eSPC, an Online Data Analysis Platform for Molecular Biophysics

ThermoAffinity 1.0 User Documentation

October 2022

Table of Contents

Overview

ThermoAffinity has seven panels (Figure 1). Panels 1-2 contain the necessary steps to analyze user data. The Simulate data Panel can be used before doing an experiment to analyze the expected change in the signal depending on the binding affinity and the protein and ligand concentrations.

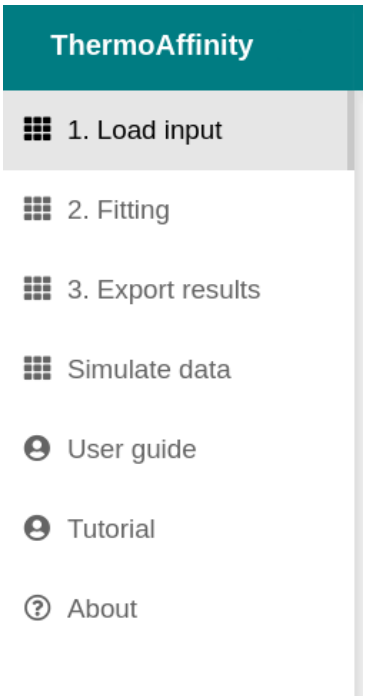
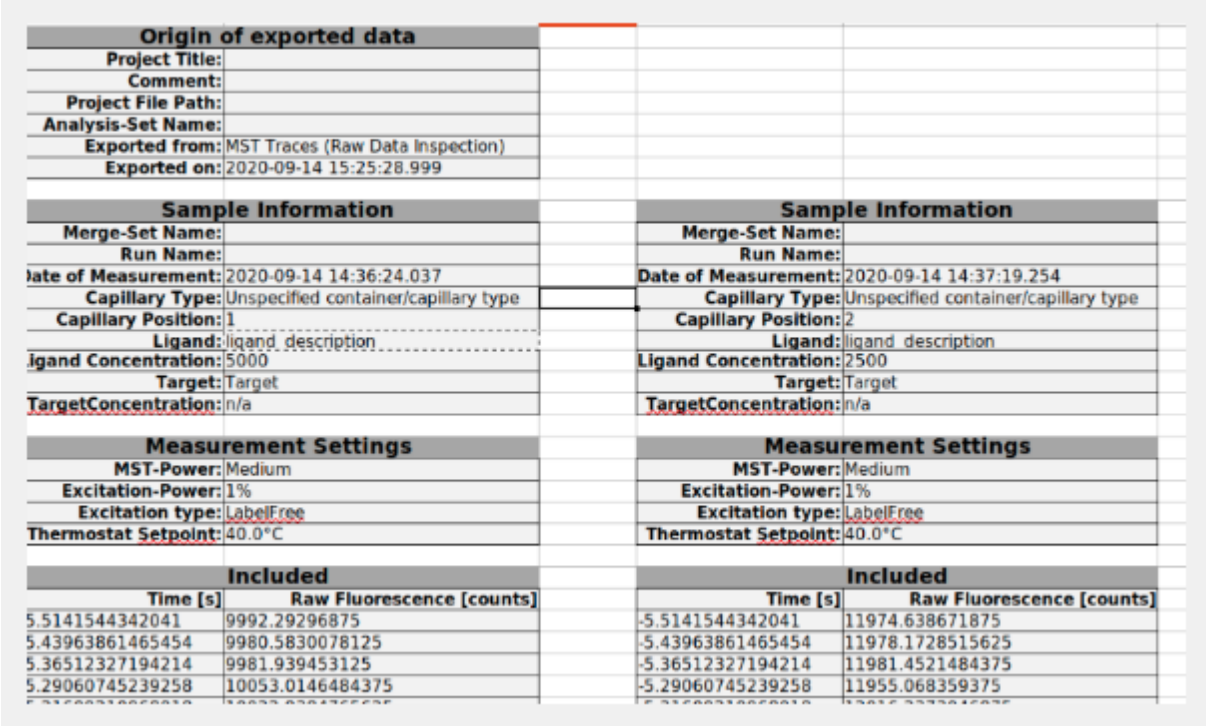


Figure 1. ThermoAffinity online tool panels.

