CoTeDe Documentation

Release 0.17.0

Guilherme Castelão

Jul 15, 2018

Contents

CHAPTER 1

User Documentation

CoTeDe at a glance

1.1 Overview

This package is intended to quality control temperature and salinity profiles by applying a sequence of tests. For CTD profiles and TSG timeseries it uses the PySeabird package, so it can interpret directly the SeaBird's .cnv output file.

This is the result from several generations of quality control systems, which started in 2006, while I was in charge of the quality control of termosalinographs at AOML-NOAA, USA. Later I was advising the quality control of the brazilian hydrography of PIRATA.

CoTeDe can apply different quality control procedures:

- The default GTSPP or EGOOS procedure;
- A custom set of tests, including user defined thresholds;
- A novel approach based on Anomaly Detection, described by Castelao 2015;

1.1.1 My opinion on quality control

Quality control is different than data processing. On the processed data, the quality control/quality assurance means to classify what looks fine. It is very important that the data is properly sampled and processed. If all you have is bad data, quality control procedures can't go back on time and fix improper sampling, but only tell you that the data might not be good.

Once I was requested to quality control over 10 years of hydrographic data. It didn't take long to learn that all those cruises lack the proper procedure, which did compromise the data. That was not a quality control job, but data mining, in the sense of trying to rescue the most I could. A CTD cast is far easier than reach the moon, but there is a proper way to do it.

1.1.2 Why the name CoTeDe?

Since NOAA I wanted to combine the multiple tests, but I didn't really knew how to do that. In 2011 I learned the anomaly detection technique, but I only formalize the procedure in 2013, when I spent few months in Toulouse. The full name of this package is CoTe De l'eau, which I understand as something near to "rating the water". The short name is cotede, to make easier for the users to remember, since it is the quality control of COnductivity TEmperature and DEpth (cotede). The french name is a kind of tribute to the great time that I spent in France with Bia and the croissants that were converted in code lines.

1.2 Installation

1.2.1 Requirements

- Python 2.6 (>=2.6.5), 2.7, 3.1 or 3.2
- Numpy (>=1.1)
- PySeabird

Optional requirement

- NetCDF4, is strongly recommended, since it allows to export data into netCDF as well as to read data required for some tests. With NetCDF4 the tests "at sea" and "climatology comparison" can run acessing local files, the fastest way to do it.
- PyDap, if you want to run the climatology test accessing the WOA data from an OpenDAP server, and no not installed NetCDF4, you will need to install PyDAP.
- Matplotlib, is a powerfull library for data visualization. It is required for the graphic tools, like the visual inspection and classification of the data.

Note: Without netCDF4 nor PyDAP it is not possible to run "at sea" neither "climatology comparison" tests.

1.2.2 Installing CoTeDe

Using pip

If you don't already have PIP running on your machine, first you need to install pip, then you can run::

```
$ pip install cotede
```

Alternative

Might be a good idea to install without update the dependencies, like:

```
$ pip install --no-deps cotede
```

Note: The `--no-deps` flag is optional, but highly recommended if you already have Numpy installed, otherwise pip will sometimes try to "help" you by upgrading your Numpy installation, which may not always be desired.

1.2.3 Custom setup

The directory .cotederc is the default home directory for CoTeDe support files, including the user custom QC setup. To use another directory, one can set and environment variable COTEDE_DIR. For example, if you use bash you could include the following lines in your .barsh_profile:

export COTEDE_DIR='~/my/different/path'

1.2.4 Optional

Climatology and bathymetry

The climatology comparison test and the at sea test can run acessing a local file, which is probably the fastest way to do it. To download the required files you can inside python run this:

```
>>> from cotede.utils import supportdata
>>> supportdata.download_supportdata()
```

That will create, if doesn't already exist, a directory in your home: ~/.cotederc/data, and place the required WOA09 and etopo5 files there. That was it, you're ready to run cotede in place with any of the preset configurations. Remember to run this before leave the dock, while you still have cheap and fast access to the network.

