pyFRET: A Python Library for Single Molecule Fluorescence Data Analysis

Rebecca R. Murphy*†, Sophie E. Jackson†, David Klenerman†

Abstract—Single molecule Förster resonance energy transfer (smFRET) is a powerful experimental technique for studying the properties of individual biological molecules in solution. However, as adoption of smFRET techniques becomes more widespread, the lack of available software, whether open source or commercial, for data analysis, is becoming a significant issue. Here, we present pyFRET, an open source Python package for the analysis of data from single-molecule fluorescence experiments from freely diffusing biomolecules. The package provides methods for the complete analysis of a smFRET dataset, from burst selection and denoising, through data visualisation and model fitting. We provide support for both continuous excitation and alternating laser excitation (ALEX) data analysis. pyFRET is available as a package downloadable from the Python Package Index (PyPI) under the open source three-clause BSD licence, together with links to extensive documentation and tutorials, including example usage and test data. Additional documentation including tutorials is hosted independently on ReadTheDocs. The code is available from the free hosting site Bitbucket. Through distribution of this software, we hope to lower the barrier for the adoption of smFRET experiments by other research groups and we encourage others to contribute modules for specific analysis needs.

Index Terms—smFRET, single-molecule, confocal, python, fluorescence

1 Introduction

Förster resonance energy transfer (FRET) [Forster48] is a physical process that allows the study of molecular interactions and intramolecular distances. FRET is the non-radiative transfer of energy between two fluorescent molecules, where the fraction of energy transferred varies with the sixth power of the inter-fluorophore distance, providing an extremely sensitive readout of the distance between two fluorophores. Since its first demonstration, [ha96], single-molecule FRET (smFRET) has grown in popularity as a tool to investigate the structure and dynamics of biomolecules diffusing in solution [haran03], [schuler02], [weiss00].

In a smFRET experiment, biological molecules are labelled with two fluorescent dyes, selected such that the emission spectrum of one dye (the donor, D) overlaps with the excitation spectrum of the other (the acceptor, A). When the donor and acceptor are physically close in space, exciting the donor dye can result in emission from the acceptor dye, where the proportion of emission from the acceptor and donor, known as the FRET Efficiency, E, depends on the distance, r between the two dyes and R_0:, the Förster distance, a dye dependent constant that describes the dye separation at which 50% energy transfer is achieved.

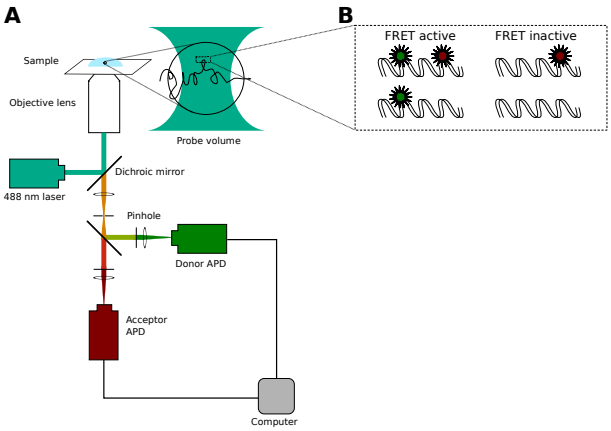

Fig. 1: *Instrumentation for a smFRET experiment.*

$$E = \frac{1}{1+(\frac{r}{R_0})^6}$$

Experimentally, a laser beam is focused through a highly dilute solution of labelled molecules onto a diffraction-limited focal point. When a labelled molecule diffuses through the laser beam, the donor dye is excited and photons are emitted (Fig. 1). Emitted photons are collected through the objective and separated by a dichroic mirror into donor and acceptor photons. The ratio of acceptor to donor photons allows calculation of E for that fluorescent event:

$$E = \frac{n_A}{n_A + \gamma \cdot n_D}$$

for n_A and n_D photons in the acceptor and donor channels respectively and γ an experimentally determined instrument-dependent factor that corrects for unequal detection efficiencies. The thousands of fluorescent events collected during an experiment are used to construct a FRET efficiency histogram, which is typically fitted with a gaussian distribution to identify populations of fluorescent species [ha96], [nir11].

