MeltR

Automated fitting of RNA/DNA absorbance melting curves and fluorescence binding isotherms in R

Jacob P. Sieg [1],[2], Philip C. Bevilacqua [1],[2],[3]

[1]Department of Chemistry, Pennsylvania State University, University Park, PA 16802.

[2]Center for RNA Molecular Biology, Pennsylvania State University, University Park, PA 16802.

[3]Department of Biochemistry and Molecular Biology, Pennsylvania State University, University Park, PA 16802.

Last update, 05/17/2022

Contents

MeltR

Automated fitting of RNA/DNA absorbance melting curves and fluorescence binding isotherms in R

Jacob P. Sieg[1],[2], Philip C. Bevilacqua[1],[2],[3]

[1]Department of Chemistry, Pennsylvania State University, University Park, PA 16802.

[2]Center for RNA Molecular Biology, Pennsylvania State University, University Park, PA 16802.

[3]Department of Biochemistry and Molecular Biology, Pennsylvania State University, University Park, PA 16802.

Last update, 05/16/2022

1 Overview

MeltR is a R package, written by Jacob Sieg, that fits nucleic acid folding data to molecular models to obtain thermodynamic parameters. MeltR automates the trivial but time-consuming tasks associated with fitting nucleic acids thermodenaturation data, leading to facile conversion of raw data into useful thermodynamic parameters.

MeltR was inspired by Meltwin.[1] MeltR and Meltwin have the same utility: easy and consistent fitting to obtain thermodynamic parameters. The main drawback of MeltR is that it is ran from your R console, whereas Meltwin has a graphical-user-interface. However, the MeltR syntax is not complicated, and MeltR has other advantages: (1) A current versions of MeltR can be downloaded from GitHub by entering two lines of code in your R console, whereas Meltwin has been out of support for years. (2) MeltR supports fitting fluorescence binding isotherms to obtain thermodynamic parameters. (3) MeltR can be ran in bulk. (4) MeltR can be run on any oporating system that is compatible with R. (5) Anecdotally, MeltR is more robust than Meltwin and requires less input from the user.

The core of MeltR is the “meltR.A” function for fitting absorbance melting curves and the “meltR.F” function for fitting fluorescence binding isotherms.

The creators of MeltR are interested in improving MeltR by adding new molecular models. Areas of interest include multistate models and models to describe G-quadruplex RNA. Please email Jacob Sieg at jus841@psu.edu for suggestions.

2 MeltR installation

MeltR is written in R. You can find instructions for installing R here: “https://www.r-project.org/”.

MeltR can be installed from “https://github.com/JPSieg/MeltR” by typing two lines into your R console.

```
install.packages("devtools")
devtools::install_github("JPSieg/MeltR")
```

Most of MeltR is written entirely in base R and requires no other dependencies. However, if you want to use the BLTrimmer (see section 4.2.7) in parallel mode, you will need to install “doParallel” and its dependencies. In your R console:

```
install.packages("doParallel")
```

Lastly, Rstudio, the tidyverse, and ggrepel, are recommended software and packages for running MeltR. You can find instructions for installing Rstudio here: “https://www.rstudio.com/products/rstudio/download/”. The free version is excellent. Tidyverse and ggrepel can be installed from your R console.

```
install.packages("tidyverse")
install.packages("ggrepel")
```

3 Theory

3.1 Fitting fluorescence isotherms

MeltR can obtains thermodynamic parameters from fluorescence binding isotherms for heteroduplex DNA and RNA using the non-linear regression function in base R, "nls". In this strategy, a quencher labeled strand is titrated into different wells in a qPCR plate containing a constant concentration of fluorophore labeled strand. The fluorophore labeled strand binds to the quencher labeled strand, reducing the fluorescence emission (E) and resulting in an apparent fluorescence binding isotherm, where the shape of the curve is determined by Equation 1.

$$E = F_{max} + (F_{min} - F_{max}) * F(K_D, [A]_T, [B]_T) \qquad (1)$$

Where F_{max} is the fluorescence emission of unbound fluorophore labeled strand (A), F\textsubscript{min is the fluorescence emission of the fluorophore labeled strand completely bound to a quencher labeled strand (B), and f(K_D, $[A]_T$, $[B]_T$) is the mole fraction of the bound fluorophore labeled strand (AB) to the total fluorophore labeled strand.

