eSPC, an Online Data Analysis Platform for Molecular Biophysics

FoldAffinity 1.0 User Documentation

June 2026

Table of Contents

1. Load input
    - 1.1. Input file (raw data)
    - 1.2. Median filter (smoothing)
    - 1.3. Melting temperature (T_m) estimation using the first derivative
2. Fit fluorescence
    - 2.1. Model
    - 2.2. Curve fitting
    - 2.3. Fitting errors
3. Fit unfolded fraction
    - 3.1. Models
    - 3.2. Curve fitting
    - 3.3. Fitting errors
4. Alternative T_m fitting: Binding affinity from the observed melting temperatures
    - 4.1 Model
    - 4.2. Curve fitting
    - 4.3. Fitting errors

Contact details

Overview

FoldAffinity has 9 panels (Figure 1). Panels 1-3 contain the necessary steps to analyze user data using the isothermal approach. The Panel Tm fitting can be used to estimate binding affinities directly from the observed melting temperatures. 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.

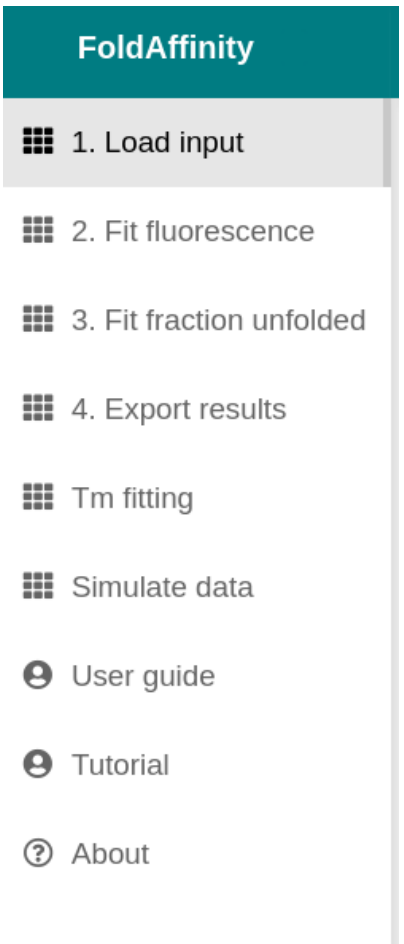

Figure 1. FoldAffinity online tool panels.

1. Load input

1.1. Input file (raw data)

FoldAffinity accepts as input several kind of files:

A) The xlsx file (processed) generated by the **Nanotemper Prometheus** machine that has one sheet called 'Overview' with a column called 'Sample ID' with the names of the samples (ligand concentration) (Figure 2), and four

sheets called 'Ratio', '330nm', '350nm' and 'Scattering'. The first column of the signal sheet ('Ratio', '330nm', '350nm', 'Scattering') should be called 'Time [s]'. The second column should have the temperature data and all subsequent columns store the fluorescence data (Figure 3). The order of the fluorescence columns should match the order of the 'Sample ID' column in the 'Overview' sheet.

B
Sample ID
A1 GuHCl 0.05 M
A2 GuHCl 0.52 M
A3 GuHCl 1 M
A4 GuHCl 1.47 M
A5 GuHCl 1.95 M
A6 GuHCl 2.38 M
A7 GuHCl 2.5 M
A8 GuHCl 2.61 M
A9 GuHCl 2.73 M
A10 GuHCl 2.85 M
A11 GuHCl 2.97 M
A12 GuHCl 3.09 M
B1 GuHCl 3.21 M
B2 GuHCl 3.33 M
B3 GuHCl 3.45 M
B4 GuHCl 3.56 M
B5 GuHCl 3.68 M
B6 GuHCl 4.25 M
B7 GuHCl 4.82 M
B8 GuHCl 5.39 M

Figure 2. Example of the 'SampleID' column in the 'Overview' sheet required by FoldAffinity to load the Nanotemper spreadsheet input file.

