Dye/DNA Plates Documentation

Release v3.0.0

Robert F. DeJaco

Mar 15, 2023

CONTENTS:

1 Modules **1**

1.1 Wells . . . 1
1.2 Get Data . . . 3
1.3 Quantifying Noise and Linearity . . . 5
1.4 Parameter Extraction . . . 6
1.5 Plotting . . . 9
1.5.1 Raw Data . . . 9
1.5.2 Quantifying Noise and Linearity . . . 10
1.5.3 Parameters . . . 11
1.6 Util . . . 12

2 Indices and tables **15**

Python Module Index **17**

CHAPTER

ONE

MODULES

1.1 Wells

src.wells.**column_row_to_well**(*ix: int*, *iy: int*) → str

Convert indices to well

```
>>> column_row_to_well(0, 0)
'A1'
>>> column_row_to_well(0, 7)
'H1'
>>> column_row_to_well(11, 0)
'A12'
>>> column_row_to_well(11, 7)
'H12'
>>> well_to_row(column_row_to_well(11, 7))
7
>>> well_to_column(column_row_to_well(11, 7))
11
```

Parameters

- **ix** (*int*) – a index
- **iy** (*int*) – y index

Returns well

Return type str

src.wells.**number_to_column**(*number: int*) → int

```
>>> number_to_column(1)
1
>>> number_to_column(0)
0
>>> number_to_column(11)
11
>>> number_to_column(95)
11
>>> number_to_column(84)
0
```

src.wells.**number_to_row**(*number: int*) → int

```
>>> number_to_row(0)
0
>>> number_to_row(1)
0
>>> number_to_row(11)
0
>>> number_to_row(12)
1
>>> number_to_row(95)
7
>>> number_to_row(84)
7
```

src.wells.**number_to_well**(*number: int*) → str
Return number of well

```
>>> number_to_well(0)
'A1'
>>> number_to_well(11)
'A12'
>>> number_to_well(95)
'H12'
>>> number_to_well(84)
'H1'
```

src.wells.**well_to_column**(*well: str*) → int
Convert well name to column

```
>>> well_to_column('H100')
Traceback (most recent call last):
    ...
ValueError: '100' is not in list
>>> well_to_column('H1')
0
>>> well_to_column('A1')
0
>>> well_to_column('B12')
11
>>> well_to_column('B13')
Traceback (most recent call last):
    ...
ValueError: '13' is not in list
>>> well_to_column('Z1')
0
```

Parameters **well** (*str*) – name of well

Returns **ix** – a-index for well

Return type int

src.wells.**well_to_number**(*well: str*) → int
Return number of well

```
>>> well_to_number("A1")
0
>>> well_to_number("A12")
```

(continues on next page)

(continued from previous page)

```
11
>>> well_to_number("H12")
95
>>> well_to_number("H1")
84
```

src.wells.**well_to_row**(*well: str*) → int

Well to y index

```
>>> well_to_row('H100')
7
>>> well_to_row('H1')
7
>>> well_to_row('A1')
0
>>> well_to_row('B1')
1
>>> well_to_row('Z1')
Traceback (most recent call last):
    ...
ValueError: 'Z' is not in list
```

Parameters **well** (*str*) – well name

Returns **iy** – y-index of well

Return type int

1.2 Get Data

class src.get_data.**CombinedData**(**replicates*)

Combined data from several replicate plates. See changes to F and C in *Raw Data and Scaling* portion of *Results and Discussion.*