1. Load input

1.1. Input file (raw data)

ThermoAffinity accepts as input the spreadsheet generated by NanoTemper Technologies. This file contains one sheet called ‘RawData’ where the first column has different cells with the following labels: ‘Capillary Position:’, ‘Ligand:’, ‘Ligand Concentration:’, and ‘Time [s]’. Then, the second column stores the associated information: ‘1’, ‘ligand description’, ‘5000’, and ‘Raw Fluorescence [counts]’ (Figure 2). The next capillary information is going to be read from columns 4 and 5, then from columns 7 and 8, etc.



Origin of exported data	
Project Title:	
Comment:	
Project File Path:	
Analysis-Set Name:	
Exported from:	MST Traces (Raw Data Inspection)
Exported on:	2020-09-14 15:25:28.999

Sample Information	
Merge-Set Name:	
Run Name:	
ate of Measurement:	2020-09-14 14:36:24.037
Capillary Type:	Unspecified container/capillary type
Capillary Position:	1
Ligand:	ligand_description
igand Concentration:	5000
Target:	Target
TargetConcentration:	n/a

Measurement Settings	
MST-Power:	Medium
Excitation-Power:	1%
Excitation type:	LabelFree
Thermostat Setpoint:	40.0°C

Included	
Time [s]	Raw Fluorescence [counts]
5.5141544342041	9992.29296875
5.43963861465454	9980.5830078125
5.36512327194214	9981.939453125
5.29060745239258	10053.0146484375

Sample Information	
Merge-Set Name:	
Run Name:	
Date of Measurement:	2020-09-14 14:37:19.254
Capillary Type:	Unspecified container/capillary type
Capillary Position:	2
Ligand:	ligand_description
Ligand Concentration:	2500
Target:	Target
TargetConcentration:	n/a

Measurement Settings	
MST-Power:	Medium
Excitation-Power:	1%
Excitation type:	LabelFree
Thermostat Setpoint:	40.0°C

Included	
Time [s]	Raw Fluorescence [counts]
-5.5141544342041	11974.638671875
-5.43963861465454	11978.1728515625
-5.36512327194214	11981.4521484375
-5.29060745239258	11955.068359375

Figure 2. Example of the spreadsheet required to load the MST experiment result into ThermoAffinity.

ThermoAffinity can also load a file with or without a header and two columns separated by spaces, commas, or semicolons. The first column has the ligand concentration and the second the signal value. This file can have two additional columns: protein concentration, and/or experiment ID.

1.2. Normalization

The signal of each curve is divided by the mean value of the signal before the T-jump ($time <= 0$).

1.3. Median filter (smoothing)

The median filter calculates the median value of a temperature rolling window.

1.4. Hot and cold region selection

The thermophoretic signal of the hot (F_{Hot}) and cold (F_{Cold}) regions are averaged to get the $Fnorm$values ($\frac{F_{Hot}}{F_{Cold}}$).

2. Fitting

2.1 Model

The models implemented in ThermoAffinity are useful for all cases where a signal can be described by a linear combination of the unbound protein and complex. This can be $Fnorm = F_{hot} / F_{cold}$ (thermophoresis shift) or the initial fluorescence.

The signal is fitted using a simple model where the contribution of the complex and the unbound protein is given by the following equation:

$$Signal(K_d, L_0, P_0) = RF1 * P(K_d, L_0, P_0) + RF2 * PL(K_d, L_0, P_0) \quad (1)$$

where P and PL are respectively the unbound free protein and the bound protein. P_0 and L_0 are respectively the total protein and ligand concentration and K_d is the equilibrium dissociation constant linked to the chemical equilibrium

$$P + L \leftrightarrow PL,\ K_d = (P * L) / PL \quad (2)$$

and

$RF1$ and $RF2$ are parameters that represent the signal per unit of concentration.

Using Equation 46 and the fact that the total ligand and protein concentrations are constant, we can transform the signal to:

$$Signal(K_d, L_0, P_0) = 0.5 * ((K_d + P_0 + L_0) - \sqrt{(K_d + P_0 + L_0)^2 - 4 * P_0 L_0})) * (RF2 - RF1) + RF1 * P_0 \quad (3)$$

Two binding sites

In the case of the binding sites, the signal can be explained by a linear combination of the amount of free unbound protein, right-bound complex (PL), left-bound complex (LP), and double-bound complex (LPL).

$$Signal(L_0, P_0, K_{d,1}, K_{d,2}, cFactor) = RF1 * P + RF2 * PL + RF3 * LP + RF4 * LPL \quad (4)$$

where $RF1$, $RF2$, $RF3$and $RF4$are parameters to fit that represent the signal per units of concentration, L_0, P_0, $K_{d,1}$, $K_{d,2}$and $cFactor$ are respectively the total protein concentration, total ligand concentration, the equilibrium dissociation constant 1, the

equilibrium dissociation constant 2, and cooperativity factor. The associated chemical equilibria are

$$P + 2L \leftrightarrow PL + L \quad (5)$$
$$P + 2L \leftrightarrow LP + L \quad (6)$$
$$PL + L \leftrightarrow LPL \quad (7)$$
$$LP + L \leftrightarrow LPL \quad (8)$$

with equations

$$P = K_{d,1} * K_{d,2} * (L_0 - L) / ((K_{d,1} + K_{d,2} + 2 * L) * L) \quad (9)$$
$$PL = (L * P / K_{d,2}) \quad (10)$$
$$LP = (L * P / K_{d,1}) \quad (11)$$
$$LPL = \frac{LP*L}{K_{d,2} * cFactor} \quad (12)$$

where L is the free ligand concentration that corresponds to the the only physical root of the equation

$$X^3 + pX^2 + qX + r \quad (13)$$

$$p = [K_{d,1} + K_{d,2} + (2 * P_0 - L_0) / cFactor] * cFactor \quad (14)$$
$$q = [(P_0 - L_0) * (K_{d,1} + K_{d,2}) + K_{d,1} * K_{d,2}] * cFactor \quad (15)$$
$$r = [- L_0 * K_{d,1} * K_{d,2}] * cFactor \quad (16)$$

Due to the number of parameters, we have simplified this model to some alternatives.

For parameters $RF1$, $RF2$, $RF3$and $RF4$, we have

a) $RF2 = RF3 = RF1 + \Delta F$ & $RF4 = RF1 + 2\Delta F$
b) $RF1 = RF2$& $RF4 = RF3 = RF1 + \Delta F$

For $K_{d,1}$, $K_{d,2}$and $cFactor$,

a) $K_{d,1}$=$K_{d,2}$& $cFactor = 1$
b) $K_{d,1}$=$K_{d,2}$& $cFactor \neq 1$
c) $K_{d,1}$≠$K_{d,2}$& $cFactor = 1$

2.2 Initial estimates and boundaries of the parameters

To improve the convergence of the fitting procedure, initial estimates and boundaries are estimated as follows.

Parameter	Initial value
RF2	$\frac{min(signal)}{P_0}$ if $maxLigSignal <= minLigSignal$ else $\frac{max(signal)}{P_0}$
RF1	$\frac{max(signal)}{P_0}$ if $maxLigSignal <= minLigSignal$ else $\frac{min(signal)}{P_0}$
K_d,$K_{d,1}$,$K_{d,2}$	$median(\ LigConcVec\ )$

*$maxLigSignal$ and $minLigSignal$ are respectively the signal of the position with the highest and lowest ligand (binding partner) concentration. $LigConcVec$is the vector containing the ligand concentrations.

Parameter	Lower bound	Upper bound
RF2, RF1	$min(RF1_{Init}, RF2_{Init}) * 0.7\ if\ min(RF1_{Init}, RF2_{Init}) > 0$else $min(RF1_{Init}, RF2_{Init}) * 1.4$	$max(RF1_{Init}, RF2_{Init}) * 1.4\ if\ max(RF1_{Init}, RF2_{Init}) > 0$else $max(RF1_{Init}, RF2_{Init}) * 0.7$
Kd	$min(LigConcVec) * 1.5$	$max(LigConcVec) / 1.5$
$K_{d.1}$,$K_{d,2}$	$min(LigConcVec) * 3$	$max(LigConcVec) / 3$

2.3 Curve fitting

The $Fnorm$(or initial fluorescence) versus ligand concentration is fitted using the Levenberg Marquardt algorithm. In all cases, the units of RF1 and RF2 are [1 / µM].

2.4 Fitting errors

The standard deviation of all fitted parameters is computed using the square root of diagonal values from the fit parameter covariance matrix (using the R programming language package *minpack.lm*).

This error is then used to obtain symmetric 95% t-based confidence intervals (Asymptotic). When fitting the '1:1' or the '1:2' (one Kd) binding models, we also provide the marginal asymmetric confidence interval. It has been shown that this

approach is more robust in estimating uncertainties, so we recommend reporting this result.[1]

Briefly, the lower and upper bounds of the 95 % confidence interval are given by the values of K_d satisfying

$$RSS(K_d) = RSS_0(1 + \frac{criticalValue}{n-p}) \qquad (17)$$

where RSS_0 is the residual sum of squares using the best estimates for all the parameters, $RSS(K_d)$ is the residual sum of squares using a fixed value of K_d (fitting again the other parameters), *n* is the number of data points, *p* is the number of parameters, and *critcialValue* is the critical value of the Fisher-Snedecor distribution with *n* - *p* and 1 degress of freedom and a confidence level of 95 %.

Contact details

For further assistance, please contact us:

📧 spc@embl-hamburg.de
📌EMBL (c/o DESY), Notkestrasse 85, Build. 25a, 22607 Hamburg, Germany

[1] Paketurytė, Vaida, et al. "Uncertainty in protein–ligand binding constants: asymmetric confidence intervals versus standard errors." *European Biophysics Journal* 50.3 (2021): 661-670.

Packages

ThermoAffinity is possible thanks to:

R language: R Core Team (2020). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

R package shiny: Winston Chang, Joe Cheng, JJ Allaire, Yihui Xie and Jonathan McPherson (2020). shiny: Web Application Framework for R. R package version 1.4.0.2. https://CRAN.R-project.org/package=shiny

R package viridis: Simon Garnier (2018). viridis: Default Color Maps from 'matplotlib'. R package version 0.5.1. https://CRAN.R-project.org/package=viridis

R package tidyverse: Wickham et al., (2019). Welcome to the tidyverse. Journal of Open Source Software, 4(43), 1686, https://doi.org/10.21105/joss.01686

R package pracma: Hans W. Borchers (2019). pracma: Practical Numerical Math Functions. R package version 2.2.9. https://CRAN.R-project.org/package=pracma

R package shinydashboard: Winston Chang and Barbara Borges Ribeiro (2018). shinydashboard: Create Dashboards with 'Shiny'. R package version 0.7.1. https://CRAN.R-project.org/package=shinydashboard

R package ggplot2: H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.

R package xlsx: Adrian Dragulescu and Cole Arendt (2020). xlsx: Read, Write, Format Excel 2007 and Excel 97/2000/XP/2003 Files. R package version 0.6.3. https://CRAN.R-project.org/package=xlsx

R package reshape2: Hadley Wickham (2007). Reshaping Data with the reshape Package. Journal of Statistical Software, 21(12), 1-20. URL http://www.jstatsoft.org/v21/i12/.

R package tippy: John Coene (2018). tippy: Add Tooltips to 'R markdown' Documents or 'Shiny' Apps. R package version 0.0.1. https://CRAN.R-project.org/package=tippy

R package shinyalert: Pretty Popup Messages (Modals) in 'Shiny'. R package version 1.1. https://CRAN.R-project.org/package=shinyalert

R package plotly: C. Sievert. Interactive Web-Based Data Visualization with R, plotly, and shiny. Chapman and Hall/CRC Florida, 2020.