$$f(K_D, [A]_T, [B]_T) = \frac{[AB]}{[A]_T} \qquad (2)$$

f(K_D, $[A]_T$, $[B]_T$) is a function of the total fluorophore labeled strand ($[A]_T$) and the total quenceher labeled strand ($[B]_T$), and the K_D is given by the the expression:

$$K_D = \frac{[A][B]}{[AB]} \qquad (3)$$

F(K_D, $[A]_T$, $[B]_T$) can be determined by solving the K_D expression to obtain:

$$F(K_D, [A]_T, [B]_T) = \frac{(K_D + [A]_T + [B]_T) - \sqrt{(K_D + [A]_T + B]_T)^2 - 4[A]_T[B]_T}}{2[A]_T} \qquad (4)$$

Thus, the K_D at each temperature was determined by fitting isotherms at each temperature to Equation 5, obtained by plugging Equation 4 into Equation 1.

$$E = F_{max} + (F_{min} - F_{max}) \frac{(K_D + [A]_T + [B]_T) - \sqrt{(K_D + [A]_T + B]_T)^2 - 4[A]_T[B]_T}}{2[A]_T} \qquad (5)$$

Thermodynamic parameters for helix formation were extracted by the Van't Hoff relationship (Equation 6) between the K_D and the temperature using the two methods described below.

$$ln(K_D) = \frac{dS}{R} - \frac{dH}{RT} \qquad (6)$$

The dS is the entropy of helix formation, the dH is enthalpy of helix formation, R is the gas constant, and T is the temperature in Kelvin. For a single experiment, all samples should be diluted from the same fluorophore and quencher stocks, so that MeltR can deal with systematic uncertainty in strand concentrations.

3.1.1 Fluorescence data preprocessing

MeltR performs the following data preprocessing steps before fitting:

1.) The fluorophore labeled strand concentration is optimized using the concentration optimization algorithm, described in more detail in section 3.1.3.

2.) The T_m of each sample is estimated using the first derivative of fluorescence emission as a function of temperature. First derivatives are calculated using polynomial regression, where the data are approximated with a 20th order polynomial. Then, the analytical first derivative of the polynomial was determined using calculus. The approximate T_m (where 50% of the nucleic acid is single stranded) is calculated by finding the maximum of the first derivative curve to a precision of less than 0.1 °C. This T_m is unreliable because fluorescence baselines vary unpredictable with temperature. However, the T_m is useful for qualitative comparison of stability between conditions.

2.) Isotherms are fit to Equation 5 to determine K_D and error in the K_D at each temperature using "nls" in base R. Initial values for F_{max} and F_{min} are estimated by taking the mean of the 20% highest readings in each isotherm and the 20% lowest readings respectively. Initial values for the K_D are provided by the user, by default 0.1 nM.

3.) K_Ds are filtered by magnitude and error, according to user specifications, to determine which isotherms are most reliable. First, K_Ds outside of a user specified range (10 to 1000 nM by default) are thrown out. Second, K_Ds are ranked by the error in the K_D. K_Ds that are below a user specified error quantile are thrown out. The default K_D error quantile file is 0.25, meaning the algorithm will keep the to 25% most accurate K_Ds, after filtering by magnitude.

4.) The most reliable K_Ds and temperatures are passed to Method 1 to make Van't Hoff plots and to calculate helix formation energies.

5.) Fluorescence data from the most reliable K_Ds and temperatures are passed to Method 2 for global fitting and to calculate helix formation energies.

3.1.2 Concentration optimization algorithm.

We have determined that the accuracy of fit results is highly dependent on errors in the predicted concentration of fluorophore and quencher RNA strands in stock solutions. Fit accuracy is dependent on the mole ratio of fluorophore and quencher labeled RNA in stock solutions, but not dependent on the total magnitude of both fluorophore and quencher labeled RNA concentrations in the stocks. Thus, the concentration optimization algorithm in MeltR does not need to find exact concentrations to generate accurate helix formation energies, just the mole ratio (X) of fluorophore and quencher labeled RNA in samples that are predicted to have equal concentrations of fluorophore and quencher, for example predicted 200 nM FAM-RNA and 200 nM RNA-BHQ1.

