Bias-UQ-PCR

Release v0.9.0

Robert F. DeJaco

Mar 29, 2023

CONTENTS:

CHAPTER ONE

GET DATA

src.get_data.**cycle_to_index**(*cycle: int*) → int

Parameters
`cycle` (*int*) – cycle number i, 0 to n

Returns
index in matrix or array

Return type
int

src.get_data.**file_to_numpy**(*file_abs_path: str*) → ndarray

Parameters
`file_abs_path` (*str*) – path to file name

Notes

All fluorescence values are divided by 10^6 before any analysis

Returns
n by m matrix of fluorescence, **F**

Return type
np.ndarray

src.get_data.**index_to_cycle**(*index: int*) → int

Parameters
`index` (*int*) – index of cycle (0 to n-1)

Returns
cycle number (1 to n)

Return type
int

CHAPTER

TWO

WELLS

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

Convert indices to well

Parameters

- **`ix`** (*`int`*) – x (column) index
- **`iy`** (*`int`*) – y (row) index

Returns

name of well

Return type

str

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

Get column index associated with well number

Parameters

`number` (*`int`*) – well number (0 to 95)

Returns

column (0 to 11)

Return type

int

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

Get the row number associated with a well number

Parameters

`number` (*`int`*) – well number

Returns

row number (0 to 7)

Return type

int

`src.wells.`**`number_to_well`**(*number: int*) → str

Get well name from number

Parameters

`number` (*`int`*) – well number (0 to 95)

Returns

well name (A1 to H12)

Return type
str

`src.wells.`**`well_to_column`**(*well: str*) → int

Convert well name to column

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

Returns
ix – x-index for well (row)

Return type
int

`src.wells.`**`well_to_column_row`**(*well: str*) → Tuple[int]

Convert well name to column, row

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

Returns
column index, row index

Return type
tuple(int, int)

`src.wells.`**`well_to_number`**(*well: str*) → int

Returns number of well

Parameters
`well` (*`str`*) – name of well (e.g., `"A1"`)

Returns
well number (0 to 95)

Return type
int

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

Well to y index

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

Returns
iy – y-index of well (row)

Return type
int

CHAPTER

THREE

AMPLIFICATION

class src.amplification.**Amplification**(*R: float*, *pbar: float*, *E_U0: array*, *V_U0: ndarray*)

Parameters

- **l1** (*float*) – Largest eigenvalue of $\mathbf{A}$, λ_1
- **l2** (*float*) – Smallest eigenvalue of $\mathbf{A}$, λ_2
- **x1** (*np.array*) – Right eigenvector of $\mathbf{A}$ corresponding to λ_1, $\mathbf{x}_1$
- **x2** (*np.array*) – Right eigenvector of $\mathbf{A}$ corresponding to λ_2, $\mathbf{x}_2$
- **z1** (*np.array*) – Left eigenvector of $\mathbf{A}$ corresponding to λ_1, $\mathbf{z}_1$
- **z2** (*np.array*) – Left eigenvector of $\mathbf{A}$ corresponding to λ_2, $\mathbf{z}_2$
- **K1** (*np.ndarray*) – Matrix $\mathbf{K}_1$ defined in (25)
- **K2** (*np.ndarray*) – Matrix $\mathbf{K}_2$ defined in (25)

__init__(*R: float*, *pbar: float*, *E_U0: array*, *V_U0: ndarray*)

Parameters

- **R** (*float*) – ratio of amplification probabilities, R
- **pbar** (*float*) – geometric mean of amplification probabilities, $\bar{p}$
- **E_U0** (*np.array*) – initial expected values of both strands $\mathbb{E}\left[\mathbf{U}_0\right]$, 2 by 1 matrix
- **V_U0** (*np.ndarray*) – initial variances of both strands $\mathsf{Var}\left[\mathbf{U}_0\right]$, 2 by 2 matrix

get_A() → ndarray

Returns
matrix $\mathbf{A}$ calculated by decomposition (13)

Return type
np.ndarray

get_Atoi(*i: int*) → ndarray

Parameters
i (*int*) – cycle number

Returns
matrix $\mathbf{A}^i$ calculated by decomposition (15)

