PyFolding: An open-source software package for graphing, simulation and analysis of the biophysical properties of proteins

Alan R. Lowe[1,2,3]*, Albert Perez-Riba[4], Laura S. Itzhaki[4] & Ewan R.G. Main[5]*

[1] London Centre for Nanotechnology
17-19 Gordon Street, London
WC1H 0AH, UK

[2] Structural & Molecular Biology, University College London,
Gower Street, London,
WC1E 6BT, UK

[3] Department of Biological Sciences, Birkbeck College, University of London
Malet Street, London
WC1E 7HX, UK

[4] Department of Pharmacology, University of Cambridge
Tennis Court Road, Cambridge
CB2 1PD, UK

[5] School of Biological and Chemical Sciences, Queen Mary University of London
Mile End Road, London
E1 4NS, UK

*corresponding authors (ARL: a.lowe@ucl.ac.uk, ERGM: e.main@qmul.ac.uk)

[Abstract]

For many years, curve fitting software has been heavily utilized to fit simple models to various types of biophysical data. Although such software packages are easy to use for simple functions, they are often expensive and present substantial impediments to applying more complex models or for the analysis of large datasets. One field that is relient on such data analysis is the thermodynamics and kinetics of protein folding. Over the past decade, increasingly sophisticated analytical models have been generated, but without simple tools to enable routine analysis. Consequently, users have needed to generate their own tools or otherwise find willing collaborators. Here we present *PyFolding*, a free, open source, and extensible Python framework for graphing, analysis and simulation of the biophysical properties of proteins. To demonstrate the utility of *PyFolding*, we have used it to analyze and model experimental protein folding and thermodynamic data. Examples include: (i) multi-phase kinetic folding fitted to linked equations, (ii) global fitting of multiple datasets and (iii) analysis of repeat protein thermodynamics with Ising model variants. Moreover, we demonstrate how *Pyfolding* is easily extensible to novel functionality beyond applications in protein folding via the addition of new models. Example scripts to perform these and other operations are supplied with the software, and we encourage users to contribute notebooks and models to create a community resource. Finally, we show that *PyFolding* can be used in conjunction with Jupyter notebooks as an easy way to share methods and analysis for publication and amongst research teams.

[Introduction]

The last decade has seen a shift in the analysis of experimental protein folding and thermodynamic stability data from the fitting of individual datasets using simple models to increasingly complex models using global optimization over multiple large datasets [examples include Refs: (3-21)]. This shift in focus has required moving from user-friendly, but expensive software packages to bespoke solutions developed in computing environments such as MATLAB and Mathematica or by using in-house solutions [examples include: (3, 6, 12, 21, 22)]. However, as these methods of analysis have become more essential, simple curve fitting software no longer provides sufficient flexibility to implement the models. Thus, there is increasingly a need for substantially more computational expertise than previously required. In this respect the protein folding field contrasts with other fields, for example x-ray crystallography, where free or inexpensive and user-friendly interfaces and analysis packages have been developed (23).

Here we present *PyFolding*, a free, open-source and extensible framework for graphing, analysis and simulation. At present, it is customised for the analysis and modelling of protein folding kinetics and thermodynamic stability. To demonstrate these and other functions we present a number of examples as Jupyter notebooks. The software, coupled with the supplied models / Jupyter (iPython) notebooks, can be used by researchers with less programming expertise to access more complex models/analyses and share their work with others. Moreover, *PyFolding* also enables researchers to automate the time-consuming process of combinatorial calculations, fitting data to multiple models or multiple models to specific data. This enables novice users to simply replace the filenames of the datasets with their own and execute the same calculations for their systems. For more advanced users, new models and functionality can be added with ease by utilising the template models. The Jupyter notebooks provided also show how *PyFolding* provides an easy way to share analysis for publication and amongst research teams.

[Materials & Methods]

PyFolding was developed using Python 2.7 and additional libraries NumPy, SciPy and Matplotlib. Analyses were performed on either an i5 Macbook Pro with 8Gb RAM running macOS Sierra, a Dell Precision T3600 Workstation running Ubuntu 16.04LTS with 64Gb RAM and an NVIDIA GTX1080 GPU or a virtual PC running Windows 10 (64 bit) in VirtualBox on an i7 Macbook Air. Example data for the associated notebooks were taken from existing publications or extracted from original publications using *engauge digitizer* (https://github.com/markummitchell/engauge-digitizer). The *PyFolding* software, notebooks and example data are distributed through github at https://github.com/quantumjot/PyFolding.