Variables

- **F** (*np.ndarray*) – Fluorescence data, $\mathbf{F}_{\mathbf{D}}^{t}$ in Equation (13a)
- **C** (*np.ndarray*) – Dye concentrations, $\mathbf{C}_{\mathbf{D}}^{t}$ in Equation (13a)
- **D** (*float*) – DNA concentration, $\mathbf{D}$ in Equation (13a)
- **t** (*str*) – Type of DNA, t, "SS" or "DS"
- **N** (*int*) – number of nucleobases per single strand, N
- **M_tls** (*np.array*) – $\mathbf{M}^{\mathrm{TLS}}$ in Equation (18a), set externally, defaults to np.array([])
- **C_hat** (*np.array*) – $\widehat{\mathbf{C}}$ in Equation (18a), set externally, defaults to np.array([])
- **V_M** (*np.array*) – $V(\mathbf{M})$ (see section S1.3 in supporting material), defaults to np.array([])
- **V_C** (*np.array*) – $V(\mathbf{C})$ (see section S1.3 in supporting material), defaults to np.array([])
- **M_std** (*np.array*) – $\sqrt{V(\mathbf{M})}$, defaults to np.array([])
- **C_std** (*np.array*) – $\sqrt{V(\mathbf{C})}$, defaults to np.array([])

__init__ (**replicates*) → None
Combine replicate $F_d^{t,l}$ and C

Parameters **replicates** (*typing.List[Data]*) – list of plates to gather together as replicates

make_subset (*F_min*)
Make subset of data as described in *Raw Data and Scaling* portion of manuscript

---

Note: **F** and **C** are overwritten.

---

class src.get_data.**RawData** (*fluorescence_file_name*, *B_d*, *t*, *l*, *N*, *dye_conc_file_name='dye_conc_uM.csv'*)
Stores raw data.

Variables

- **F** (*np.ndarray*) – Fluorescence data, $F_d^{t,l}$ (see Eq. 11)
- **C** (*np.ndarray*) – Dye concentrations, C (see Eq. 10)
- **B** (*float*) – DNA concentration, B_d in mol/L
- **t** (*str*) – Type of DNA, t, "SS" or "DS" or "None".
- **l** (*str*) – Replicate name, l is "A", "B", or code:*"C"*.
- **N** (*int*) – number of nucleobases per single strand

__init__ (*fluorescence_file_name*, *B_d*, *t*, *l*, *N*, *dye_conc_file_name='dye_conc_uM.csv'*)
Scale the data before interpolating/solving optimization problem.

Parameters

- **fluorescence_file_name** (*str*) – name of fluorescence file within data folder
- **B_d** (*float*) – Total concentration of nucleobases in mol/L B_d
- **dye_conc_file_name** (*str, optional*) – name of dye concentration file name within data folder, defaults to "dye_conc_uM.csv"
- **t** (*str*) – Type of DNA, "SS", "DS", or "None".
- **l** (*str*) – Replicate name, ℓ is A, B, or C
- **N** (*int*) – number of nucleobases per single strand

src.get_data.**excel_to_data** (*f_name: str*, *channel='GREEN'*)
Convert "Raw Data" sheet of excel file to pandas dataframe. Uses Equation (9) of manuscript to calculate temperature associated with each cycle.

Parameters

- **f_name** (*str*) – name of excel file
- **channel** (*str, optional*) – name of channel to investigate, defaults to "GREEN"

Returns Formatted data frame, with wells sorted from A1, A2 … H11, H12

Return type pd.DataFrame

src.get_data.**get_C** (*file_name*)
Get total dye concentration associated with each well.

Parameters **file_name** (*str*) – CSV file containing values of concentrations of dye in units of μ mol/L. Formatted like a 96-well plate:

Row	1	2	3	4	5	6	7	8	9	10	11	12
A	.	.	.	.	.	.	.	.	.	.	.	.
B	.	.	.	.	.	.	.	.	.	.	.	.
C	.	.	.	.	.	.	.	.	.	.	.	.
D	.	.	.	.	.	.	.	.	.	.	.	.
E	.	.	.	.	.	.	.	.	.	.	.	.
F	.	.	.	.	.	.	.	.	.	.	.	.
G	.	.	.	.	.	.	.	.	.	.	.	.
H	.	.	.	.	.	.	.	.	.	.	.	.