MeltR uses a fluorescence binding isotherm from a temperature where the K_D is more than 10 times less than the fluorophore labeled strand concentration. Under these conditions, the shape of the curve will be independent of the K_D , and the isotherm can be used as a Job plot. For example, at a 200 nM predicted fluorophore labeled strand concentration, the shape of the binding curve will be independent of K_D if the K_D is less than 10 nM. The curve will resemble a hockey stick (Figure 1A) composed of two straight lines. The first line, where $[A]_T > [B]_T$, will decrease as the $[B]_T$ is increased and the fluorophore labeled strand is saturated. The second line, where $[A]_T < [B]T_T$, will be horizontal as the $[B]_T$ is increased because the fluorophore labeled strand is saturated. The intersection of the first and second line will occur at:

$$[A]_T = [B]_T \qquad (7)$$

The absolute concentration of $[A]_T$ and $[B]_T$ cannot be known with precicion. However, the mole ratio of the error (X) in the fluorophore and quencher stocks can be estimated from this data point at Equation 8.

$$\frac{[A]_{T-estimated}}{X} = [B]_T \qquad (8)$$

Where $[A]_{\text{T-estimated}}$ is the estimated total A concentration, or the predicted concentration based on of the stock, and X is given by Equation 9 based on the actual concentration determined from the experiment.

$$X = \frac{[A]_T}{[B]_T} \qquad (9)$$

MeltR fits an overdetermined isotherm to a binding curve (selected by the user but by default the isotherm collected at the lowest temperature) to a modified version of Equation 5 to determine X (Figure 1 B).

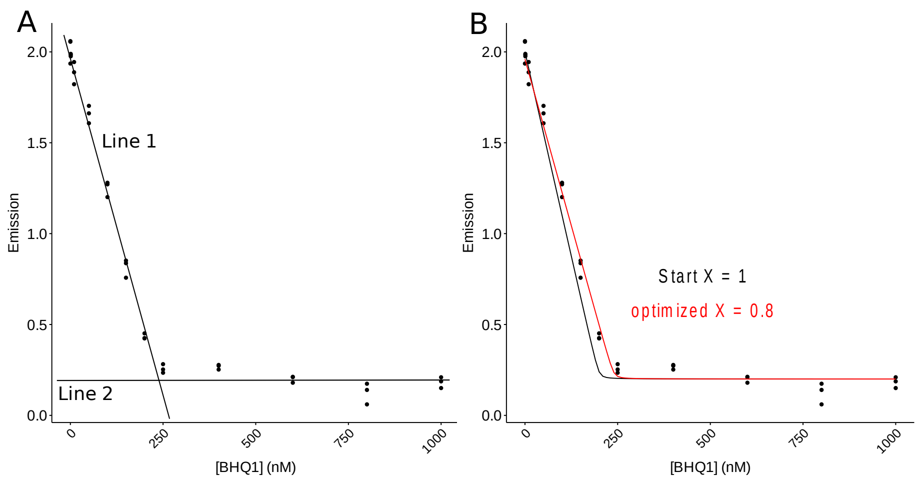


Figure 1: Job plot used by the concentration algorithm. Data are modeled with a KD of 0.1 nM and a FAM-RNA strand concentration of 250 nM. (A) Isotherms resemble a job plot, where the intersection of Line 1 and Line 2 can be used to determine X. (B) The same modeled data with the concentration algorithm starting at X = 1 and ending at X = 0.8.

$$E = F_{max} + (F_{min} - F_{max}) * \frac{(K_D + [A]_{T-estimated} + [B]_T X) - \sqrt{(K_D + [A]_{T-estimated} + B]_T X)^2 - 4[A]_{T-estim}}}{2[A]_{T-estimated}}$$

By default, the fit to determine X allows X and K_D to float. The user should also use an argument called "low_K", to set the Kd in the optimization fit to several K_D that are more than 10 times less than the fluorophore labeled strand concentrations, as described in Section 4.1.3. The user should then inspect X from several iterations of the optimization algorithm set to different "low_K" values to make sure it is similar to the iteration that allows the K_D to float.