[Results & Discussion]

PyFolding is implemented in Python and is distributed as a lightweight, open-source library through *github* and can be downloaded with instructions for installation from the authors' site[1]. *PyFolding* has several dependencies, requiring Numpy, Scipy and Matplotlib. These are now conveniently packaged in several Python frameworks, enabling easy installation of *PyFolding* even for those who have never used Python before (described in the "SETUP.md" file of *PyFolding* and as a series of instructional videos to demonstrate the installation and use of *PyFolding*[2]). As part of *PyFolding*, we have provided many commonly used folding models, such as two- and three-state equilibrium folding and various equivalent kinetic variations, as standard (S.I. Jupyter notebook 1-4 & 8). Functions and models themselves are open source and are thus available for inspection or modification by both reviewers and authors. Moreover, due to the open source nature, users can introduce new functionality by adding new models into the library building upon the template classes provided. We encourage users to contribute notebooks and models to create a community resource.

Fitting and evaluation of typical folding models within *PyFolding*: *PyFolding* uses a hierarchical representation of data internally. Proteins exist as objects that can have metadata as well as multiple sets of kinetic and thermodynamic data associated with them. Input data such as chevron plots or equilibrium denaturation curves can be supplied as comma separated value files (.CSV). Once loaded, each dataset is represented in *PyFolding* as an object, associating the data with numerous common calculations. Models are represented as functions that can be associated with the data objects you wish to fit. As such, datasets can have multiple models and *vice versa* enabling automated fitting and evaluation (S.I. Jupyter notebooks 1-3). Parameter estimation for simple (non-Ising) models is performed using the Levenberg–Marquardt non-linear least-mean-squares optimization algorithm to optimize the appropriate objective function [as implemented in SciPy (24)]. The output variables (with standard error) and fit of the model to the dataset (with R^2

[1] https://github.com/quantumjot/PyFolding
[2] https://github.com/quantumjot/PyFolding/wiki

coefficient of determination & 95 % confidence levels) can be viewed within *PyFolding* and/or the fit function and parameters written out as a CSV file for plotting in your software of choice (S.I Jupyter notebook 1-3). Importantly, by representing proteins as objects, containing both kinetic and equilibrium datasets, *PyFolding* enables users to perform and automate higher-level calculations such as Phi-value analysis (25, 26), which can be tedious and time-consuming to perform otherwise (S.I Jupyter notebook 3). Moreover, users can define their own calculations so that more complex data analysis can be performed. For example, multiple kinetic phases of a chevron plot (fast and slow rate constants of folding) can be fitted to two linked equations describing the slow and fast phases of a 3-state folding regime (Figure 1, S.I Jupyter Notebook 4). We believe that this type of fitting is extremely difficult to achieve with the commercial curve fitting software commonly employed for analysing these data, owing to the complexity of parameter sharing amongst different models and datasets.

More "complex" fitting, evaluation and simulations using the Ising Model: Ising models are statistical thermodynamic nearest-neighbour models that were initially developed for ferromagnetism (27, 28). Subsequently, they have been used with great success in both biological and non-biological systems to describe order-disorder transitions (12). Within the field of protein folding and design they have been used in a number of instances to model phenomena such as helix to coil transitions, beta-hairpin formation, prediction of protein folding rates/thermodynamics and with regards to the postulation of downhill folding (6, 12, 20, 29-34). Most recently two types of one-dimensional (1-D) variants have been used to probe the equilibrium and kinetic un/folding of repeat proteins (3, 12, 17, 21, 22, 35, 36). The most commonly used, and mathematically less complex, has been the 1-D homopolymer model (also called a homozipper). Here, each arrayed element of a protein is treated as an identical, equivalent independently folding unit, with interactions between units via their interfaces. Analytical partition functions describing the statistical properties of this system can be written. By globally fitting this model to, for example, chemical denaturation curves for a series of proteins that differ only by their number of identical units, the intrinsic energy of a repeated unit and the interaction energy between the folded units can be delineated. However, this simplified model cannot describe the majority of naturally occurring proteins where subunits differ in their stabilities, and varying topologies and/or non-canonical interfaces exist. In these cases, a more sophisticated and mathematically more complex heteropolymer Ising model must be used. Here the partition functions required to fit the data are dependent on the topology of interacting units and thus are unique for each analysis.

At present, there is no freely available software that can globally fit multiple folding datasets to a heteropolymer Ising model, and only a few that can adequately implement a homopolymer Ising model. Therefore, most research groups have had to develop bespoke solutions to enable analysis of their data (3, 21, 22, 35, 36). Significantly, in *PyFolding* we have implemented methods to enable users to easily fit datasets of proteins with different topologies to both the homozipper and heteropolymer Ising models. To achieve this goal *PyFolding* presents a flexible framework for defining any non-degenerate 1-D protein topology using a series of primitive protein folding "domains/modules" (Figure 2). Users define their proteins' 1-D topology from these domains (S.I. Jupyter notebook 5-6). *PyFolding* will then automatically calculate the correct partition function for the defined topology, using the matrix formulation of the model [as previously described (12)], and globally fit the equations to the data as required (S.I. Jupyter notebook 5-6). The same framework also enables users to simulate the effect of changing the topology, a feature that is of great interest to those engaged in rational protein design (S.I. Jupyter notebook 7).

