Bit twiddling sucks & 0x00F8300

QARTOD and netCDF

```
float sea_water_salinity(time=446);
  :_FillValue = -9999.0f; // float
  :units = "1";
  :standard_name = "sea_water_salinity";
  :long_name = "Sea Water Salinity";
  :comment = "The unit of salinity is PSU, which is dimensionless. The units attribute should be given as 1e-3 or 0.001 i.e. parts per thousand if salinity is in PSU.'
  :coordinates = "time latitude longitude depth";
  :source = "Observational data from a buoy";
  :platform = "platform";
  :cell_methods = "time: point depth: point";
  :valid_min = 0.08f; // float
  :valid_max = 29.8f; // float
  :ancillary_variables = "sea_water_salinity_flatline_qc sea_water_salinity_gap_qc sea_water_salinity_range_qc sea_water_salinity_gradient_qc sea_water_salinity_spike_

byte sea_water_salinity_qc(time=446);
  :_Unsigned = "true";
  :long_name = "Sea Water Salinity Primary QC";
  :comment = "Primary QC flag";
  :coordinates = "time latitude longitude depth";
  :source = "Observational data from a buoy";
  :platform = "platform";
  :_FillValue = 9B; // byte
  :standard_name = "sea_water_salinity status_flag";

byte sea_water_salinity_flatline_qc(time=446);
  :_Unsigned = "true";
  :_FillValue = 9B; // byte
  :long_name = "Sea Water Salinity Flat Line Test";
  :standard_name = "sea_water_salinity status_flag";
  :flag_values = 1B, 2B, 3B, 4B, 9B; // byte
  :flag_meanings = "GOOD NOT_EVALUATED SUSPECT BAD MISSING";
  :comment = "When some sensors and/or data collection platforms fail, the result can be a continuously repeated observation of the same value. This test compares the
  :references = "http://www.ioos.noaa.gov/qartod/welcome.html";

byte sea_water_salinity_gap_qc(time=446);
  :_Unsigned = "true";
  :_FillValue = 9B; // byte
  :long_name = "Sea Water Salinity Gap Test";
  :standard_name = "sea_water_salinity status_flag";
  :flag_values = 1B, 2B, 3B, 4B, 9B; // byte
  :flag_meanings = "GOOD NOT_EVALUATED SUSPECT BAD MISSING";
  :comment = "Test determines that the most recent data point has been measured and received within the expected time window (TIM_INC) and has the correct time stamp (
  :references = "http://www.ioos.noaa.gov/qartod/welcome.html";

byte sea_water_salinity_range_qc(time=446);
  :_Unsigned = "true";
  :_FillValue = 9B; // byte
  :long_name = "Sea Water Salinity Gross Range Test";
  :standard_name = "sea_water_salinity status_flag";
  :flag_values = 1B, 2B, 3B, 4B, 9B; // byte
  :flag_meanings = "GOOD NOT_EVALUATED SUSPECT BAD MISSING";
  :comment = "All sensors have a limited output range, and this can form the most rudimentary gross range check. No values less than a minimum value or greater tha n t
  :references = "http://www.ioos.noaa.gov/qartod/welcome.html";

byte sea_water_salinity_gradient_qc(time=446);
  :_Unsigned = "true";
  :_FillValue = 9B; // byte
  :long_name = "Sea Water Salinity Rate of Change Test";
  :standard_name = "sea_water_salinity status_flag";
  :flag_values = 1B, 2B, 3B, 4B, 9B; // byte
```

QARTOD and netCDF

- Further Considerations:
  - To include test values in variable attributes
  - QC Test Runtime

GliderDAC

GliderDAC

Coming soon to GliderDAC is automated QC

Manual for QC of Glider Data

Challenges:

- Gradients over pressure AND time
- Accurately separating profiles (yo)

Challenges specific to GliderDAC

- Preserving data provider QC
- Where to store QC alongside Provider Data
  - We have a policy not to modify any uploaded datasets directly.
  - We need to combine our QC results in the final dataset published

Community Library

Community Library

The core logic of our QARTOD implementation is available online at:

https://github.com/ioos/qartod/

It would be good to see this used as the reference implementation to increase consistent usage across projects and improve overall QC coverage across users.

We're working on adding a command line tool for this library to apply QC to local files.