	A	B	C	D
1		Capillary	1	2
2		Sample ID	A1	
3	Time [s]	Temperature [°C]	Fluorescence [counts]	Fluorescence [counts]
4	7.0	25.000	0.940	0.934
5	24.3	25.054	0.939	0.935
6	33.3	25.108	0.943	0.933
7	40.6	25.162	0.939	0.936
8	46.8	25.215	0.942	0.933
9	52.5	25.269	0.941	0.933
10	57.4	25.323	0.942	0.934
11	62.1	25.377	0.941	0.934
12	66.7	25.431	0.942	0.934
13	71.0	25.485	0.940	0.933

Figure 3. Example of the 'Ratio' sheet required by FoldAffinity to load the Nanotemper spreadsheet input file.

B) The xls file generated by the **ThermoFluor Assay in a qPCR machine**. One sheet called 'RFU' where the first row has the sample positions (header), the

first column has the temperature data and all subsequent columns store the fluorescence data (Figure 4).

A	B	C	D	E	F	G	H
	A01	A02	A03	A04	A05	A06	A07
5	64.79	501.82	398.53	61.91	73.26	129.38	38.53
6	63.14	513.32	416.32	63.13	72.41	130.21	40.43
7	61.52	522.98	437.17	64.34	72.21	131.14	42.45
8	59.75	529.89	459.98	64.97	72.52	131.29	42.69
9	57.78	535.95	483.14	65.89	72.30	131.90	43.18
10	55.73	540.72	504.85	67.40	71.83	131.75	42.82
11	54.00	545.15	527.02	68.86	71.27	131.86	42.98
12	52.82	549.80	549.27	70.20	71.55	131.55	42.65
13	52.14	554.45	570.59	70.37	72.86	131.79	42.15
14	51.42	558.53	589.62	70.41	74.39	132.50	41.38

Figure 4. Example of the ‘RFU’ sheet required by FoldAffinity to load ThermoFluor data.

C) The xlsx file generated by the **Prometheus Panta** instrument. This file has one sheet called ‘Overview’ with a column called 'Sample ID' with the names of the samples (Figure 2), and one sheet called ‘Data Export’ where all the data is stored (Figure 5). The ‘Data Export’ sheet columns should have the following order:

Temperature capillary 1 ; Ratio capillary 1 ; … ; Temperature capillary 1 ; 350 nm capillary 1 ; … ; Temperature capillary 1 ; 330 nm capillary 1 ; … ; Temperature capillary 1 ; scattering capillary 1 ; … ; Temperature capillary 2 ; Ratio capillary 2; … ; Temperature capillary *n* ; Ratio capillary *n*.

Columns whose names include "Derivative" are not read.

Temperature for Cap.1 (°C)	Ratio 350 nm / 330 nm for Cap.1
24.9993991851807	0.668331980705261
25.0000820159912	0.668272197246552
25.0013084411621	0.668575644493103
25.001501083374	0.66842645406723
25.0040893554687	0.667966544628143
25.0060691833496	0.668425559997559

Figure 5. Example of the ‘Data Export’ sheet required by FoldAffinity to load the Prometheus Panta spreadsheet input file.

D) The text file (.txt) generated by the **QuantStudio™ 3 System instrument** (Figure 6). Comments should be at the beginning of the file and start with ‘*’. The header is the first line with more than 6 words, i.e. ‘Well’, ‘Well Position’, … , ‘Target Name’. The column number 2 has the sample IDs. The columns number 4 and 5 have the signal and temperature data.

```
* Post-read Stage/Step =
* Pre-read Stage/Step =
* Quantification Cycle Method = Ct
* Signal Smoothing On = true
* Stage where Melt Analysis is performed = Stage2
* Stage/ Cycle where Ct Analysis is performed = Stage0, Step0
* User Name = user

[Melt Curve Raw Data]
Well    Well Position   Reading Temperature     Fluorescence    Derivative      Target Name
10      A10     1       24.913  4,828.486       -1,144.751      Target 1
10      A10     2       25.028  4,951.091       -905.159        Target 1
```

Figure 6. Example of the input file required by FoldAffinity to load the QuantStudio™ 3 System instrument data.

E) The xlsx file generated by the **Nanotemper Prometheus Tycho** instrument. This file has one sheet called 'Results' (with 6 columns named '#', 'Capillary label',..., 'Sample Brightness') (Figure 7), and one sheet called 'Profiles_raw' where the fluorescence data is stored.