Returns Mapping of well name ("A1",...) to dye concentration [units of mol/L]

Return type Dictionary

1.3 Quantifying Noise and Linearity

src.noise_removal.**compute_M_LS**(*F*, *C*)
Calculate $\mathbf{M}$ by least-squares approximation

Returns $\mathbf{M}^{\mathrm{LS}}$, see Eq. 17.

Return type np.array

src.noise_removal.**compute_M_plus**(*F*, *c_plus*)
Get updated guess for M

Parameters

- **F** (*np.ndarray*) – Fluorescence matrix $\mathbf{F}$
- **c_plus** (*np.array*) – Concentration matrix updated $\mathbf{c}_+$

Returns $\mathbf{M}_+$ by Eq. S3b (see supporting material).

Return type np.array

src.noise_removal.**compute_c_plus**(*F*, *C*, *M_minus*, *rho_squared*)
Compute updated guess of concentrations, $\mathbf{c}_+$

Parameters

- **F** (*np.ndarray*) – Fluorescence $\mathbf{F}$
- **C** (*np.array*) – Dye Concentration $\mathbf{C}$
- **M_minus** (*np.array*) – Guess for $\mathbf{M}$, $\mathbf{M}_-$
- **rho_squared** (*float*) – Weight, ρ^2

Returns $\mathbf{c}_+$ by Eq. S3a (see supporting material).

Return type np.ndarray

src.noise_removal.**predictor_corrector**(*F*, *C*, *rho_squared*, *maxiter=100000*, *print_iter=True*)
Solve Eqs. 18 with predictor–corrector algorithm

Parameters

- **F** (*np.ndarray*) – Fluorescence data $\mathbf{F}$
- **C** (*np.ndarray*) – Dye concentration data $\mathbf{C}$
- **rho_squared** (*float*) – Weighting factor for concentrations, ρ^2 in Equation (18a)
- **maxiter** (*int, optional*) – maximum iterations allowed, by default 100000
- **print_iter** (*bool, optional*) – whether or not to print the total number of iterations performed, by default True

Returns **(M, c)** – solution, $(\mathbf{M}^{\mathrm{TLS}}, \widehat{\mathbf{C}})$

Return type tuple(np.array, np.array)

1.4 Parameter Extraction

class src.parameter_extraction.**Parameters**(*cls1: src.get_data.CombinedData*, *cls2: src.get_data.CombinedData*)

Stores multiple instances of CombinedData for one DNA type

Notes

$\mathbf{M}^{\mathrm{TLS}}$ and $\widehat{\mathbf{C}}$ are defined in Eq. 18a.

Variables

- **N** (*int*) – number of nucleobases, N
- **M1** (*np.array*) – $\mathbf{M}^{\mathrm{TLS}}$ associated with $\mathbf{D} = 1$
- **M2** (*np.array*) – $\mathbf{M}^{\mathrm{TLS}}$ associated with $\mathbf{D} = 2$. Several high temperatures are removed to reflect smaller temperature range associated with $\mathbf{D} = 1$
- **C1** (*np.array*) – $\widehat{\mathbf{C}}$ associated with $\mathbf{D} = 1$
- **C2** (*np.array*) – $\widehat{\mathbf{C}}$ associated with $\mathbf{D} = 2$
- **r** (*np.array*) – r, as defined in Equation (S7).
- **V_C1** (*np.array*) – $V(\mathbf{C})$ associated with $\mathbf{D} = 1$
- **V_C2** (*np.array*) – $V(\mathbf{C})$ associated with $\mathbf{D} = 2$
- **V_M1** (*np.array*) – $V(\mathbf{M})$ associated with $\mathbf{D} = 1$
- **V_M2** (*np.array*) – $V(\mathbf{M})$ associated with $\mathbf{D} = 2$. Several high temperatures are removed to reflect smaller temperature range associated with $\mathbf{D} = 1$
- **dT** (*float*) – Change in temperature from one cycle to next, ΔT

__init__(*cls1: src.get_data.CombinedData*, *cls2: src.get_data.CombinedData*)

Initialize data

Note: Since different temperature ranges for each, need to make subset of dataset that has more temperatures. Dataset with lower DNA concentration `cls1` always has less temperatures.