1.2.5 Testing

I maintain a suite of tests to check CoTeDe while I keep changing and improving the code. You can use it to test if everything runs as expected on your machine. For that, in the directory where is the source code you can simply run in the shell (i.e. outside Python):

```
$ python setup.py test
```

1.3 Getting Started with CoTeDe

1.3.1 Inside python

First load the module:

```
>>> import cotede
```

Now you're able to load the CTD data:

```
>>> pqc = cotede.qc.fProfileQC('example.cnv')
```

The keys() will give you the data loaded from the CTD, similar to the output from the seabird.fCNV:

```
>>> pqc.keys()
```

To see one of the read variables listed on the previous step:

```
>>> pqc['temperature']
```

The flags are stored at pqc.flags and is a dictionary, being one item per variable evaluated. For example, to see the flags for the secondary salinity instrument, just do:

```
>>> pqc.flags['salinity2']
```

or for a specific test:

```
>>> pqc.flags['salinity2']['gradient']
```

To evaluate a full set of profiles at once, use the class ProfileQCCollection, like::

```
>>> dataset = ProfileQCCollection('/path/to/data/', inputpattern=".*\.cnv")
>>> dataset.flags['temperature'].keys()
```

The class cotede.qc.ProfileQCed is equivalent to the cotede.qc.ProfileQC, but it already mask the non approved data (flag != 1). Another it can also be used like::

```
>>> from seabird import cnv
>>> data = cnv.fCNV('example.cnv')

>>> import cotede.qc
>>> ped = cotede.qc.ProfileQCed(data)
```

1.3.2 More examples

I keep a notebooks collection of practical examples to Quality Control CTD data . If you have any suggestion, please let me know.

1.4 Command line (ctdqc)

A CTD data file can be quality controled from the shell script using the command line ctdqc. On this way it's easy to run the quality control from the shell, for example in a cron script for operational procedures.

In the shell one can run:

```
$ ctdqc MyData.cnv
```

A new file is created, MyData_qced.nc with depth, temperature and salinity, with the respective quality control flags. It's used the default cotede setup of tests.

With the command line it's easy to run in a full collection of cnv files, like:

```
for file in `find ./my_data_directory -iname '*.cnv'`;
do ctdqc $file;
done
```

This shell script will search for all .cnv files inside the directory ./my_data_directory (and sub-directories), evaluate each file and create on the side of the original data a netCDF with the QC flags.

In the future I'll turn this ctdqc command much more flexible.

1.5 Tests for Quality Control

These are the tests available, and can be explicity accessed at cotede.qctests. Most of them simply reproduce the procedure recommended by GTSPP, EuroGOOS, IMOS, ARGO and others.

Although I slightly modified the names of some Q.C. test, the concept behind is still the same. The goal was to normalize all tests to return True if the data is good and False if the data is bad. For example, ARGO's manual define "Impossible Date Test", while here I call it "*Valid Date*".

The result of each test for each measurement is coded according to recommendation of IOC given in the table below. For example, if the climatology database is not available, the output flag would be 0, while a fail on the same climatology test would return a flag 3 for GTSPP procedure.

Flag table

Flag	Meaning
0	No QC was performed
1	Good data
2	Probably good data
3	Probably bad data
4	Bad data
9	Missing data

1.5.1 Tests

Valid Date

Check if there is a valid date and time associated with the measurement.

Valid Position

Check if there is a valid position associated with the measuremnt. It should have a latitude between -90 and 90, and a longitude between -180 and 360.

Location at Sea

Check if the position is at sea, which is evaluated using ETOPO1, a bathymetry with resolution of 1 minute. It is considered at sea if the interpolated position has a negative vertical level.

This test implicitly requires to be approved by the *Valid Position* test.

Global Range

This test evaluates if the measurement is a possible value in the ocean in normal conditions. The thresholds used are extreme values, wide enough to accommodate all possible values and do not discard uncommon, but possible, conditions.