R package tableHTML: Theo Boutaris, Clemens Zauchner and Dana Jomar (2019). tableHTML: A Tool to Create HTML Tables. R package version 2.0.0. https://CRAN.R-project.org/package=tableHTML

R package rhandsontable: Jonathan Owen (2018). rhandsontable: Interface to the 'Handsontable.js' Library. R package version 0.3.7. https://CRAN.R-project.org/package=rhandsontable

R package remotes: Jim Hester, Gábor Csárdi, Hadley Wickham, Winston Chang, Martin Morgan and Dan Tenenbaum (2020). remotes: R Package Installation from Remote Repositories, Including 'GitHub'. R package version 2.1.1. https://CRAN.R-project.org/package=remotes

R package devtools: Hadley Wickham, Jim Hester and Winston Chang (2020). devtools: Tools to Make Developing R Packages Easier. R package version 2.3.0. https://CRAN.R-project.org/package=devtools

R package shinyjs: Dean Attali (2020). shinyjs: Easily Improve the User Experience of Your Shiny Apps in Seconds. R package version 1.1. https://CRAN.R-project.org/package=shinyjs

R package data.table: Matt Dowle and Arun Srinivasan (2019). data.table: Extension of data.frame. R package version 1.12.8. https://CRAN.R-project.org/package=data.table

R package reticulate: Kevin Ushey, JJ Allaire and Yuan Tang (2020). reticulate: Interface to 'Python'. R package version 1.16. https://CRAN.R-project.org/package=reticulate

R package shinycssloaders: Andras Sali and Dean Attali (2020). shinycssloaders: Add CSS Loading Animations to 'shiny' Outputs. R package version 0.3. https://CRAN.R-project.org/package=shinycssloaders

R package nlstools: Florent Baty, Christian Ritz, Sandrine Charles, Martin Brutsche, Jean-Pierre Flandrois, Marie-Laure Delignette-Muller (2015). A Toolbox for Nonlinear Regression in R: The Package nlstools. Journal of Statistical Software, 66(5), 1-21. URL http://www.jstatsoft.org/v66/i05/

R package minpack.lm: Timur V. Elzhov, Katharine M. Mullen, Andrej-Nikolai Spiess and Ben Bolker (2016). minpack.lm: R Interface to the Levenberg-Marquardt

Nonlinear Least-Squares Algorithm Found in MINPACK, Plus Support for Bounds. R package version 1.2-1. https://CRAN.R-project.org/package=minpack.lm

R package broom: David Robinson, Alex Hayes and Simon Couch (2020). broom: Convert Statistical Objects into Tidy Tibbles. R package version 0.7.1. https://CRAN.R-project.org/package=broom

R pacakge data.table: Matt Dowle and Arun Srinivasan (2021). data.table: Extension of `data.frame`. R package version 1.14.2. https://CRAN.R-project.org/package=data.table

Python3.7 language: Van Rossum, G., & Drake, F. L. (2009). Python 3 Reference Manual. Scotts Valley, CA: CreateSpace.

Python package numpy: Travis E, Oliphant. A guide to NumPy, USA: Trelgol Publishing, (2006). Stéfan van der Walt, S. Chris Colbert, and Gaël Varoquaux. The NumPy Array: A Structure for Efficient Numerical Computation, Computing in Science & Engineering, 13, 22-30 (2011), DOI:10.1109/MCSE.2011.37

Python package pandas: Wes McKinney. Data Structures for Statistical Computing in Python, Proceedings of the 9th Python in Science Conference, 51-56 (2010)

Python package scipy: Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E.A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. (2020) SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods, 17(3), 261-272.

Python package xlrd: https://xlrd.readthedocs.io/en/latest/index.html

Python package natsort: https://natsort.readthedocs.io/en/master/