Parameters

- **cls1** (CombinedData) – Data of DNA type at $\mathbf{D} = 1$

- **cls2** (CombinedData) – Data of DNA type at $\mathbf{D} = 2$

get_K() → numpy.array
Get $\mathbf{K}$ from vectorized version of Eq. 21.

Returns $\mathbf{K}$

Return type np.array

get_K_std() → numpy.array
Get standard deviation estimate of $\mathbf{K}$

Returns

$$\sqrt{\Delta\mathbf{M}^2\left(4V(\mathbf{M}_1)+2V(\mathbf{M}_2)\right)+\frac{\Delta H^2}{8\Delta\mathbf{M}^4}\left(V(\mathbf{M}_1)+V(\mathbf{M}_2)\right)}$$

where $\Delta H := 2\mathbf{M}_1 - \mathbf{M}_2$, $\Delta\mathbf{M} := \mathbf{M}_2 - \mathbf{M}_1$

Return type np.array

get_dg() → numpy.array
Get free energy of dye binding, Δg, vectorized version of Eq. 25.

Returns

Return type np.array

get_dg_std() → numpy.array
Get estimate of standard deviation in Δg.

Returns

$$\frac{RT_j}{2\Delta\mathbf{M}_j\left(2\mathbf{M}_{j1}-\mathbf{M}_{j2}\right)}\sqrt{\mathbf{M}_{j2}^2V(\mathbf{M}_{j1})+\mathbf{M}_{j1}^2V(\mathbf{M}_{j2})}$$

where $\Delta\mathbf{M}_j = \mathbf{M}_{j2} - \mathbf{M}_{j1}$.

Return type np.array

get_dh() → numpy.array
Get differential enthalpy of binding, Δh as the vectorized version of Eq. 26.

Returns

Return type np.array

get_dh_std() → numpy.array
Get estimate of error in Δh_j

Returns

Return type np.array

get_f() → numpy.array
Get $\mathbf{f}$ from vectorized version of Eq. 23.

Returns $\mathbf{f}$

Return type np.array

get_f_std() → numpy.array
Get standard deviation estimate of $\mathbf{f}$

Returns

$$\sqrt{V(\mathbf{M}_1) + V(\mathbf{M}_2) + \frac{E_B^2 V_A + E_A^2 V_B}{E_B^4}}$$

where

$$\begin{aligned} V_A &:= (V_B + H_+^2)(V(\mathbf{M}_1) + V(\mathbf{M}_2) + \Delta\mathbf{M}^2) - E_A^2 \\ V_B &:= 2V(\mathbf{M}_1) + V(\mathbf{M}_2) \\ E_A &:= \Delta\mathbf{M} H_+ \\ E_B &:= 2\mathbf{M}_1 - \mathbf{M}_2 \\ \Delta\mathbf{M} &:= \mathbf{M}_2 - \mathbf{M}_1 \\ H_+ &:= 2\mathbf{M}_1 + \mathbf{M}_2 \end{aligned}$$

Return type np.array

get_theta_b_all_1(*n*)

Returns $\theta_{b,j,1}$ from Eq. S5 pointwise in j The index b is the total number of wells.

Returns

Return type np.array

src.parameter_extraction.**calculate_relative_brightness**(*f_SS*, *f_DS*)

Calculate relative brightness, Equation (24).

Parameters

- **f_SS** (*np.array*) – Molar fluorescence of single-stranded DNA, $\mathbf{f}^{\mathrm{SS}}$.
- **f_DS** (*np.array*) – Molar fluorescence of double-stranded DNA, $\mathbf{f}^{\mathrm{DS}}$.

Returns Relative brightness, $\mathbf{f}_j^{\mathrm{DS}}/\mathbf{f}_j^{\mathrm{SS}}$ for each j associated with SS. (see Eq. 24).

Return type np.array

src.parameter_extraction.**calculate_relative_brightness_err**(*SS_M1*, *SS_M2*, *DS_M1*, *DS_M2*, *SS_V_M1*, *SS_V_M2*, *DS_V_M1*, *DS_V_M2*) → numpy.array

Estimate error in relative brightness (Eq. 24).

Parameters

- **SS_M1** (*np.array*) – $\mathbf{M}_1^{\mathrm{SS}}$
- **SS_M2** (*np.array*) – $\mathbf{M}_2^{\mathrm{SS}}$
- **DS_M1** (*np.array*) – $\mathbf{M}_1^{\mathrm{DS}}$
- **DS_M2** (*np.array*) – $\mathbf{M}_2^{\mathrm{DS}}$
- **SS_V_M1** (*np.array*) – $V(\mathbf{M}_1^{\mathrm{SS}})$ _description_
- **SS_V_M2** (*np.array*) – $V(\mathbf{M}_2^{\mathrm{SS}})$
- **DS_V_M1** (*np.array*) – $V(\mathbf{M}_1^{\mathrm{DS}})$
- **DS_V_M2** (*np.array*) – $V(\mathbf{M}_2^{\mathrm{DS}})$

Returns Estimate of error in relative brightness

Return type np.array

1.5 Plotting

1.5.1 Raw Data

src.plot_raw_data.**make_figure_2**(*SS_A_1: src.get_data.RawData*, *SS_B_1: src.get_data.RawData*, *SS_C_1: src.get_data.RawData*, *SS_A_2: src.get_data.RawData*, *SS_B_2: src.get_data.RawData*, *DS_A_1: src.get_data.RawData*, *DS_B_1: src.get_data.RawData*, *DS_A_2: src.get_data.RawData*, *DS_B_2: src.get_data.RawData*, *A_1: src.get_data.RawData*)

Makes figure 2

Parameters

- **SS_A_1** (RawData) – Data associated with $(t, l, d) = (SS, A, 1)$ (see Table 1 of main text)
- **SS_B_1** (RawData) – Data associated with $(t, l, d) = (SS, B, 1)$ (see Table 1 of main text)
- **SS_C_1** (RawData) – Data associated with $(t, l, d) = (SS, C, 1)$ (see Table 1 of main text)
- **SS_A_2** (RawData) – Data associated with $(t, l, d) = (SS, A, 2)$ (see Table 1 of main text)
- **SS_B_2** (RawData) – Data associated with $(t, l, d) = (SS, B, 2)$ (see Table 1 of main text)
- **DS_A_1** (RawData) – Data associated with $(t, l, d) = (DS, A, 1)$ (see Table 1 of main text)
- **DS_B_1** (RawData) – Data associated with $(t, l, d) = (DS, B, 1)$ (see Table 1 of main text)
- **DS_A_2** (RawData) – Data associated with $(t, l, d) = (DS, A, 2)$ (see Table 1 of main text)
- **DS_B_2** (RawData) – Data associated with $(t, l, d) = (DS, B, 2)$ (see Table 1 of main text)
- **A_1** (RawData) – Data associated with $(l, d) = (A, 1)$ (without DNA, see Table 1 of main text)

src.plot_raw_data.**make_figure_S1**()

Makes Figure S1.

src.plot_raw_data.**make_figure_S3**()

Makes Figure S3.

src.plot_raw_data.**plot_linemap**(*cls: src.get_data.RawData*, *ax*, *colorbar=False*, *get_ticks=False*, *ordered_by_row=True*)

Plot $F_d^t(T_j, C_i)$ vs C_i for various temperatures T_j (colors)

Parameters

- **cls** (*Data*) – Instance of data (i.e., a dataset)
- **ax** (*axis*) – Matplotlib axis to plot on

- **colorbar** (*bool, optional*) – whether or not to plot colorbar, in which case the axis is colorbar axis, by default False
- **get_ticks** (*bool, optional*) – whether or not to return list of ticks, by default False
- **ordered_by_row** (*bool, optional*) – whether or not well concentrations are ordered by row, by default True

Returns Only returns list of `get_ticks=True`.

