Seabird Documentation

Release 0.5.3

Guilherme Castelão

Nov 09, 2017

Contents

CHAPTER 1

User Documentation

Seabird at a glance

1.1 Overview

Seabird is a popular brand of sensors used for hydrographic measurements around the world, and that means a great deal of historical CTD data. These hydrographic profiles are usually available as ASCII files, containing the data itself, and plenty of fundamental metadata, such as position, date, calibration coefficients, and much more. Typically, these files are not hard for a human to interpret, but their format has changed over time, so it is a problem for automated processing.

While working with several years of CTD data from the project PIRATA, I realized that the first problem is just to be able to properly read all the data. I built this Python package with the goal to parse, in a robust way, the different historical Seabird output data file formats, and return that data in a uniform structure.

At this point, my goal is to have an object with attributes parsed from the header, and the data in (NumPy) Masked Arrays, so that the user doesn't need to manually determine the version and details of a .cnv file, but will still have it in a standard pattern, ready to use. Taking advantage of the basic library, this package includes some binary commands to output content as ASCII, but in a persistent format, or to convert it into a NetCDF file.

ATTENTION: this is not an official Sea-Bird package, so if you have trouble with it, please do not complain to Sea-Bird. Instead, open an issue at GitHub (https://github.com/castelao/seabird/issues), and I'll try to help you.

1.2 Installation

1.2.1 Requirements

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

Optional requirement

- NetCDF4, if you want to be able to export the data into netCDF files.

1.2.2 Installing Seabird

Using pip

First you need to install pip, then you can run:

pip install seabird

Alternative

pip install –no-deps seabird

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.

1.3 Getting Started with Seabird

1.3.1 Inside python

```
>>> import seabird
```

In a python script, one can use like this:

```
>>> from seabird.cnv import fCNV
>>> profile = fCNV('your_file.cnv')
>>> profile.attributes  # It will return the header, as a dictionary.
>>> profile.keys() # It will list the available variables.
>>> profile['TEMP2'] # If TEMP2 was on the .keys(), this is how you get the data. It␣
↪will be a masked array.
```

The data from a profile is hence treated as it was a dictionary of Masked Arrays. To plot it, one could:

```
>>> import matplotlib.pyplot as plt
>>> plt.plot(profile['depth'], profile['TEMP'], '.')
>>> plt.show()
```

1.3.2 From the terminal

One way to use is running on the shell the cnvdump. Independent of the historical version of the cnv file, it will return a default structure:

```
seabird cnvdump your_file.cnv
```

That can be used in a regular shell script. For example, let's consider a directory cruise1 with several sub directories, one for each leg of the cruise. One could list all the latitudes of each CTD cast like:

```
for file in `find ./cruise1 -iname '*.cnv'`
do seabird cnvdump $file | grep latitude
done
```

Now let's get that list ordered by the latitude:

```
for file in `find ./cruise1 -iname '*.cnv'`
do
    echo -n  `seabird cnvdump $file | grep latitude`
    echo -n " "
    echo $file
done | sort -n > mylist.txt
```

To convert a .cnv to a standard NetCDF, run:

```
seabird cnv2nc your_file.cnv
```

1.3.3 More examples

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

1.4 Indices and tables

- genindex
- modindex
- search