Return type
np.ndarray

`get_EXi_over_EYi`(*cycles: array*) → array

Calculate $\mathbb{E}\left[X_i\right]/\mathbb{E}\left[Y_i\right]$ for a variety of cycles i

Parameters
`cycles` (*`np.array`*) – cycles (integers) to calculate for each i

Returns
$\mathbb{E}\left[X_i\right]/\mathbb{E}\left[Y_i\right]$

Return type
np.array

`get_E_Ui`(*i: int*) → array

Parameters
`i` (*`int`*) – cycle number

Returns
matrix $\mathbb{E}\left[\mathbf{U}_i\right] = \mathbf{A}^i\mathbb{E}\left[\mathbf{U}_0\right]$

Return type
np.array

`get_V_Ui`(*i: int*) → ndarray

Parameters
`i` (*`int`*) – cycle number

Returns
$\mathsf{Var}\left[\mathbf{U}_i\right]$ obtained by Equation (27)

Return type
np.ndarray

`initialize`()

Initialize $\mathbf{K}_\ell$ via (25), $\nu_{j,k}$ via (28a), and $\eta^\ell_{j,k}$ via (28b).

`src.amplification.`**`eval_R`**(*p_fr: float*, *p_rf: float*)

Calculate R from p_{rf} and p_{fr}

Parameters

- **`p_fr`** (*`float`*) – probability of forward to reverse amplification, p_{fr}
- **`p_rf`** (*`float`*) – probability of reverse to forward amplification, p_{rf}

Returns
$R = \sqrt{\frac{p_{\text{rf}}}{p_{\text{fr}}}}$

Return type
float

`src.amplification.`**`get_pfr_prf`**(*R: float*, *pbar: float*)

Get p_{fr} and p_{rf} from R and $\bar{p}$

Parameters

- **`R`** (*`float`*) – square root of ratio of probabilities, R
- **`pbar`** (*`float`*) – geometric mean of probabilities, $\bar{p}$

Return type
tuple(p_fr, p_rf)

CHAPTER

FOUR

MOLAR FLUORESCENCE

class src.molar_fluorescence.**MolarFluorescence**(*C: array*, *files: List[str]*, *name: str*)

Parameters

- **f** (*np.ndarray*) – molar fluorescences for each cycle/well f, n by number of wells matrix Determined in units of fluorescence divided by (pmol/L)
- **df** (*np.ndarray*) – standard deviation in molar fluorescences σ, n by number of wells matrix Determined in units of fluorescence divided by (pmol/L)
- **cv** (*np.ndarray*) – Coefficient of variation in molar fluorescences $\sigma/\mathtt{f}$,
- **q** (*int*) – number of plates (different concentrations)
- **n** (*int*) – number of cycles
- **m** (*int*) – number of wells
- **F** (*np.ndarray*) – raw fluorescence data (scaled by 10^6 as described in get_data.py). Tensor of dimensions $n \times m \times q$

__init__(*C: array*, *files: List[str]*, *name: str*)

Parameters

- **C** (*np.array*) – concentrations of reporter in pmol/L
- **files** (*List[str]*) – list of file names (absolute paths)
- **name** (*str*) – name of reporter (e.g., FAM, Probe)

calculate()

Perform calculations of f and standard deviation σ

print_cv()

Print information relating to coefficient of variation

CHAPTER

FIVE

KINETIC PCR

class `src.kinetic_PCR.`**`HydrolysisProbes`**(*C: float*, *Vol: float*, *f_plus: ndarray*, *f_minus: ndarray*, *R: float*, *pbar: float*, *E_U0: array*, *V_U0: ndarray*, *s_plus: ndarray*, *s_minus: ndarray*, ***kwargs*)