To determine a globally optimal set of parameters that minimises the difference between the experimental datasets and the simulated unfolding curves, *PyFolding* uses the stochastic differential evolution optimization algorithm (37) implemented in SciPy (24). In practice, experimental datasets may not adequately constrain parameters during optimization of the objective function, despite yielding an adequate curve fit to the data. It is therefore essential to carefully assess the output of the model to verify the validity of any topologies and the resultant parameters. A description of how *PyFolding* provides the error estimates and determines how constrained parameters are is given in the error analysis section below. As with the simpler models, *PyFolding* can be used to visualise the global minimum output variables (with standard errors) and the fit of the model to the dataset (with R^2 coeff. of determination) (S.I. Jupyter notebook 5-6). The output can also be exported as a CSV file for plotting in your software of choice. In addition, *PyFolding* outputs a graphical representation of the topology used to fit the data and a graph of the denaturant dependence of each subunit used (Figure 2). Thus, *PyFolding* enables non-experts to create and analyse protein folding datasets with either a homopolymer or heteropolymer Ising model for any reasonable 1-D protein topology. Moreover, once the 1-D topology of your protein has been defined, *PyFolding* can also be used to simulate and thereby predict folding behavior of both the whole protein and the sub-units that it has been composed of (S.I. Jupyter notebook 7). In principle, this type of approach could be extended to higher dimensional topologies, thus providing a framework to enable rational protein design.

Error Analysis: We calculate various metrics to assess the quality of the output from *PyFolding*. All independent non-constant variables are reported with a standard error of each parameter, *i*:

$$SE(i) = \operatorname{cov}(i,i) \cdot \sqrt{\frac{\sum(y_{fit} - y_{obs})^2}{DOF}} \qquad (1)$$

where cov is the covariance matrix (where $\operatorname{cov}(i,i)$ represents the variance of parameter *i*), y_{fit} are the y-values of the fit at the observed x values, y_{obs} are the observed y values of the data and DOF represents the degrees of freedom (the number of data-points minus the number of free variables). From these values we can also calculate the confidence interval (nominally at 95%) where, the confidence interval for parameter *i* is :

$$CI(i) = P_i \pm t(95\%, DOF) \cdot SE(i) \qquad (2)$$

where P_i is the value of parameter *i* and $t(95\%, DOF)$ is the t-distribution at 95% with DOF degrees of freedom. Finally, we report the coefficient of determination (R^2) as a statistical measure of the error between the data and the fitted model:

$$R^2 = 1 - \frac{\sum(y_{fit} - y_{obs})^2}{\sum(y_{fit} - \overline{y_{obs}})^2} \qquad (3)$$

where $\overline{y_{obs}}$ represents the mean of the observed data.

In all models other than the heteropolymer Ising model we utilise a gradient optimiser such as the Levenberg-Marquardt algorithm that yields a covariance matrix of the fitted parameters. However, since we must utilise a different optimization method (the differential evolution optimiser) for the global fitting of heteropolymer Ising models, we calculate the errors in a slightly different way. The optimiser does not yield a covariance matrix as default, so we calculate a numerical approximation based on the Jacobian matrix (here, a matrix of numerical approximations of all the partial differentials of all variables) as follows:

$$\mathrm{cov} = (\mathbf{J}^{\mathrm{T}} \cdot \mathbf{J})^{-1} \cdot MSE \tag{4}$$

where $\mathbf{J}$ is the Jacobian matrix, and MSE is the mean squared error of the fit.

In *PyFolding* we have provided estimates of the standard error and confidence intervals for each parameter (calculated as described above) using this numerical approximation of the covariance matrix. In general, estimating errors for the parameters or the uniqueness of the solution in heteropolymer models is a complex problem, owing to the method of optimization used. Interestingly, Barrick and coworkers used Bootstrap analysis to evaluate parameter confidence intervals (12). However, many of the published studies either do not describe how error margins were determined or simply list the error between the data and curve fit. Here, when confronted with ill-posed datasets or poorly chosen topologies, which can produce an adequate curve fit to the data (as measured by R^2), *PyFolding*'s numerical error approximation becomes unstable leading to large errors. Thus, in evaluating the determinant of the Jacobian as well as the estimated errors it is possible to assess the quality of the model and the validity of the solution - large errors show that the model parameters are not properly constrained. In such cases, *PyFolding* raises the appropriate warnings to enable the user to quickly interpret the results and adjust the topologies and members of a dataset appropriately.

[Conclusion]

Here we have shown that *PyFolding,* in conjunction with Jupyter notebooks, enables researchers with minimal programming expertise the ability to fit both "typical" and complex models to their thermodynamic and kinetic protein folding data. The software is free and can be used to both analyse and simulate data with models and analyses that expensive commercial user-friendly options cannot. In particular, we have incorporated the ability to fit and simulate equilibrium unfolding experiments with user defined protein topologies, using a matrix formulation of the 1-D heteropolymer Ising model. This aspect of *PyFolding* will be of particular interest to groups working on protein folds composed of repetitive motifs such as Ankyrin repeats and TPRs, given that these proteins are increasingly being used as novel antibody therapeutics (38-41) and biomaterials (42-47). Further, as analysis can be performed in Jupyter notebooks, it enables novice researchers to easily use the software and for groups to share data and methods. We have provided a number of example notebooks and accompanying video tutorials as a resource accompanying this manuscript, enabling other users to recreate our data analysis and modify parameters. Finally, due to *PyFolding*'s extensible framework, it is straightforward to extend, thus enabling fitting and modelling of other systems or phenomena such as protein-protein and other protein-binding interactions. Such extensions can be rapidly and seamlessly deployed as a community resource thus broadening the functionality of the software.

[Author Contributions]

ARL wrote the software. ARL and ERGM developed the models. ARL, ERGM, LSI and APR tested the software and performed data analysis and simulations. ARL and ERGM created the supplementary Jupyter Notebooks. ERGM created the online tutorials. ERGM and ARL wrote the manuscript, and all authors edited and approved the manuscript.

[Acknowledgements]

We would like to thank Dr. Jonathan Phillips for insightful discussion, helpful comments and suggestions. LSI acknowledges the support of a Senior Fellowship from the UK Medical Research Foundation. AP was supported by a BBSRC Doctoral Training Programme scholarship and an Oliver Gatty Studentship. ERGM and LSI labs acknowledge support from a Leverhulme Trust project grant.

[References]

1. Main, E. R., K. F. Fulton, and S. E. Jackson. 1999. Folding pathway of FKBP12 and characterisation of the transition state. J Mol Biol 291:429-444.
2. Low, C., U. Weininger, P. Neumann, M. Klepsch, H. Lilie, M. T. Stubbs, and J. Balbach. 2008. Structural insights into an equilibrium folding intermediate of an archaeal ankyrin repeat protein. Proc Natl Acad Sci U S A 105:3779-3784.
3. Millership, C., J. J. Phillips, and E. R. G. Main. 2016. Ising Model Reprogramming of a Repeat Protein's Equilibrium Unfolding Pathway. J Mol Biol 428:1804-1817.
4. Jackson, S. E., and A. R. Fersht. 1991. Folding of chymotrypsin inhibitor 2. 1. Evidence for a two-state transition. Biochemistry 30:10428-10435.
5. Schatzle, M., and T. Kiefhaber. 2006. Shape of the free energy barriers for protein folding probed by multiple perturbation analysis. J Mol Biol 357:655-664.
6. Naganathan, A. N., and V. Munoz. 2014. Thermodynamics of downhill folding: multi-probe analysis of PDD, a protein that folds over a marginal free energy barrier. Journal of Physical Chemistry. B 118:8982-8994.
7. Ferreiro, D. U., and P. G. Wolynes. 2008. The capillarity picture and the kinetics of one-dimensional protein folding. Proc Natl Acad Sci U S A 105:9853-9854.
8. Barrick, D., D. U. Ferreiro, and E. A. Komives. 2008. Folding landscapes of ankyrin repeat proteins: experiments meet theory. Curr Opin Struct Biol 18:27-34.
9. DeVries, I., D. U. Ferreiro, I. E. Sanchez, and E. A. Komives. 2011. Folding kinetics of the cooperatively folded subdomain of the IkappaBalpha ankyrin repeat domain. J Mol Biol 408:163-176.
10. Maxwell, K. L., D. Wildes, A. Zarrine-Afsar, M. A. De Los Rios, A. G. Brown, C. T. Friel, L. Hedberg, J. C. Horng, D. Bona, E. J. Miller, A. Vallee-Belisle, E. R. Main, F. Bemporad, L. Qiu, K. Teilum, N. D. Vu, A. M. Edwards, I. Ruczinski, F. M. Poulsen, B. B. Kragelund, S. W. Michnick, F. Chiti, Y. Bai, S. J. Hagen, L. Serrano, M. Oliveberg, D. P. Raleigh, P. Wittung-Stafshede, S. E. Radford, S. E. Jackson, T. R. Sosnick, S. Marqusee, A. R. Davidson, and K. W. Plaxco. 2005. Protein folding: defining a "standard" set of experimental conditions and a preliminary kinetic data set of two-state proteins. Protein Science 14:602-616.
11. Wensley, B. G., S. Batey, F. A. Bone, Z. M. Chan, N. R. Tumelty, A. Steward, L. G. Kwa, A. Borgia, and J. Clarke. 2010. Experimental evidence for a frustrated energy landscape in a three-helix-bundle protein family. Nature 463:685-688.
12. Aksel, T., and D. Barrick. 2009. Analysis of repeat-protein folding using nearest-neighbor statistical mechanical models. Methods in Enzymology 455:95-125.
13. Mallam, A. L., and S. E. Jackson. 2007. A comparison of the folding of two knotted proteins: YbeA and YibK. J Mol Biol 366:650-665.
14. Scott, K. A., L. G. Randles, and J. Clarke. 2004. The folding of spectrin domains II: phi-value analysis of R16. J Mol Biol 344:207-221.
15. Hutton, R. D., J. Wilkinson, M. Faccin, E. M. Sivertsson, A. Pelizzola, A. R. Lowe, P. Bruscolini, and L. S. Itzhaki. 2015. Mapping the Topography of a Protein Energy Landscape. J Am Chem Soc 137:14610-14625.
16. Tsytlonok, M., P. O. Craig, E. Sivertsson, D. Serquera, S. Perrett, R. B. Best, P. G. Wolynes, and L. S. Itzhaki. 2013. Complex energy landscape of a giant repeat protein. Structure 21:1954-1965.
17. Javadi, Y., and E. R. Main. 2009. Exploring the folding energy landscape of a series of designed consensus tetratricopeptide repeat proteins. Proc Natl Acad Sci U S A 106:17383-17388.

