

My Project

Generated by Doxygen 1.8.13

Contents

1 Python module for binary_c

1

Chapter 1

Python module for binary_c

Based on a original work by Jeff Andrews (can be found in old_solution/ directory) updated and extended for Python3 by Robert Izzard, David hendriks

Warning : THIS CODE IS EXPERIMENTAL!

r.izzard@surrey.ac.uk http://personal.ph.surrey.ac.uk/~ri0005/binary_c.html
09/06/2019

Requirements

- Python3
- binary_c version 2.1+
- requirements.txt (no?)

Environment variables

Before compilation you should set the following environment variables:

- required: `BINARY_C` should point to the root directory of your binary_c installation
- recommended: `LD_LIBRARY_PATH` should include `$BINARY_C/src` and whatever directories are required to run binary_c (e.g. locations of libgsl, libmemoize, librinterpolate, etc.)
- recommended: `LIBRARY_PATH` should include whatever directories are required to build binary_c (e.g. locations of libgsl, libmemoize, librinterpolate, etc.)

Build instructions

To build the module, make sure you have built `binary_c` (with `make` in the `binary_c` root directory), its shared library (with `make libbinary_c.so` in the `binary_c` root directory), and set environment variables as described above, then run the following code in t:

```
make clean
make
```

Then to test the Python module:

```
python3 ./python_API_test.py
```

You will require whatever libraries with which `binary_c` was compiled, as well as the compiler with which Python was built (usually `gcc`, which is easily installed on most systems).

If you want to be able to import the `binary_c` module correctly for child directories (or anywhere for that matter), execute or put the following code in your `.bashrc/.zshrc`:

```
export LD_LIBRARY_PATH=<full path to directory containing libbinary_c_api.so>:$LD_LIBRARY_PATH
export PYTHONPATH=<full path to directory containing libbinary_c_api.so>:$PYTHONPATH
```

Usage notes

When running a jupyter notebook and importing `binary_c`, it might happen that the module `binary_c` cannot be found. I experienced this when I executed Jupyter Notebook from a virtual environment which didnt use the same python (version/binary/shim) as the one I built this library with. Make sure jupyter does use the same underlying python version/binary/shim. That resolved the issue for me.

Also: I figured that having `binaryc` output the log like "`<LOG HEADER> t=10e4 ...`" (i.e. printing the parameter names as well as their values) would be useful because in that way one can easily have python read that out automatically instead of having to manually copy the list of parameter names.

See `examples/ dir` for some working examples