Return type None or list

1.5.2 Quantifying Noise and Linearity

src.plot_noise_removal.**plot_Chat_vs_C**(*Cs*, *C_hats*, *C_stds*, *fname*)
Plot comparison between $\mathbf{C}$ and $\widehat{\mathbf{C}}$. Used to plot Fig. 6

Parameters

- **Cs** (*tuple of np.arrays*) – each element of the tuple corresponds to $\mathbf{C}$ for a specific $\mathbf{D}$ and type.
- **C_hats** (*tuple of np.arrays*) – each element of the tuple corresponds to $\widehat{\mathbf{C}}$ for a specific $\mathbf{D}$ and type.
- **C_stds** (*tuple of np.arrays*) – each element of the tuple corresponds to the estimate for the standard deviation in $\widehat{\mathbf{C}}$ for a specific $\mathbf{D}$ and type.
- **fname** (*str*) – file name to save as figure (relative path).

src.plot_noise_removal.**plot_Fhat_vs_F**(*Fs*, *Fhats*, *Ts*, *fname*, *sname='\\widehat{\\mathbf{F}}_{ji}^\\mathrm{LS}'*)

Parameters

- **Fs** (*tuple of np.array*) – each element of the tuple corresponds to $\mathbf{F}$ for a specific $\mathbf{D}$ and type.
- **Fhats** (*tuple of np.array*) – each element of the tuple corresponds to $\mathbf{F}$ for a specific $\mathbf{D}$ and type.
- **Ts** (*tuple of np.array*) – each element of the tuple corresponds to the temperatures associated with a specific $\mathbf{D}$ and type.
- **fname** (*str*) – name of figure to plot (relative path)
- **sname** (*str*) – name of symbol in latex

src.plot_noise_removal.**plot_error_C**(*ax*, *C*, *Chat*, *Chat_std=None*, ***kwargs*)
Plot Errors in Dye concentration

Parameters

- **ax** (*matplotlib.pyplot.axis*) – plots on this axis
- **C** (*np.array*) – Nominal dye concentrations determined experimentally (dimensionless)
- **Chat** (*np.array*) – Dye concentrations determined from total least squares
- **Chat_std** (*np.array*) – Estimate of standard deviations in dye concentrations determined from total least squares
- **kwargs** – Used to determine color if not provided. Also used when calling `ax.plot`.

src.plot_noise_removal.**plot_error_F**(*ax*, *F_k_tl*, *F_hat_k_tl*, *T*)
Plot the predicted fluorescence $\widehat{\mathbf{F}}_{ijk}^{t\ell}$ against $\mathbf{F}_{ijk}^{t\ell}$ for $i = 1, \ldots, n$ and $j = 1, \ldots, n$.

Parameters

- **ax** (*matplotlib.pyplot.axes*) – axes to plot on
- **F_kl** (*np.ndarray*) – Experimentally measured fluorescence $\mathbf{F}_k^{t\ell}$.
- **F_hat_kl** (*np.ndarray*) – Predicted fluorescence $\widehat{\mathbf{F}}_k^{t\ell}$.
- **T** (*np.array*) – Temperatures in K

Returns image for colorbar

Return type img

src.plot_noise_removal.**plot_figure7**(*M_mean*, *M_std*, *T*, *color_D1='C0'*, *color_D2='C1'*)
Plot figure 7