#	Capillary label	Ti#1	Ti#2	Ti#3	Initial Ratio	Δ Ratio	Sample Brightness
1	Sample1	54.2			0.6700	0.2637	763.4
2	Sample2	54.7			0.5777	0.1825	688.3
3	Sample3	89.7			0.6071	0.0950	382.1
4	Sample4				0.6230	0.0480	73.6
5	Sample5						
6	Sample6						

Figure 7. Example of the 'Results' sheet required by FoldAffinity to retrieve the number of samples. This example file was generated by the Nanotemper Prometheus Tycho instrument.

The 'Profiles_raw' sheet columns should have the following structure (Figure 8):

One row with information about the recorded signal, e.g., 'Ratio 350 nm / 330 nm', 'Brightness @ 330 nm', 'Brightness @ 350 nm'.

One row with the capillary numbers.

One row with the time, temperature and sample names.

The remaining rows store the data.

	Signal:	Ratio 350 nm / 330 nm			
	Capillary:	1	2	3	4
Time [s]	Temperature [°C]	Sample1	Sample2	Sample3	Sample4
81.5	35.1	0.6699	0.5774	0.6065	0.6233
81.7	35.2	0.6701	0.5781	0.6063	0.6160
81.9	35.3	0.6705	0.5774	0.6079	0.6229
82.1	35.4	0.6704	0.5772	0.6056	0.6272

Figure 8. Example of the 'Profiles_raw' sheet required by FoldAffinity to load the temperature and signal data. This example file was generated by the Nanotemper Prometheus Tycho instrument.

F) The text file (.txt) generated by Agilent's **MX3005P qPCR instrument**. The data format is the following:

```
Line 1:      Header
Line 2:      Segment  2 Plateau  1 Well  1
Line 3:      ROX
Line 4:      1        1706      25.0
Line 5:      2        2581      25.8
Line n:      70          4845       93.7
Line n+1:    Segment  2 Plateau  1 Well  2
Line n+2:    ROX
Line n+3:    1           1707       25.0
```

The data of each well is separated by rows containing the sentence 'Segment No Plateau No Well No'. The rows after the line with 'ROX' contain the fluorescence and temperature data (second and third columns respectively).

G) The JSON file (.supr) exported by the SUPR-DSF instrument software from AppliedPhotophysics (https://www.photophysics.com/product-pages/supr-dsf/).

The JSON file should have the following structure:

- An item called 'Samples' that contains:
    - A sub-item called 'SampleName'
    - A sub-item called 'WellLocations'
- An item called 'Wells' that contains:
    - A sub-item called '_scans'
    - A sub-item called 'PhysicalLocation'
- An item called 'Wavelengths'

Additionally, within the '_scans' sub-item, there are further sub-items called:

- 'Temperature'
- 'Signal'

Below you'll find a minimal example.


```
{
  "Samples": [
    {
      "SampleName": "SampleA1",
      "WellLocations": "A1"
    }
  ],
  "Wells": [
    {
      "_scans": [
        {
          "Temperature": 20,
          "Signal": [
            7993,
            9579,
            10742
          ]
        },
        {
          "Temperature": 30,
          "Signal": [
            8993,
            10579,
            11742
          ]
        }
      ],
      "PhysicalLocation": "A1"
    }
  ],
  "Wavelengths": [
    310,
    311,
    312
  ]
}
```


After loading the file, the temperature data will be interpolated using steps of 0.5 degrees. Additionally, if there is wavelength data near 330 nm and 350 nm, the 'Ratio 350nm / 330nm' will be automatically calculated and available for further analysis.

H) A csv file with the temperature data in the first column and the signal data in the subsequent columns. The condition labels are read from the header.

	A	B	C	D
1	Temperature	Condition 1	Condition 2	Condition 3
2	10	20	20	20
3	11	30	30	30
4	12	40	40	40
5	13	50	50	50
6	14	60	60	60
7	15	70	70	70
8	16	80	80	80
9	17	90	90	90
10	18	100	100	100
11	19	110	110	110
12	20	120	120	120
13	21	130	130	130
14	22	140	140	140

Figure 9. Example of a csv file that can be imported into FoldAffinity.