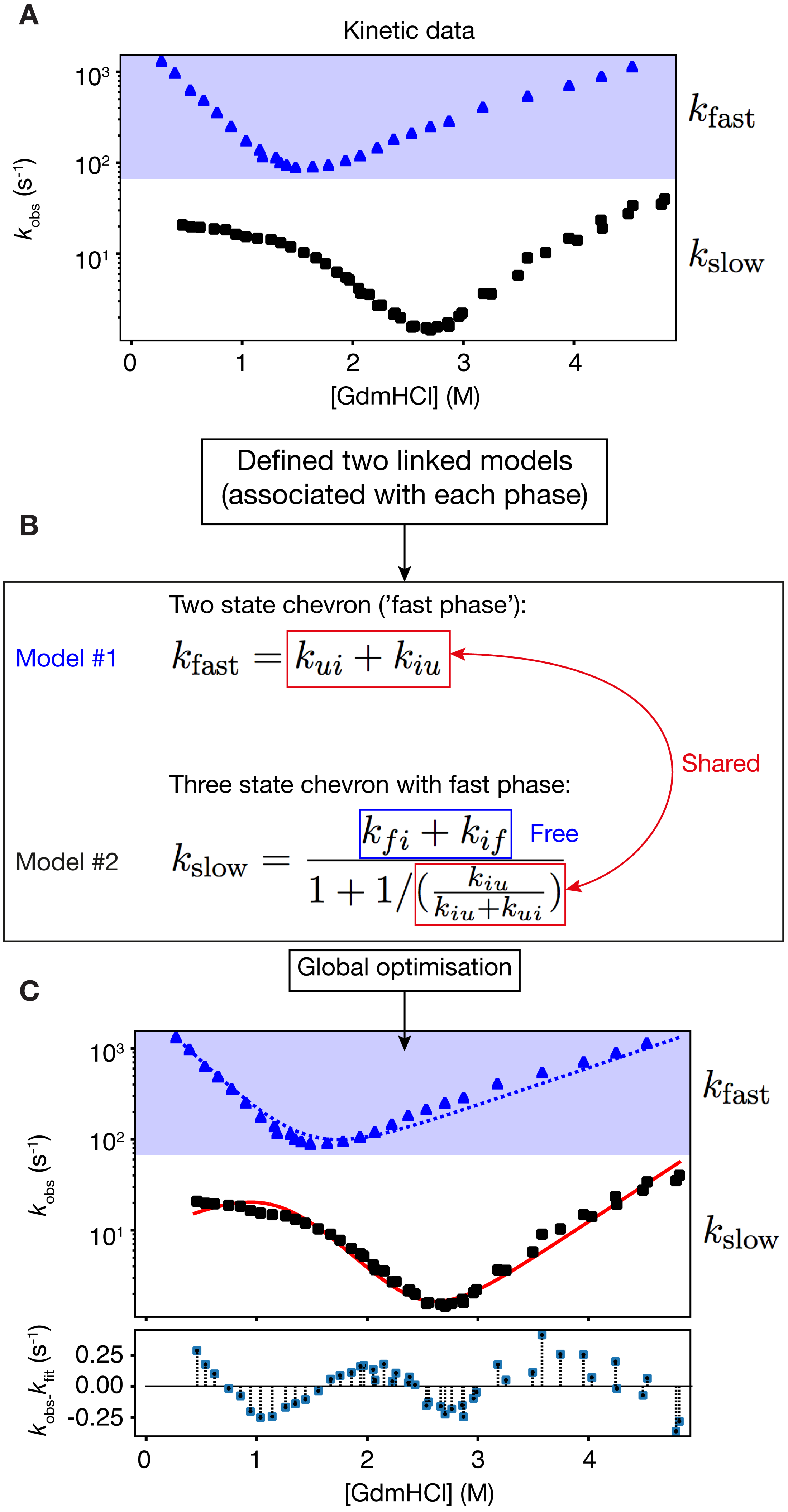
18. Lowe, A. R., and L. S. Itzhaki. 2007. Biophysical characterisation of the small ankyrin repeat protein myotrophin. J Mol Biol 365:1245-1255.
19. Xu, M., O. Beresneva, R. Rosario, and H. Roder. 2012. Microsecond folding dynamics of apomyoglobin at acidic pH. Journal of Physical Chemistry. B 116:7014-7025.
20. Garcia-Mira, M. M., M. Sadqi, N. Fischer, J. M. Sanchez-Ruiz, and V. Munoz. 2002. Experimental identification of downhill protein folding. Science 298:2191-2195.
21. Aksel, T., A. Majumdar, and D. Barrick. 2011. The contribution of entropy, enthalpy, and hydrophobic desolvation to cooperativity in repeat-protein folding. Structure 19:349-360.
22. Kajander, T., A. L. Cortajarena, E. R. Main, S. G. Mochrie, and L. Regan. 2005. A new folding paradigm for repeat proteins. Journal of the American Chemical Society 127:10188-10190.
23. Winn, M. D., C. C. Ballard, K. D. Cowtan, E. J. Dodson, P. Emsley, P. R. Evans, R. M. Keegan, E. B. Krissinel, A. G. Leslie, A. McCoy, S. J. McNicholas, G. N. Murshudov, N. S. Pannu, E. A. Potterton, H. R. Powell, R. J. Read, A. Vagin, and K. S. Wilson. 2011. Overview of the CCP4 suite and current developments. Acta Crystallogr D Biol Crystallogr 67:235-242.
24. Jones, E., T. Oliphant, P. Peterson, and others. 2001. SciPy: Open source scientific tools for Python.
25. Serrano, L., A. Matouschek, and A. R. Fersht. 1992. The folding of an enzyme. III. Structure of the transition state for unfolding of barnase analysed by a protein engineering procedure. J Mol Biol 224:805-818.
26. Fersht, A. R., A. Matouschek, and L. Serrano. 1992. The folding of an enzyme. I. Theory of protein engineering analysis of stability and pathway of protein folding. J Mol Biol 224:771-782.
27. Brush, S. G. 1967. History of the Lenz-Ising Model. Reviews of Modern Physics 39:883-893.
28. Niss, M. 2005. History of the Lenz-Ising model 1920-1950: From ferromagnetic to cooperative phenomena. Arch Hist Exact Sci 59:267-318.
29. Zimm, B. H., and J. K. Bragg. 1959. Theory of the Phase Transition between Helix and Random Coil in Polypeptide Chains. The Journal of Chemical Physics 31:526-535.
30. Munoz, V., P. A. Thompson, J. Hofrichter, and W. A. Eaton. 1997. Folding dynamics and mechanism of beta-hairpin formation. Nature 390:196-199.
31. Munoz, V., and W. A. Eaton. 1999. A simple model for calculating the kinetics of protein folding from three-dimensional structures. Proc Natl Acad Sci U S A 96:11311-11316.
32. Kubelka, J., E. R. Henry, T. Cellmer, J. Hofrichter, and W. A. Eaton. 2008. Chemical, physical, and theoretical kinetics of an ultrafast folding protein. Proc Natl Acad Sci U S A 105:18655-18662.
33. Kubelka, G. S., and J. Kubelka. 2014. Site-specific thermodynamic stability and unfolding of a de novo designed protein structural motif mapped by 13C isotopically edited IR spectroscopy. J Am Chem Soc 136:6037-6048.
34. Lai, J. K., G. S. Kubelka, and J. Kubelka. 2015. Sequence, structure, and cooperativity in folding of elementary protein structural motifs. Proc Natl Acad Sci U S A 112:9890-9895.
35. Wetzel, S. K., G. Settanni, M. Kenig, H. K. Binz, and A. Pluckthun. 2008. Folding and unfolding mechanism of highly stable full-consensus ankyrin repeat proteins. J Mol Biol 376:241-257.
36. Aksel, T., and D. Barrick. 2014. Direct observation of parallel folding pathways revealed using a symmetric repeat protein system. Biophys J 107:220-232.
37. Storn, R., and K. Price. 1997. Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11:341-359.

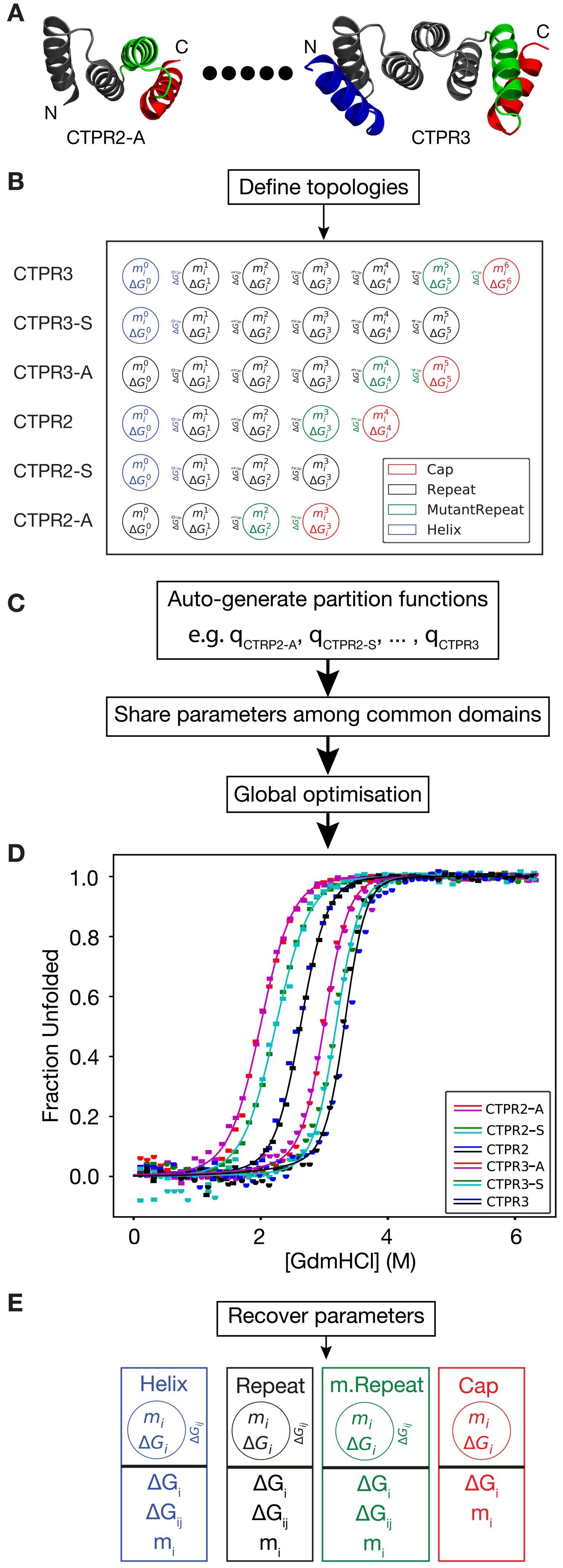
38. Rasool, M., A. Malik, M. Hussain, K. A. Haq, K. Butt, M. A. B. Ashraf, M. I. Naseer, M. Asif, R. Shaikh, M. Z. Mustafa, Q. Alam, G. Rasool, W. Ahmad, A. Haque, and M. A. Kamal. 2017. DARPins Bioengineering and its Theranostic Approaches: Emerging Trends in Protein Engineering. Curr Pharm Design 23:1610-1615.
39. Jost, C., and A. Pluckthun. 2014. Engineered proteins with desired specificity: DARPins, other alternative scaffolds and bispecific IgGs. Curr Op Struct Biol 27:102-112.
40. Ernst, P., and A. Pluckthun. 2017. Advances in the design and engineering of peptide-binding repeat proteins. Biol Chem 398:23-29.
41. Cortajarena, A. L., F. Yi, and L. Regan. 2008. Designed TPR modules as novel anticancer agents. ACS Chem Biol 3:161-166.
42. Sawyer, N., E. B. Speltz, and L. Regan. 2013. NextGen protein design. Biochem Soc Trans 41:1131-1136.
43. Main, E. R., J. J. Phillips, and C. Millership. 2013. Repeat protein engineering: creating functional nanostructures/biomaterials from modular building blocks. Biochem Soc Trans 41:1152-1158.
44. Grove, T. Z., L. Regan, and A. L. Cortajarena. 2013. Nanostructured functional films from engineered repeat proteins. Journal of the Royal Society, Interface 10:20130051.
45. Phillips, J. J., C. Millership, and E. R. G. Main. 2012. Fibrous Nanostructures from the Self-Assembly of Designed Repeat Protein Modules. Angew Chem Int Edit 51:13132-13135.
46. Grove, T. Z., and L. Regan. 2012. New materials from proteins and peptides. Curr Opin Struct Biol 22:451-456.
47. Grove, T. Z., J. Forster, G. Pimienta, E. Dufresne, and L. Regan. 2012. A modular approach to the design of protein-based smart gels. Biopolymers 97:508-517.

[Figure Legends]

Figure 1: Work flow example of the fitting linked equations in *PyFolding*. (A) Unfolding and folding kinetics (chevron plots) showing the distinct fast and slow phases for the 3-state folding thermophilic AR protein (tANK) identified in the archaeon *Thermoplasma* (2) are loaded into *PyFolding* as `Chevron` objects. **(B)** Two linked models (functions) are associated with the chevron data. These describe the fast (Model #1) and slow phases (Model #2) of the chevrons. Certain rate constants and their associated m-values, are shared between the two models. The other parameters are "free" and associated and fitted only in the slow phase model. **(C)** Global optimization within *PyFolding* enables simultaneous fitting of the two models with shared parameters to the two respective phases. The resultant fits for the fast (blue dotted line) and slow phases (red solid line) are shown overlaid on the observed data. The residuals show the difference between the slow phase observations and fit. These calculations can be found in SI Jupyter Notebook 4.

Figure 2: Work flow example of global optimization of a Heteropolymer Ising model in *PyFolding*. (A) GdmHCl-induced equilibrium denaturations of a series of single-helix deletion CTPRn proteins are loaded into *PyFolding* as `EquilibriumDenaturation` objects. In the figure we schematically represent these as individual protein structures corresponding to the smallest in the series (CTPR2-A) upto (dots) the largest (CTPR3) (3). The figures were made with Pymol and individual helices are coloured by the user defined topology used by the ising model - Helix (blue), Repeat (black), a mutant Repeat (green) or a Cap (red). **(B)** Using *PyFolding*'s built-in primitive protein folding "domains/modules", one can define topologies for each protein in the series. Each primitive is a container for several thermodynamic parameters to describe the intrinsic and interfacial stability terms. **(C)** Using the topologies defined in (B), *PyFolding* will automatically generate the appropriate partition functions (q) for each protein in the series using a matrix formulation, and share parameters between other proteins in the series. **(D)** A final global fitting step finds the optimal set of parameters to describe the series. **(E)** The optimal parameters (and their estimated errors/confidence intervals) for each domain primitive are recovered and output for the user. These calculations can be found in SI Jupyter Notebook 6.


A
Kinetic data
k_{obs} (s^{-1})
10^3
10^2
10^1
k_{fast}
k_{slow}
0
1
2
3
4
[GdmHCl] (M)
Defined two linked models (associated with each phase)
B
Two state chevron ('fast phase'):
Model #1
$k_{\mathrm{fast}} = \boxed{k_{ui} + k_{iu}}$
Shared
Three state chevron with fast phase:
Model #2
$k_{\mathrm{slow}} = \dfrac{\boxed{k_{fi} + k_{if}}}{1 + 1/\boxed{\left(\frac{k_{iu}}{k_{iu}+k_{ui}}\right)}}$
Free
Global optimisation
C
k_{obs} (s^{-1})
10^3
10^2
10^1
k_{fast}
k_{slow}
k_{obs}- k_{fit} (s^{-1})
0.25
0.00
-0.25
0
1
2
3
4
[GdmHCl] (M)


A
N
C
CTPR2-A
N
C
CTPR3
B
Define topologies
CTPR3
CTPR3-S
CTPR3-A
CTPR2
CTPR2-S
CTPR2-A
m_i^0 ΔG_i^0
ΔG_{ij}^0
m_i^1 ΔG_i^1
ΔG_{ij}^1
m_i^2 ΔG_i^2
ΔG_{ij}^2
m_i^3 ΔG_i^3
ΔG_{ij}^3
m_i^4 ΔG_i^4
ΔG_{ij}^4
m_i^5 ΔG_i^5
ΔG_{ij}^5
m_i^6 ΔG_i^6
Cap
Repeat
MutantRepeat
Helix
C
Auto-generate partition functions
e.g. $q_{CTRP2-A}$, $q_{CTPR2-S}$, ... , q_{CTPR3}
Share parameters among common domains
Global optimisation
D
Fraction Unfolded
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
[GdmHCl] (M)
CTPR2–A
CTPR2–S
CTPR2
CTPR3–A
CTPR3–S
CTPR3
E
Recover parameters
Helix
m_i ΔG_i ΔG_{ij}
ΔG_i
ΔG_{ij}
m_i
Repeat
m_i ΔG_i ΔG_{ij}
ΔG_i
ΔG_{ij}
m_i
m.Repeat
m_i ΔG_i ΔG_{ij}
ΔG_i
ΔG_{ij}
m_i
Cap
m_i ΔG_i
ΔG_i
m_i