* *Corresponding author: rrm33@cam.ac.uk*
† *Department of Chemistry, University of Cambridge*

A smFRET experiment involves several computational challenges. Bursts of fluorescence emission, corresponding to a molecule diffusing through the laser beam, must be identified against a noisy background and identified bursts must be denoised. Correction of photobleaching effects, donor-acceptor crosstalk and other photophysical artifacts must also be applied to get accurate intramolecular distance information. Multiple methods of burst selection and analysis have been developed and applied to the analysis of smFRET data [weiss00], [deniz01], [gell06], [nir06], [kapanidis05], [muller05], [doose07], [kudryavtsev2012], [eggeling01]. However, software for analysis of smFRET data has thus far been developed on an ad hoc basis, with individual groups preparing and maintaining their own analysis scripts.

This method of software development has created several problems for the smFRET research community that are typical of research programming projects [wilson06], [merali10].

Firstly, there is the problem of "reinventing the wheel" [mirams13]. Within smFRET research groups, programming ability is not a standard skill, despite the need for sophisticated data analysis and use of custom data collection hardware. It is common for researchers with programming skills to maintain their own series of data-analysis scripts which may be wholly dependent on particular hardware tools or analysis packages. Other researchers, who may lack the skills to maintain and develop even simple scripts, are dependent on black-box techniques provided by their colleagues. Consequently, data analysis is dependent on scripts written and maintained by just a few researchers. Loss of programming expertise when these team members leave can result in significant difficulties for the remaining group members, who are then dependent on poorly documented code that they do not fully understand how to use. Furthermore, the lack of available open source software often requires new researchers in the field of smFRET to completely reimplement standard analysis techniques in order to become independently productive.

Secondly, the need for many researchers to develop and maintain their own analyis tools has significant impact on research productivity. The requirement to reimplement standard analysis techniques consumes valuable time that could better be used in experimental research or in developing and benchmarking improved analysis tools. Furthermore, most researchers have no formal training in software engineering, with the result that analysis software can vary hugely in quality and is frequently poorly documented and maintained, making it difficult for other researchers to understand and use. New analysis scipts are often added in an ad hoc manner, with the result that straighforward tasks are performed using an unwieldy mess of spaghetti code, transforming simple modifications into complex undertakings requiring significant time investment. Poorly maintained code adds an additional barrier to open sharing of resources as groups are embarrassed to share low-quality software.

Finally, there is the issue of research reproducibility. Different research groups use widely differing tools to complete similar tasks. New methods of data collection and analysis are frequently developed [kapanidis05], [nir06], [sisamakis2010]. However, when software is not released to the community, it is difficult for researchers, who must often implement poorly described methodologies entirely from scratch, to verify results or to adopt new techniques in their own research. As a consequence, new techniques are poorly benchmarked, making it difficult to understand whether a new analysis adds quality or merely complexity, whilst adoption of useful new methods is relatively slow. These three issues of productivity, reliability and reproducibility, all linked to the problem of poorly maintained softwared and lack of software development skills, are now becoming a key bottleneck in smFRET research.

We have developed pyFRET, a fully open source library, written in Python, for the analysis of smFRET data. To our knowledge, this is the first open source code ever released by the smFRET research community. Our library aims to address the issues described above by providing a simple toolkit for smFRET data analysis. pyFRET is a small library, consisting of just 700 lines of Python code (including inline comments). However, it contains functions for all key steps in analysis of smFRET data, including burst selection; cross-talk subtraction and burst denoising; data visualisation; and construction and simple fitting of FRET efficiency histograms. In providing this toolkit to the smFRET research community, we hope to facilitate the wider adoption of smFRET techniques in biological research as well as to provide a framework for open communication about and sharing of data analsyis tools.

2 Design and Implementation

2.1 Implementation

pyFRET provides two key classes for manipulation of smFRET data. The FRET data object describes two fluorescence channels, corresponding to time-bins containing photons collected from donor (the donor channel, D) and acceptor (the acceptor channel, A) fluorophores. The ALEX data object describes four fluorescence channels, corresponding to the four temporal states in a smFRET experiment using Alternating Laser Excitation (ALEX), namely the donor channel when the donor laser is switched on (D_D); the donor channel when the acceptor laser is switched on (D_A); the acceptor channel when the donor laser is on (A_D); and the acceptor channel when the acceptor laser is on (A_A). These data channels are implemented as numpy arrays, allowing efficient computation and selection operations.

The data analysis workflow is illustrated in Figure 2. Following initialization of data objects, background subtraction, event selection, cross-talk correction and calculation of the FRET efficiency can each be performed with a single call to a pyFRET function.

pyFRET provides built-in functions to generate the most common plot types used in the field. For example, proximity ratio histograms, which allow identification and analysis of different fluorescent populations, can be generated using the proximity_ratio method (Fig. 3 A). For ALEX data, scatter plots with projected histograms can be constructed (Fig. 3 B). Further plotting options are shown in Fig. 3 C and D.

2.2 Compatibility Considerations

pyFRET is written in Python. Both Python 2 (v2.7) and Python 3 (v3.3) are supported. pyFRET requires three further

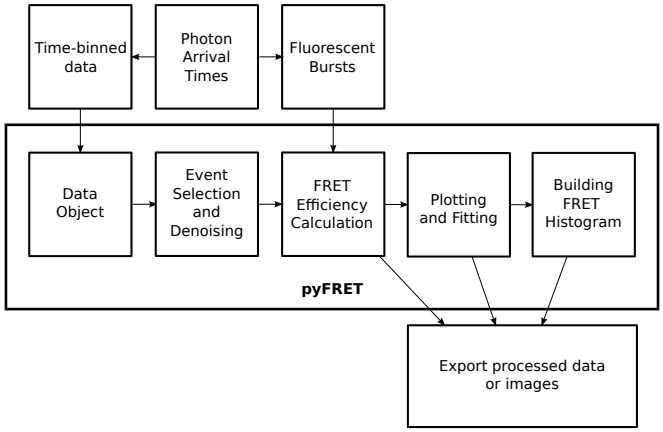

Fig. 2: *Typical workflow for data analysis using pyFRET.*

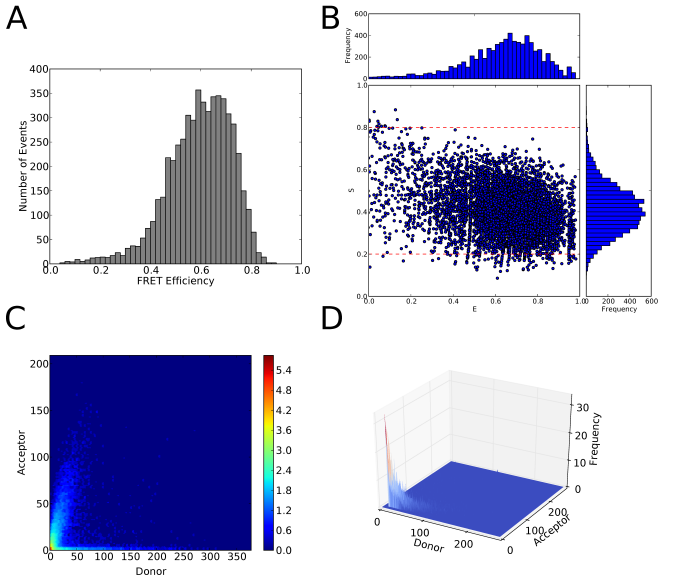

Fig. 3: *Figures made using pyFRET. A) A Proximity Ratio histogram. B) A scatter-plot of FRET efficiency and fluorophore stoichiometry from ALEX data. C) A heatmap of event frequencies. D) A 3D plot of event frequencies.*

Python libraries, namely numpy [numpy] and scipy [scipy] for data manipulation, and matplotlib [matplotlib] for data visualisation. Installation of pyFRET using the pip install method supported by PyPI will facilitate automatic installation of these packages if they are not already included in your Python build.

The lack of open source software in the smFRET community has led to a proliferation of esoteric file-types used for data collection and storage. To make pyFRET as usable as possible for a wide range of smFRET researchers, the pyFRET data structures can be initialised using arrays of time-binned photons. The tutorial also demonstrates using pyFRET's file-parsing functions scripts to create pyFRET objects from common filetypes.

pyFRET currently provides basic tools for analysis and visualisation of smFRET data. In the interest of providing the pyFRET infrastructure to smFRET researchers at an early stage, we are choosing to release our software at a relatively early stage of development. pyFRET provides a complete toolchain for analysis of time-binned smFRET data, but does not currently include a burst-search algorithm for identification of fluorescent bursts from photon arrival times [nir06]. Fluorescent bursts identified using a burst search algorithm can be analysed using pyFRET by initialising a pyFRET data object from the paired burst photon frequencies. Denoising and cross-talk correction is achieved in exactly the same manner as for time-binned data, but thresholding is not required. We encourage researchers who wish to use pyFRET in its current implementation for data visualisation and analysis, but whose data consists of time-stamped photon arrivals to apply their own burst selection algorithms to generate arrays of fluorescent bursts that can be manipulated using pyFRET methods. We also welcome pull requests and contributions to the bitbucket repository.

3 Experimental Methods

We tested the pyFRET library using DNA duplexes dual-labelled with Alexa Fluor 488 and Alexa Fluor 647. The duplex sequences and labelling sites are shown in Tables 1 (Donor strand) and 2 (Acceptor strands). DNA duplexes were prepared by mixing a 1.1 molar excess of the appropriate acceptor strand with the donor strand, heating to 95 C for 10 minutes, then gradual cooling to room temperature. FRET data were collected for 15 minutes using continuous excitation at 488 nm and binned in intervals of 1 ms. ALEX data were collected for 15 minutes using alternating excitation at 488 and 640 nm, with a modulation rate of 0.1 ms, a dead-time of 0.1 μ s and a delay compensation of 3 μ s. ALEX data were then binned in intervals of 1 ms. The scripts and configuration files used to analyse these data using pyFRET can be found in the "bin" folder of the pyFRET repository.

4 Results

As an example of the analysis that can be performed using pyFRET, we collected data from dual-labelled DNA duplexes with various dye-dye separation distances, using both FRET and ALEX excitation patterns. We then analysed the data using

Donor Construct	Sequence
Donor	TACTGCCTTTCTGTATCGC5TATCGCGTAGTTACCTGCCTTGCATAGCCACTCATAGCCT

TABLE 1: *DNA sequence of the donor-labelled strand, where 5 is a deoxy-T nucleotide, labelled with Alexa Fluor 488 at the C6 amino position*

Separation	Acceptor Sequence
4	AGGCTATGAGTGGCTATGCAAGGCAGGTAACTACGCGATAAGCGA6
6	AGGCTATGAGTGGCTATGCAAGGCAGGTAACTACGCGATAAGCGATA6
8	AGGCTATGAGTGGCTATGCAAGGCAGGTAACTACGCGATAAGCGATACA6
10	AGGCTATGAGTGGCTATGCAAGGCAGGTAACTACGCGATAAGCGATACAGA6
12	AGGCTATGAGTGGCTATGCAAGGCAGGTAACTACGCGATAAGCGATACAGAAA6

TABLE 2: *Preparing the dual-labelled dsDNA. An acceptor-labelled ssDNA, with the sequence shown was annealed to the indicated donor construct, to yield a dual-labelled construct with the labels separated by the given number of base pairs. In the displayed acceptor-strand sequences, 6 is a deoxy-T nucleotide, labelled with Alexa Fluor 647 at the C6 amino position..*

the pyFRET analysis pipeline. Timebins were background corrected and events were selected using a fixed threshold. FRET efficiency histograms were constructed and fitted to a single gaussian distribution. The mean FRET efficiencies were then plotted against the dye separation distance to show the characteristic sigmoidal curve. Results of the analysis are show in Fig. 4 (FRET) and Fig. 5 (ALEX). An example analysis script to produce a fitted smFRET histogram is shown below. Here, the parameters auto_donor, auto_acceptor, cross_DtoA, cross_AtoD and g_factor are user-supplied experimentally determined correction factors; T_donor and T_acceptor are user-supplied thresholds for event selection. Auto_donor and auto_acceptor are the background autofluorescence in the donor and acceptor channels respectively; cross_DtoA and cross_AtoD are the crosstalk in the acceptor and donor channels, caused by direct excitation of the acceptor dye by the donor laser and the donor dye by the acceptor laser respectively; g_factor is the correction factor γ described above; and T_donor and T_acceptor are photon count thresholds above which a time-bin is classified as containing a fluorescent event. Realistic parameter values are shown in the snippet below.

```
from pyFRET import pyFRET as pft

# read data
my_directory = "path/to/my/files"
list_of_files = ["file1.csv", "file2.csv", "file3.csv"]
my_data = pft.parse_csv(my_directory, list_of_files)

# define constants
auto_donor = 0.3        # background autofluorescence
auto_acceptor = 0.2
T_donor = 15            # photon count thresholds
T_acceptor = 15
cross_DtoA = 0.05       # cross-talk
cross_AtoD = 0.01
g_factor = 1.0          # detection correction factor


# background correction and event selection
my_data.subtract_bckd(auto_donor, auto_acceptor)
my_data.threshold_AND(T_donor, T_acceptor)
my_data.subtract_crosstalk(cross_DtoA, cross_AtoD)

# make histogram of FRET efficiency and fit
my_data.build_histogram(filepath, csvname, \
```

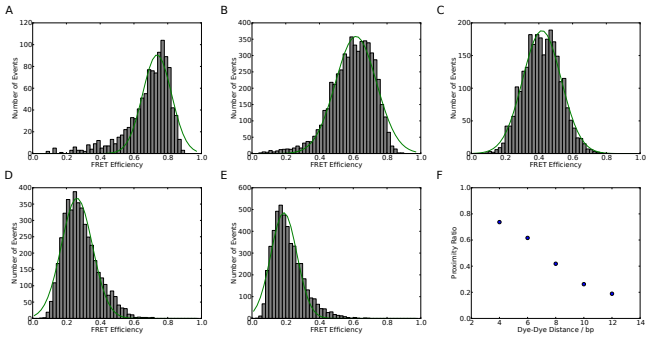

Fig. 4: *Analysis of FRET data from DNA duplexes using pyFRET. A - E: Fitted FRET histograms from DNA duplexes labelled with a dye-dye separation of 4, 6, 8, 10 and 12 base pairs respectively. F) Characteristic sigmoidal curve of FRET efficiency against dye-dye distance.*

```
    gamma=g_factor, bin_min=0.0, bin_max=1.0, \
    bin_width=0.02, image=True, imgname="my_histogram", \
    imgtype="png", gauss=True, gaussname="gaussfit")
```

5 Conclusion

pyFRET is available to download from PyPI under an open source three-clause BSD licence. The source code is available from Bitbucket [bitbucket]. Documentation can also be found there, whilst a more extensive tutorial, including example scripts, can be found on our website at ReadTheDocs [RTD].

pyFRET currently provides basic tools for burst selection and denoising, based on simple thresholding and noise subtraction techniques. We are aware that more sophisticated methodologies exist and are currently working to produce and open source burst selection algorithm based on photon arrival times [nir06] as well as stochastic denoising algorithms [kudryavtsev2012]. We have also developed a novel analysis method based on Bayesian statistics [murphy14], for which source code is available (https://bitbucket.org/rebecca_roisin/fret-inference}) and which will be folded into the pyFRET library. We are also working to increase support for the wide variety of file formats that result from custom-built data collection hardware.

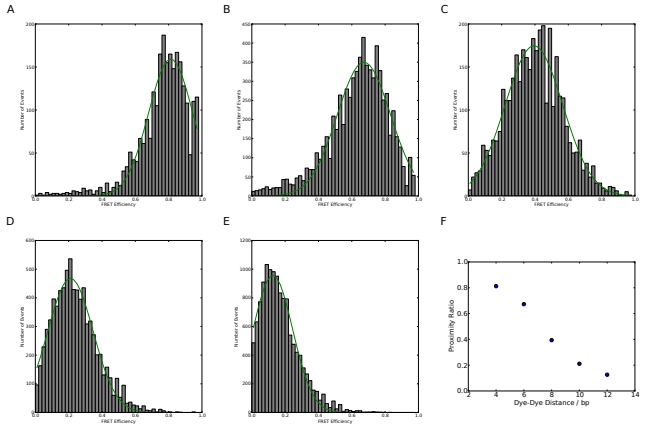

Fig. 5: *Analysis of ALEX data from DNA duplexes using pyALEX. A - E: Fitted FRET histograms from DNA duplexes labelled with a dye-dye separation of 4, 6, 8, 10 and 12 base pairs respectively. F) Characteristic sigmoidal curve of FRET efficiency against dye-dye distance.*

smFRET is a fast-developing and active research field and we want to support scientific progress through development of high-quality usable software. We are keen to work with others to enable their use of and contribution to the pyFRET library. We welcome requests for custom analysis requirements and are happy to support others who wish to contribute additional code to the pyFRET infrastucture.

REFERENCES

[Forster48] T. Förster. *Zweischenmolekulare energiewanderung und-fluoreszenz*, Annalen der Physik, 2:55-75, 1948.

[ha96] T. Ha, T. Enderle, D. F. Ogletree, D. S. Chemla, P. R. Selvin and S. Weiss. *Probing the interaction between two single molecules: Fluorescence resonance energy transfer between a single donor and a single acceptor*, Proc. Natl. Acad. Sci. U.S.A., 93(13):6264-6268, June 196.

[haran03] G. Haran. *Single-molecule fluorescence spectroscopy of biomolecular folding*, J. Phys.: Condens. Matter, 15(32):R1291-R1317, August 2003.

[schuler02] B. Schuler, E. A. Lipman and E. A. Eaton. *Probing the free-energy surface for protein folding with single-molecule fluorescence spectroscopy*, Nature, 419(6908):743-747, October 2002.

[weiss00] S. Weiss. *Measuring conformational dynamics of biomolecules by single molecule fluorescence spectroscopy*, Nat. Struct. Mol. Biol, 7(9):724-729, September 2002.

[deniz01] A. A. Deniz, T. A. Lawrence M. Dahan D. S. Chemla, P. S. Schultz and S. Weiss. *Ratiometric single-molecule studies of freely diffusing molecules*, Annu. Rev. Phys. Chem., 52:233-253, 2002.

[gell06] C. Gell, D. Brockwell and A. Smith. *Handbook of single molecule fluorescence*, OUP (Oxford), 2006.

[nir06] E. Nir, X. Michalet, K. M. Hamadani, T. A. Laurence, D. Neuhauser, Y. Kovchegov and S. Weiss. *Shot-noise limited single-molecule FRET histograms: Comparison between theory and experiments*, J. Phys. Chem. B, 110(44):22103-22124, November 2006.

[kapanidis05] A. N. Kapanidis, T. A. Laurence, N. K. Lee, E. Margeat, X. Kong and S. Weiss. *Alternating-laser excitation of single molecules*, Acc. Chem. Res., 38:532-533, 2005.

[muller05] B. K. Muller, E. Zaychikov, C. Brauchle and D. C. Lamb. *Pulsed interleaved excitation*, Biophys. J., 89(5):3502-3522, November 2005.

[doose07] S. Doose, M. Heilemann, X. Michalet, S. Weiss and A. N. Kapanidis. *Periodic acceptor excitation spectroscopy of single molecules*, Eur. Biophys. J., 36:669-674, 2007.

[sisamakis2010] E. Sisamakis, A. Valeri, S. Kalinin, P. J. Rothwell and C. A. M. Seidel. *Accurate Single-Molecule FRET Studies Using Multiparameter Fluorescence Detection*, Methods in Enzymology, 475:455-514, 2010.

[kudryavtsev2012] V. Kudryavtsev, M. Sikor, S. Kalinin, D. Mokranjac, C. A. M. Seidel and D. C. Lamb. *Combining MFD and PIE for Accurate Single-Pair Förster Resonance Energy Transfer Measurements*, ChemPhysChem, 13:1060-1078, 2012.

[eggeling01] C. Eggeling, S. Berger, L. Brand, J. R. Fries, J. Schaffer, A. Volkmer and C. A. M. Seidel. *Data registration and selective single-molecule analysis using multi-parameter fluorescence detection*, J. Biotechnol., 86:163-180, 2001.

[wilson06] G. Wilson. *Where's the real bottleneck in scientific computing?*, American Scientist, 94:5-6, 2006.

[merali10] Z. Merali. *Computational science: Error, why scientific programming does not compute*, Nature, 467:775-777, 2010.

[mirams13] G. R. Mirams,C. J. Arthurs, M. O. Bernabeu, R. Bordas, J. Cooper, A. Corrias, Y. Davit, S.-J. Dunn, A. G. Fletcher, D. G. Harvey, M. E. Marsh, J. M. Osborne, P. Pathmanathan, J. Pitt-Francis, J. Southern, N. Zemzemi and D. J. Gavaghan. *Chaste: An Open Source C++ Library for Computational Physiology and Biology*, PLOS Comp. Biol., 9:e1002970-e1002970, 2013.

[murphy14] R. R. Murphy, G. Danezis, M. H. Horrocks, S. E. Jackson and D. Klenerman. *Bayesian Inference of Accurate Population Sizes and FRET Efficiencies from Single Diffusing Biomolecules*, Anal. Chem., http://dx.doi.org/10.1021/ac501188r, 2014.

[numpy] S. van der Walt, S. C. Colbert and G. Varoquaux. *The NumPy Array: A Structure for Efficient Numerical Computation*, Computing in Science & Engineering, 13:22-30, 2011.

[scipy] K. J. Millman and M. Aivazis. *Python for Scientists and Engineers*, Computing in Science & Engineering, 13:9-12, 2011.

[matplotlib] J. D. Hunter. *Matplotlib: A 2D graphics environment*, IEEE Comp. Soc., 9(3):90-95, 2007.

[nir11] E. Nir, X. Michalet, K. M. Hamadani, T. A. Laurence, D. Neuhauser, Y. Kovchegov and S. Weiss. *Shot-Noise Limited Single-Molecule FRET Histograms: Comparison between Theory and Experiments*, J. Phys. Chem. B., 110(44):22103-22124, November 2011.

[RTD] http://pyfret.readthedocs.org/

[bitbucket] https://bitbucket.org/rebecca_roisin/pyfret_release