I) The spreadsheet file (.xlsx) exported by the UNCLE instrument. In this file, there is one sheet per measured condition. The name of the condition is extracted from the cell located in the first row and fifth column. The second row has the temperature data in the format 'Temp: xx, Time: xx'. The signal matrix starts at the fifth row, second column. The wavelength data is in the first column.

	A	B	C	D	
1				Sample Name	1
2		Temp :25, Time:134.9	Temp :25.49, Time:18	Temp :25.99, Time:23	Te
3	Wavelength	Intensity	Intensity	Intensity	In
4					
5	311.911193847656	1157.984	1136.362	1118.812	
6	312.387084960938	1194.752	1185.197	1162.002	
7	312.862884521484	1244.214	1232.758	1214.172	
8	313.338684082031	1299.843	1286.473	1264.072	
9	313.814392089844	1358.016	1330.473	1312.704	
10	314.290100097656	1410.235	1378.81	1368.539	
11	314.765716552734	1483.058	1447.716	1421.273	
12	315.241333007813	1525.251	1492.625	1492.752	

Figure 10. Example of one sheet from the UNCLE xlsx file.

J) The spreadsheet file (.xlsx) exported by the AUNTY instrument. In this file, there is one sheet per measured condition. The name of the condition is extracted from the sheet name. The first column has the temperature data, while the second row contains the wavelength data. The signal matrix starts at the third row, second column.

	A	B	C	D
1		wavelength		
2	temperature	250	250.49	250.98
3	14.99	100.00	100.00	100.00
4	15.48	100.00	100.00	100.00
5	16.03	100.00	100.00	100.00
6	16.54	100.00	100.00	100.00
7	17.05	100.00	100.00	100.00
8	17.55	100.00	100.00	100.00

Figure 11. Example of one sheet from the AUNTY xlsx file.

K) The .txt file exported by the Bio-RAD CFX Opus. Internally, it is actually a .tsv file, in which the first column corresponds to the temperatures of the measurements, and the rest of the columns correspond to the samples. It must be noted that because all rows begin with a tab space, when the file is loaded as a .tsv file, it may show an empty column at the beginning.

	A	B	C	D	E
1		Temperature	A1	A2	A3
2		25	2200.12608043939	2203.66658249555	2200.7049554228
3		25.5	2199.34705137452	2202.32984188707	2200.37788344663
4		26	2198.56802230964	2200.99310127859	2200.05081147046
5		26.5	2197.78899324477	2199.65636067011	2199.72373949429
6		27	2197.00996417989	2198.31962006163	2199.39666751812

Figure 12. Example of a .txt file produced by the Bio-RAD instrument.

1.2. Median filter (smoothing)

The median filter consists of calculating the median value of a temperature rolling window.

1.3. Melting temperature (T_m) estimation using the first derivative

A non-model approach to estimate the melting temperature involves estimating the maximum or the minimum of the first derivative, depending on the way the signal changes with the temperature. In FoldAffinity, the Tm values are estimated as follows. First, the median value of the first derivative in the interval $[min(temperature) + 6; min(temperature) + 11]$and $[max(temperature) - 11; max(temperature) - 6]$ is calculated. Then, we obtain the mean of those two median values and add it (if it positive), or subtract it (if it is negative), to the first derivative in the interval $[min(temperature) + 1; max(temperature) - 1]$. This is done to shift the derivative

baseline. Last, if the absolute value of the minimum (of the derivative) is higher than the absolute value of the maximum, we use the minimum to estimate the Tm. Otherwise, we use the maximum. If many curves are present, we always use the same option.

To compute the derivative we use the Savitzky-Golay function as implemented in numpy (scipy.signal.savgol_filter — SciPy v1.6.1 Reference Guide) with a polynomial degree 4 and window size of 10 degrees.

2. Fit fluorescence

2.1. Model

Each fluorescence versus temperature curve is fitted with a two-state folding model where the signal is the sum of the fluorescence of the folded and unfolded states.

$$F(T) = F_{obs}(IF + T * SF) + U(IU + T * SU) \quad (1)$$

where F_{obs} and U are respectively the observed folded and unfolded fractions, $IF$and $IU$are the intercept of the folded and unfolded fractions, $IU$and SU are the slope of the folded and unfolded fractions, and

$$F_{obs}(K_{u,obs}) = 1 / (1 + K_{u,obs}) \quad (2)$$

$$U(K_{u,obs}) = K_{u,obs} / (1 + K_{u,obs}) \quad (3)$$

with

$$K_{u,obs}(T) = e^{(-\Delta G_{obs} / RT)} = \frac{U}{F+FL} \quad (4)$$

$$\Delta G_{obs}(T) = \Delta H_{obs} * (1 - \frac{T}{T_{m,obs}+273.15}) + \\ - C_p * (T_{m,obs} + 273.15 - T + T * log(\frac{T}{T_{m,obs}+273.15})) \quad (5)$$

where R is the gas constant,$\Delta G_{obs}(T)$is the free energy of unfolding, $U$is the equilibrium concentration of the unfolded species, F is the equilibrium concentration of the folded unbound species, $FL$is the equilibrium concentration of the folded bound species, $T_{m,obs}$ is the observed melting temperature, $\Delta H_{obs}$is the enthalpy of unfolding, and $C_p$is the heat capacity at a constant temperature.

Equation 18 is thermodynamically correct only when there is no ligand involved ($K_{u,obs} = K_u = U / F$). When there is ligand present, $Ku_{obs} = U / (F + FL)$. In spite of this, Bai *et al.* and Niebling *et al.* have proven that this equation allows a correct estimation of the equilibrium dissociation constant.[1,2]

2.2. Curve fitting

Once the data is loaded in FoldAffinity, the first and last 10 degrees of each fluorescence melting curve is fitted using the equation of a line to obtain initial values of $InterceptFolded$,$SlopeFolded$,$InterceptUnfolded$ and $SlopeUnfolded$parameters. Then, each curve is fitted individually using the Levenberg Marquardt (damped least-squares) algorithm to estimate ΔH_{obs}, $T_{m,obs}$and C_p. These values can be used directly or the whole data can be fitted again to force shared values of the slope parameters and / or C_p value.

2.3. Fitting errors

The standard deviation of all fitted parameters is computed using the square root of diagonal values from the fit parameter covariance matrix reported by scipy.curve_fit function. These values are an approximation (underestimation) of the real errors.

3. Fit unfolded fraction

3.1. Models

The models implemented in FoldAffinity are based on the coupling between ligand binding and protein folding and require that the ligand is completely soluble at all the measured concentrations and temperatures.

One binding site

At a fixed temperature, if we assume that the ligand can only bound the folded state, a one binding site system can be described with the following reactions.

$$U + L \leftrightarrow F + L \leftrightarrow FL \qquad (6)$$

[1] Bai, N., Roder, H., Dickson, A., & Karanicolas, J. (2019). Isothermal analysis of ThermoFluor data can readily provide quantitative binding affinities. *Scientific reports*, *9*(1), 1-15.

[2] Niebling, S., Burastero, O., Bürgi, J., Günther, C., Defelipe, L. A., Sander, S., ... & García-Alai, M. (2021). FoldAffinity: binding affinities from nDSF experiments. *Scientific reports*, *11*(1), 1-17.

where U, F, FL and L are respectively the unfolded state, folded state, bound folded state and ligand. The associated equations and principle of mass conservation are

$$K_u = U / F \quad (7)$$
$$K_d = (F * L) / FL \quad (8)$$
$$P_0 = U + F + FL \quad (9)$$
$$L_0 = L + FL \quad (10)$$

where K_u and K_d are respectively the unfolding and the equilibrium dissociation constant, $P_0$and $L_0$are respectively the total protein and total ligand concentration.

If we knew $K_u$and K_d, FL could be obtained by solving

$$0 = FL^2 + pFL + q \quad (11)$$
$$p = P_0 / (K_u + 1) + L_0 + K_d \quad (12)$$
$$q = P_0 L_0 / (K_u + 1) \quad (13)$$

And then$U$, and F could be calculated to determine the unfolded fraction ($U / (F + FL)$). Therefore, at a fixed temperature, the unfolded fraction versus ligand concentration curve allows to estimate $K_u$and K_d.

Two binding sites (microscopic constants)

The system is described by the reactions

$$U + 2L \leftrightarrow F + 2L \quad (14)$$
$$F + 2L \leftrightarrow FL + L \quad (15)$$
$$F + 2L \leftrightarrow LF + L \quad (16)$$
$$FL + L \leftrightarrow LFL \quad (17)$$
$$LF + L \leftrightarrow LFL \quad (18)$$

where F is the free folded protein, FL and LF are the two possible protein-ligand complexes and LFL is the protein with two ligands. F, FL, LF and LFL depend on the Kds, total ligand concentration L_0 and free ligand concentration L_{free}in the following way

$$F = K_{d,1} * K_{d,2} * (L_0 - L_{free}) / (K_{d,1} + K_{d,2} + 2L_{free}) / L_{free} \quad (19)$$
$$FL = L_{free} * F / K_{d,2} \quad (20)$$

$$FL = L_{free} * F / K_{d,1} \quad (21)$$

$$LFL = L_{free} * LF / K_{d,2} \quad (22)$$

We can obtain the value of L_{free} by solving

$$X^3 + pX^2 + qX + r = 0 \quad (23)$$

where

$$p = K_{d,1} + K_{d,2} + (2 * P_0 - L_0) \quad (24)$$

$$q = (P_0 - L_0) * (K_{d,1} + K_{d,2}) + K_{d,1} * K_{d,2} * (1 + K_u) \quad (25)$$

$$r = - L_0 * K_{d,1} * K_{d,2} * (1 + K_u) \quad (26)$$

For simplicity, for now we only provide the option to fit this model using $K_{d,1} = K_{d,2}$.

3.2. Curve fitting

The unfolded fraction versus ligand concentration curve is fitted using the Levenberg Marquardt (damped least-squares) algorithm to estimate K_d (or $K_{d,1}$, $K_{d,2}$) and K_u at the chosen temperature.

3.3. Fitting errors

The standard deviation of all fitted parameters is computed using the square root of diagonal values from the fit parameter covariance matrix.

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.[3]

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

$$RSS(K_d) = RSS_0(1 + \frac{criticalValue}{n-p}) \quad (27)$$

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

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 95 %.

4. Alternative Tm fitting: Binding affinity from the observed melting temperatures

4.1 Model

If we suppose that the enthalpy (ΔH) and entropy (ΔS) of unfolding, and the K_d do not change significantly in the vicinity of the melting temperature T_m of the protein, the free energy of unfolding for a one binding-site system can be expressed as:

$$\Delta G(T_{m,Obs}) = \Delta H - T_{m,Obs}\Delta S + RT_{m,Obs} * ln(1 + \frac{L_{free}}{Kd}) \quad (28)$$

and for a two-binding sites system,

$$\Delta G(T_{m,Obs}) = \Delta H - T_{m,Obs}\Delta S + RT_{m,Obs} * ln(1 + L_{free} * \frac{L_{free}+K_{d,1}+K_{d,2}}{K_{d,1}*K_{d,2}}) \quad (29)$$

Taking into account that at the T_m, without ligand, we have the following equality:

$$\Delta H = T_m \Delta S \quad (30)$$

If we approximate the free ligand concentration using the total ligand concentration we can fit the observed melting temperatures by using

$$T_{m,Obs} = T_m(1 - \frac{RT_m ln(1+L_{free}/K_d)}{\Delta H})^{-1} \quad (31)$$

in the case of one-binding site, or

$$T_{m,Obs} = T_m(1 - \Delta H^{-1}(RT_m ln(1 + L_{free} * \frac{L_{free}+K_{d,1}+K_{d,2}}{K_{d,1}*K_{d,2}})))^{-1} \quad (32)$$

in case of two-binding sites.

4.2. Curve fitting

The observed melting temperature calculated from the first derivative as a function of the total ligand concentration is fitted using the Levenberg Marquardt (damped least-squares) algorithm to estimate ΔH and K_d (or $K_{d,1}$, $K_{d,2}$).

4.3. Fitting errors

The standard error of all fitted parameters is based on the covariance matrix (using the R programming language package *minpack.lm*).

When fitting the '1:1' or the '1:2' (one Kd) binding models, we also provide the marginal asymmetric confidence interval (See Section 3.3.).

Contact details

For further assistance, please contact us:

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

Packages

FoldAffinity 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

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/