Regional Range

Digit Rollover

Every sensor has a limit of bits available to store the sample value, with this limit planned to cover the possible range. A spurious value over the bit range would be recorded as the scale rollover, resulting in a misleading value inside the possible scale. This test identifies extreme jumps on consecutive measurements, that area wider than expected, suggesting a rollover error.

The difference on consecutive measurements must be smaller or equal to the threshold to be approved.

Gradient

This test compares

$$X_i = \left| V_i - \frac{V_{i+1} + V_{i-1}}{2} \right|$$

Spike

$$X_i = \left| V_i - \frac{V_{i+1} + V_{i-1}}{2} \right| - \left| \frac{V_{i+1} - V_{i-1}}{2} \right|$$

Tukey 53H

This method to detect spikes is based on the procedure initially proposed by citet{GoringNikora2002} for Acoustic Doppler Velocimeters, and similar to the one adopted by citet{Morello2011}. It takes advantage of the robustness of the median to create a smoother data series, which is then compared with the observation. This difference is normalized by the standard deviation of the observed data series after removing the large–scale variability.

For one individual measurement x_i, where i is the position of the observation, it is evaluated as follows:

1. $x^{(1)}$ is the median of the five points from x_{i-2} to x_{i+2};
2. $x^{(2)}$ is the median of the three points from $x_{i-1}^{(1)}$ to $x_{i+1}^{(1)}$;
3. $x^{(3)}$ **is the defined by the Hanning smoothing filter:** $\frac{1}{4}\left(x_{i-1}^{(2)} + 2x_i^{(2)} + x_{i+1}^{(2)}\right)$
4. x_i is a spike if $\frac{|x_i - x^{(3)}|}{\sigma} > k$, where σ is the standard deviation of the lowpass filtered data.

The default behavior in CoTeDe is to flag 4 if the test yields values higher than $k = 1.5$, and flag 1 if it is lower.

Climatology

$$X_i = \frac{V_{it} - < V_t >}{\sigma}$$

1.5.2 Quality Control procedure

CTD

GTSPP

Test	Flag		Threshold	
	if succeed	if fail	Temperature	Salinity
Valid Date	1	4		
Valid Position	1			
Location at Sea	1			
Global Range	1		-2 to 40 C	0 to 41
Gradient	1	4	10.0 C	5
Spike	1		2.0 C	0.3
Climatology	1			
'Profile Envelop'_				

EuroGOOS

Test	Flag		Threshold	
	if succeed	if fail	Temperature	Salinity
Valid Date	1	4		
Valid Position	1	4		
Location at Sea	1	4		
Global Range	1	4	-2.5 to 40	2 to 41
Digit Rollover	1	4	10.0 C	5
Gradient Cond. • < 500 • > 500	1	4	9.0 C 3.0 C	1.5 0.5
Spike Cond.	1	4		
Climatology	1			

IMOS (Incomplete)

Test	Flag		Threshold	
	if succeed	if fail	Temperature	Salinity
Valid Date	1	3		
Valid Position	1	3		
Location at Sea	1	3		
Global Range	1		-2.5 to 40	2 to 41
Gradient	1	4	10.0 C	5
Spike	1		2.0 C	0.3
Climatology	1			

TSG

Based on AOML procedure. Realtime data is evaluatd by tests 1 to 10, while the delayed mode is evaluated by tests 1 to 15.

1. Platform Identification
2. *Valid Date*
3. Impossible Location
4. *Location at Sea*
5. Impossible Speed
6. *Global Range*
7. Regional Ranges
8. *Spike*
9. Constant Value
10. *Gradient*
11. Climatology

XBT

ARGO

1. Platform Identification
2. *Valid Date* For ARGO, the year also must be later than 1997.
3. Impossible location test
4. Position on land test
5. Impossible speed test
6. *Global range*
7. Regional range test
8. Pressure increasing test
9. *Spike*
10. Top an dbottom spike test: obsolete
11. *Gradient*
12. *Digit Rollover*
13. Stuck value test
14. Density inversion
15. Grey list
16. Gross salinity or temperature sensor drift
17. Visual QC
18. Frozen profile test
19. Deepest pressure test

1.5.3 References

1.6 Anomaly Detection

Anomaly Detection is based on the concept of describe the statistical behaivior of known good data, and than use this as a reference to identify bad data by uncommon characteristics.