Specific subclass for hydrolysis probes

Parameters

- **n** (*int*) – number of cycles
- **m** (*int*) – number of wells
- **d** (*np.ndarray*) – array of incremental increases in fluorescences, n by m
- **b** (*np.ndarray*) – array of background signals, n by m
- **dd** (*np.ndarray*) – array of standard deviation in incremental increases in fluorescences, n by m
- **db** (*np.ndarray*) – array of standard deviation in background signals, n by m
- **E_F** (*np.ndarray*) – expected value of fluorescence, n by m, $\mathbb{E}\left[\mathbf{F}\right]$
- **V_F** (*np.ndarray*) – variance of fluorescence, n by m, $\mathsf{Var}\left[\mathbf{F}\right]$
- **E_DX** (*np.array*) – expected value of change in X, length n, $\mathbf{E}\left[\Delta X_i\right]$
- **V_DX** (*np.array*) – variance value of change in X, length n, $\mathsf{Var}\left[\Delta X_i\right]$
- **cycles** (*np.array*) – cycle numbers, 1 to n

`__init__`(*C: float*, *Vol: float*, *f_plus: ndarray*, *f_minus: ndarray*, *R: float*, *pbar: float*, *E_U0: array*, *V_U0: ndarray*, *s_plus: ndarray*, *s_minus: ndarray*, ***kwargs*) → None

Parameters

- **R** (*float*) – ratio of amplification probabilities, R
- **pbar** (*float*) – geometric mean of amplification probabilities, $\bar{p}$
- **E_U0** (*np.array*) – initial expected values of both strands $\mathbb{E}\left[\mathbf{U}_0\right]$, 2 array
- **V_U0** (*np.ndarray*) – initial variances of both strands $\mathsf{Var}\left[\mathbf{U}_0\right]$, 2 by 2 matrix
- **f_plus** (*np.ndarray*) – molar fluorescences for each cycle/well of active reporter $\mathbf{f}^+$, n by number of wells matrix Determined in units of fluorescence divided by (pmol/L)
- **f_minus** (*np.ndarray*) – molar fluorescences for each cycle/well of inactive reporter $\mathbf{f}^-$, n by number of wells matrix Determined in units of fluorescence divided by (pmol/L)
- **C** (*float*) – Total concentration of reporters in pmol/L, C

- **Vol** (*float*) – Total volume of solution in L, $\mathcal{V}$
- **s_plus** (*np.ndarray*) – standard deviation in molar fluorescences for each cycle/well of active reporter σ^+, n by number of wells matrix Determined in units of fluorescence divided by (pmol/L)
- **s_minus** (*np.ndarray*) – standard deviation molar fluorescences for each cycle/well of inactive reporter σ^-, n by number of wells matrix Determined in units of fluorescence divided by (pmol/L)
- **kwargs** (*dict*) – extra kwargs for Amplification class

`calculate()`

Performs calculations, filling in `E_F`, `V_F`, `E_DX`, and `V_DX`.

`get_Cov_XiX0`(*i*)

Parameters
i (*int*) – cycle number

Returns
$\mathsf{Cov}\left[X_i, X_0\right]$, see (43)

Return type
float

`get_E_DX`(*i*) → float

Parameters
i (*int*) – cycle number

Returns
$\mathbb{E}\left[\Delta X_i\right]$

Return type
float

`get_V_DX`(*i*)

Parameters
i (*int*) – cycle number

Returns
$\mathsf{Var}\left[\Delta X_i\right]$

Return type
float

`src.kinetic_PCR.`**`plot_fluorescence_curve`**(*kls:* HydrolysisProbes, *ax: axis*, *wm1: int*)

Parameters

- **kls** (HydrolysisProbes) – Contains all amplification data. Has already computed values
- **ax** (*matplotlib.axes*) – axes to plot on
- **wm1** (*int*) – well number

CHAPTER

SIX

PLOTTING FIGURES

plot_figures.**plot_figure2**(*R=0.9*, *pbar=0.85*, *max_cycle=4*)

Parameters

- **R** (*float, optional*) – choice for value of R, defaults to 0.9
- **pbar** (*float, optional*) – choice for value of $\bar{p}$, defaults to 0.85
- **max_cycle** (*int, optional*) – choice for maximum cycle to plot, defaults to 4

plot_figures.**plot_figure6**(*model: HydrolysisProbes*, *wm1: int*)

Parameters

- **model** (HydrolysisProbes) – Contains all of the parameters for calculation of fluorescence curves
- **wm1** (*int*) – well number (0 to 95)

Notes

The methods `model.E_U0` and `model.V_U0` are updated in place during plotting of each subplot

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX