chinet Documentation

Thomas-Otavio Peulen

Apr 02, 2020

CONTENTS

chinet is a C++ library to create optimize, sample, and archive global models. A global model is a model, that unites multiple data-sets and seeks for a joint description of the united dataset.

Global models can unite datasets of the same kind or datasets of different types. A typical examples of a global model in fluorescence experiments is the joint description of multiple fluorescence correlation curves in a titration and the joint description of multiple fluorescence decay curves reporting on FRET in a biomolecular structure by a single structural model.

Computing a global model with a large diverse set of different data can be computationally expensive. To reduce the computational costs and to decrease the evaluation time of a global model defined by chinet, the mutual dependencies of the model parameters are modeled by a graph structure that connects "computing nodes". When a a set of parameters is changed only computing nodes that are affected by these changes are evaluated. Independent nodes are evaluated in parallel.

The state of the evaluation graph can be written to a database for documentation purposes and reconstructed using unique identifies provided by the database.

chinet is NOT intended as ready-to-use software for specific application purposes.

CHAPTER

ONE

CONTENTS

1.1 Installation

`chinet` can either be installed from prebuilt binaries or from the source code. For most users it is recommended to install `chinet` using the prebuilt binaries. Below the installation using prebuilt binaries and the prerequisites to compile `chinet` are briefly outlined.

1.1.1 Prebuilt binaries

It is recommended to install `chinet` for Python environments via `conda` by.

```
conda install -c tpeulen chinet
```

Alternatively, `chinet` can be installed via `pip`.:

```
pip install chinet
```

1.1.2 Compliation

`chinet` can be compiled and installed using the source code provided in the git repository.

```
git clone --recursive https://github.com/tpeulen/chinet.git
cd chinet
sudo python setup.py install
```

To compile `chinet` a set of prerequisites need to be fulfilled:

1. An installed compiler.
2. The HDF5 library with C/C++ include files.
3. A recent 64bit Python installation with include files.
4. cmake
5. SWIG

Windows

On windows `chinet` is best compiled with the Visual Studio 2017. For compilation the Visual Studion Community edition is sufficient. In addition to Visual Studio the libraries and the include files as listed above need to be installed. The prebuilt binaries are compiled on Windows 10 with using 64bit anaconda Python environments miniconda using the conda build recipe that is provided with the source code in the `conda-recipe` folder.

MacOS

For MacOS the prebuilt binaries are compiled on MacOS 10.13 with the Apple clang compiler using a anaconda distribution and the provided `conda-recipe`.

Linux

The Linux prebuilt binaries are compiled on Ubuntu 18.04 in an anaconda distribution and the provided `conda-recipe`.

Conda

A conda recipe is provided in the folder 'conda-recipe' to build the chinet library with the conda build environment.

To build the library download the recipe, install the conda build package and use the provided recipe to build the library.

```
conda install conda-build
conda build conda-recipe
```

1.2 Curves

The library `chinet` uses the class `Curve` to save and operate on the data of a model. One dimensional models derive from the `Curve` class. A `Curve` object has properties for the x-values, the y-values, and the errors of the x- and y-values.

```
import chinet as flm
import numpy as np

curve = flm.Curve()
x = np.linspace(0, 6, 100)
y = np.sin(x)
curve.set_x(x)
curve.set_y(y)
(y == curve.get_y())
(x == curve.get_x())
```

`Curve` objects have the methods `add`, `mul`, `div`, and `sub` to add, multiply, divide and subtract and that take either `Curve` objects or floating point numbers as arguments. These operations act element-wise on the y-axis values of the `Curve` objects. In addition to these elementary operations, the content of a `Curve` object can be freely shifted with respect to the x-axis. For non-integer shifts the y-values are linearly interpolated. By default, if no error is provided for the y-values the error is initialized with ones.

```
import chinet as flm
import numpy as np
import pylab as p

curve = flm.Curve()
x = np.linspace(0, 6, 100)
y = np.sin(x)
curve.set_x(x)
curve.set_y(y)

p.plot(curve.get_x(), curve.get_y())

curve.shift(1.5)
p.plot(curve.get_x(), curve.get_y())

curve.add(curve)
p.plot(curve.get_x(), curve.get_y())

curve.mul(3.1)
p.plot(curve.get_x(), curve.get_y())

p.show()
```

Above, shifting, addition, and multiplication are illustrated for a curve. Note, the multiplication operation can also be used with two curves.

`Curve` objects can be saved and loaded using the methods `save` and `from_json`

```
import chinet as flm
import numpy as np

curve = flm.Curve()
x = np.linspace(0, 6, 100)
y = np.sin(x)
curve.set_x(x)
curve.set_y(y)

curve.save("test.json")
c = flm.Curve()
c.from_json("test.json")
```

A

1.3 C++ API

1.4 Glossary

FRET rate constant The FRET rate constant, $latex k_{RET}$, quantifies the FRET process by the number of quanta transferred from the donor's excited state to the acceptor per time. It depends on the mutual dipole orientation of the donor and the acceptor fluorophore, the distance between the donor and acceptor, R_{DA}, the Förster radius, R_0 of the dye pair, and the corresponding fluorescence lifetime of the donor in the absence of FRET, τ_0. The orientation factor κ^2 captures The mutual dipole orientation.

$$k_{RET} = \frac{1}{\tau_0}\kappa^2 \left(\frac{R_0}{R_{DA}}\right)^6$$

Note, for the calculation of the FRET rate constant the fluorescence lifetime has to match the Förster radius. Meaning the fluorescence lifetime of the corresponding donor fluorescence quantum yield, Φ_F^{D0} should be used.

CLSM confocal laser scanning microscopy

PDA Photon Distribution Analysis

MFD (Multiparameter Fluorescence Detection) A MFD experiments is a time-resolved fluorescence experiment which probes the absorption and fluorescence, the fluorescence quantum yield, the fluorescence lifetime, and the anisotropy of the studied chromophores simultaneously (see [KS01])

FCS (Fluorescence correlation spectroscopy) FCS is a method which relies on fluctuations on the recorded signals to characterize molecular interaction such as binding and unbinding, chemical reaction kinetics, diffusion of fluorescent molecules (see [EM74] and [MEW72])

FRET efficiency The FRET efficiency is the yield of a FRET process. A FRET process transfers energy from the excited state of a donor fluorophore to an acceptor fluorophore. The number of donor molecules in an excited state which transfers energy to an acceptor defines the yield of this energy transfer.

$$E = \frac{transfered}{excited}$$

Practically, mostly the donor and acceptor fluorescence intensities are used to obtain an experimental estimate for this yield.

FRET induced donor decay The FRET-induced donor decay is a time-resolved intensity independent measure of FRET similar to the time-resolved anisotropy defined by the ratio of the donor fluorescence decay in the presence and the absence of FRET (see: [POS17]).

Instrument response function (IRF) IRF stands for instrument response function. In time-resolved fluorescence measurements the IRF is the temporal response of the fluorescence spectrometer to a delta-pulse. Suppose a initially sharp pulse defines the time of excitation / triggers the laser, then recorded response of the fluorescence spectrometer is broadened due to: (1) the temporal response of the exciting light source, (2) the temporal dispersion due to the optics of the instrument, (3) the delay of the light within the sample, and (4) the response of the detector. As the most intuitive contribution to the IRF is the excitation profile, the IRF is sometimes called 'lamp function'. The IRF is typically recorded by minimising the contribution of (3), e.g., by measuring the response of the instrument using a scattering sample, or a short lived dye.

Time-tagged time resolved (TTTR) TTTR stands for time tagged time-resolved data or experiments. In TTTR-datasets the events, e.g., the detection of a photon, are tagged by a detection channel number. Moreover, the recording clock usually registers the events with a high time resolution of a few picoseconds. For long recording times of the detected events, a coarse and a fine clock are combined. The fine clock measures the time of the events relative to the coarse clock with a high time resolution. The time of the coarse and the fine clock is usually called macro and micro time, respectively.

FRET positioning system (FPS) FRET positioning system, FPS, is an approach to determine structural models based on a set of FRET measurements. FPS explicitly considers the spatial distribution of the dyes. This way experimental observables, i.e., FRET efficiencies may be predicted precisely. The FPS-toolkit is available from the web page of the Seidel group of the Heinrich Heine University. An implementation of accessible volume simulations (AV) used in FPS are available as open source.

Time correlated single photon counting (TCSPC) Time correlated single photon counting (TCPSC) is a technique to measure light intensities with picosecond resolution. Its main application is the detection of fluorescent light. A pulsed light source excites a fluorescent sample. A single photon detector records the emitted fluorescence photons. Thus, per excitation cycle, only a single photon is detected. Fast detection electronics records the time between the excitation pulse and the detection of the fluorescence photon. A histogram accumulates multiple detected photons to yield a time-resolved fluorescence intensity decay.

SWIG SWIG is a software development tool that connects programs written in C and C++ with a variety of high-level programming languages. SWIG can be used with different types of target languages including common

scripting languages such as Javascript, Perl, PHP, Python, Tcl and Ruby and non-scripting languages such as C#, D, Go language, Java, Octave, and R. SWIG is free software and the code that SWIG generates is compatible with both commercial and non-commercial projects. `tttrlib` is C/C++ based to provide the capability for a broad variety of languages to interface its provided functionality.

1.5 References

CHAPTER

TWO

INDICES AND TABLES

- genindex
- search

CHAPTER

THREE

LICENSE

chinet is released under the open source MIT license.

BIBLIOGRAPHY

[EM74] Elliot L Elson and Douglas Magde. Fluorescence correlation spectroscopy. i. conceptual basis and theory. *Biopolymers: Original Research on Biomolecules*, 13(1):1–27, 1974.

[KS01] Ralf Kühnemuth and Claus A. M. Seidel. Principles of single molecule multiparameter fluorescence spectroscopy. *Single Molecules*, 2:251–254, 2001. doi:10.1002/1438-5171(200112)2:4<251::aid-simo251>3.0.co;2-t.

[MEW72] Douglas Magde, Elliot Elson, and Watt W Webb. Thermodynamic fluctuations in a reacting system—measurement by fluorescence correlation spectroscopy. *Physical Review Letters*, 29(11):705, 1972.

[POS17] Thomas-Otavio Peulen, Oleg Opanasyuk, and Claus AM Seidel. Combining graphical and analytical methods with molecular simulations to analyze time-resolved fret measurements of labeled macromolecules accurately. *The Journal of Physical Chemistry B*, 121(35):8211–8241, 2017.

INDEX