Parameters

- **M_mean** (*tuple of np.array*) – each element of the tuple corresponds to $\mathbf{M}^{\mathrm{TLS}}$ for a specific $\mathbf{D}$ and type.
- **M_std** (*tuple of np.array*) – each element of the tuple corresponds to standard deviations in $\mathbf{M}^{\mathrm{TLS}}$ for a specific $\mathbf{D}$ and type.
- **T** (*tuple of np.array*) – each element of the tuple corresponds to the temperatures associated with a specific $\mathbf{D}$ and type.
- **color_D1** (*str*) – name of color for data associated with $\mathbf{D} = 1$
- **color_D2** – name of color for data associated with $\mathbf{D} = 2$

1.5.3 Parameters

src.plot_params.**plot_figure8**(*T_SS*, *f_SS*, *K_SS*, *f_std_SS*, *K_std_SS*, *T_DS*, *f_DS*, *K_DS*, *f_std_DS*, *K_std_DS*)
Plot figure 8

Parameters

- **T_SS** (*np.array*) – temperatures for single-stranded DNA
- **f_SS** (*np.array*) – molar fluorescence for single-stranded DNA
- **K_SS** (*np.array*) – partition coefficients for single-stranded DNA
- **f_std_SS** (*np.array*) – standard deviation in molar fluorescence of single-stranded DNA
- **K_std_SS** (*np.array*) – standard deviation in partition coefficients for single-stranded DNA
- **T_DS** (*np.array*) – temperatures for double-stranded DNA
- **f_DS** (*np.array*) – molar fluorescence for double-stranded dna
- **K_DS** (*np.array*) – partition coefficients for single-stranded DNA
- **f_std_DS** (*np.array*) – standard deviation in molar fluorescence of double-stranded DNA
- **K_std_DS** (*np.array*) – standard deviation in partition coefficients for double-stranded DNA

src.plot_params.**plot_figure9**(*SS: src.parameter_extraction.Parameters*, *DS: src.parameter_extraction.Parameters*)

Parameters

- **SS** (Parameters) – object containing all single-stranded DNA input and methods to perform calculations
- **DS** (Parameters) – object containing all single-stranded DNA input and methods to perform calculations

src.plot_params.**plot_figure_S2**(*SS_1: src.get_data.CombinedData*, *SS_2: src.get_data.CombinedData*, *DS_1: src.get_data.CombinedData*, *DS_2: src.get_data.CombinedData*)

Plot figure S2

Parameters

- **SS_1** (CombinedData) – object containing all replicate plates for single-stranded DNA with $\mathbf{D} = 1$
- **SS_2** (CombinedData) – object containing all replicate plates for single-stranded DNA with $\mathbf{D} = 2$
- **DS_1** (CombinedData) – object containing all replicate plates for double-stranded DNA with $\mathbf{D} = 1$
- **DS_2** (CombinedData) – object containing all replicate plates for double-stranded DNA with $\mathbf{D} = 2$

src.plot_params.**plot_figure_S4**(*SS_data: src.parameter_extraction.Parameters*, *DS_data: src.parameter_extraction.Parameters*)

Plot Figure S4

Parameters

- **SS_data** (Parameters) – parameters class that has method to calculate $\theta_{b,j,1}$ pointwise in j
- **DS_data** (Parameters) – parameters class that has method to calculate $\theta_{b,j,1}$ pointwise in j

src.plot_params.**plot_figure_S5**(*T*, *rb*, *d_rb*)

Plot figure S5, relative brightness

Parameters

- **T** (*np.array*) – array of temperatures
- **rb** (*np.array*) – Relative brightness, $\mathbf{f}_j^{\mathrm{DS}}/\mathbf{f}_j^{\mathrm{SS}}$
- **d_rb** (*np.array*) – standard deviation in relative brightness

1.6 Util

src.util.**figure_name_to_abspath**(*fname: str*) → str

Figure name to absolute path

Parameters **fname** (*str*) – name of figure

Returns absolute path to name of figure

Return type str

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

src.get_data, 3
src.noise_removal, 5
src.parameter_extraction, 6
src.plot_noise_removal, 10
src.plot_params, 11
src.plot_raw_data, 9
src.util, 12
src.wells, 1