1.6.1 Some funcionalities

rank_files()

From a list of data files, analyze all and characterize each measurement by a series of features, like for example the gradient or the difference with the climatology. Than, rank all files based on how unexpected is each feature, i.e. a measurement with a spike too intense, or too different from the climatology would show up first.

rank_files(datadir, varname, cfg=None)

- datadir: root directory with the data to be evaluated
- varname: Variable to be evaluated, like TEMP
- cfg: Q.C. rule to be considered

Return a list of all files inside datadir ordered by the probablity of being all good data.

Calibrate Anomaly Detection

Calibrate the parameters for anomaly detection to best reproduce a preset Q.C. rule (for example: GTSPP). Since the anomaly detection consider simultaneously several features together to make a final decision, it should achieve more consistent results. A measurement with several tests too close to the traditional Q.C. thresholds would be approved by the traditional approach, but would raise suspicious, or even fail, in the anomaly detection approach.

calibrate_anomaly_detection(datadir, varname, cfg=None)

1.7 Fuzzy Logic

This is the implementation of the methodoogy proposed by Timms 2011, Morello 2011 and Morello 2014. If interested in this technique, I strongly recommend you to read those articles. Figure 6 of Morello 2014 values a thousand words.

1.8 History

1.8.1 0.17 - Mar, 2016

- Implementing fuzzy procedures inside CoTeDe, thus removing dependency on scikit-fuzzy. scikit-fuzzy is broken, hence compromising tests and development of CoTeDe.

1.8.2 0.16 - Mar, 2016

- Using external package OceansDB to handle climatologies and bathymetry.

1.8.3 0.15 - Dec, 2015

- Moved procedures to handle climatology to external standalone packages.

1.8.4 0.14 - Aug, 2015

- Interface for human calibration of anomaly detection
- Implemented fuzzy logic criteria

1.8.5 0.13 - July, 2015

- Major improvements in the anomaly detection submodule
- Partial support to thermosalinographs (TSG)
- Working on WOA test to generalize for profiles and tracks
- Adding .json to default QC configuration filenames
- Moved load_cfg from qc to utils

1.8.6 0.12

Since 0.9 some of the most important changes.

- Following CF vocabulary for variables names (PRES, TEMP, PSAL...)
- Partial support to ARGO profiles
- Added density invertion test
- Included haversine to avoid dependency on MAUD.
- tox and travis support.

1.8.7 0.9 - Dec, 2013

- Going public

1.8.8 0.7.3

- Creating fProfileQC()

1.8.9 0.5.4 - Nov, 2013

- Including Tukey53H test

1.8.10 0.5.0

- Implemented ProfileQCCollection

1.8.11 0.4 - Sep, 2013

- gradient and spike tests with depth conditional thresholds
- CruiseQC
- Use default threshold values for the QC tests.

1.8.12 0.1 - May 24, 2013

- Initial release.

1.8.13 QC_ML - 2011

- QC_ML, a machine learning approach to quality control hydrographic data, the initial prototype of CoTeDe. I refactored the system I developed to quality control TSG, to evaluate the PIRATA's CTD stations for INPE. At this point I migrated from my personal Subversion server to Bitbucket, and I lost the history and logs before this point.

1.8.14 2006

- A system to automaticaly quality control TSG data on realtime for AOML-NOAA. The data was handled in a PostgreSQL database, and only the traditional tests were applied, i.e. a sequence of binary tests (spike, gradient, valid position . . .).

1.9 Indices and tables

- genindex
- modindex
- search