$[A]_T$ is then corrected with Equation 11.

$$[A]_T = \frac{[A]_{T-estimated}}{X} \qquad (11)$$

3.1.3 Method 1 Van't Hoff plot

Method 1 fits K_Ds that were passed from the preprocessing steps to Equation 6. Initial estimates for the dH and dS of helix formation are provided by the user in kcal as a list using the "vh_start" argument. The default is -70 kcal/mol and -0.180 kcal/mol/K for the enthalpy and entropy respectively, and will work for most helices. The free energy of helix formation at 37 °C (dG) was calculated from the dH and dS values provided by the fit (Equation 12).

$$dG = dH - 310.15 * dS \qquad (12)$$

The standard error of the dG was propagated from the standard error of dH and the dS from the fit, and the covariation between the two values.

$$SE_{dG} = \sqrt{{SE_{dH}}^2 + (310.15 * \frac{SE_{dS}}{dS})^2 + 2 * 310.15 * \frac{Covar_{dH,dS}}{dH * dS}} \qquad (13)$$

Helix formation energies are traditionally reported in terms of the association constant.

$$K = \frac{[AB]}{[A][B]} = frac1K_D \qquad (14)$$

The dHs and dSs reported by MeltR are also reported in terms of the association constant, which is obtained by multiplying helix formation energies from fitting K_D by negative one.

3.1.4 Method 2 Global fit

Method 2 globally fits fluorescence data that were passed from the preprocessing to Equation 15, obtained by plugging Equation 6 into Equation 5. F_{max} and F_{min} is allowed to float between temperatures and dH and dS are fixed between temperatures. Starting values for F_{max}, F_{min}, dH, and dS are obtained from the results for individual fits.

$$E = F_{max} + (F_{min} - F_{max}) * \frac{(\exp{(\frac{dS}{R} - \frac{dH}{RT})} + [A]_T + [B]_T) - \sqrt{(\exp{(\frac{dS}{R} - \frac{dH}{RT})} + [A]_T + B]_T)^2 - 4[A]_T[B]_T}}{2[A]_T}$$

The dG and the error in the dG were calculated as in method 1.

3.2 Fitting abosorbance melting curves

3.2.1 Absorbance data preprocessing

MeltR can obtain thermodynamic parameters from absorbance melting curves for heteroduplex, homoduplex (selfcomplementary), and monomolecular self-structured DNA and RNA using the method pioneered by Tinoco and colleagues and refined by Turner and colleagues.[1-3]

MeltR performs the following data preprocessing steps before fitting:

1.) RNA and DNA extinction coefficients are calculated from the sequence using the extinction coefficient data from Tinoco and colleagues.[2]

2.) The background absorbance of a user specified blank is scaled to the pathlength of the cuvette and subtracted from each curve.

3.) The total strand concentration (C_t) is calculated at a user specified temperature (Default = 90 °C). For homoduplexes and monomolecular-self structured RNA, where there is only one nucleic acid in the sample, the C_t is the total concentration of the single strand. For heteroduplexes, the C_t is the total concentration of strand 1 plus the total concentration of strand 2.

4.) High and low temperature data are trimmed (to the users specification) to ensure linear baselines.

5.) First and second derivatives are taken using polynomial regression. First, the data are approximated using a 20th order polynomial. Then, the first and second derivatives of the polynomial are determined analytically using calculus. The approximate $T_{0.5}$ (the approximate melting temperature, T_m, where 50% of the nucleic acid is single stranded) is calculated by finding the maximum of the first derivative curve and the $T_{0.75}$ (the approximate temperature where 75% of the nucleic acid is single stranded) is calculated by finding the minimum of the second derivative (Figure 2), to a precision of less than 0.1 °C.

6.) Initial parameter estimates are calculated for each curve. The initial values for slopes and intercepts of the baselines are estimated by fitting absorbance values that are greater than the 75th quantile for the uper baselines, and fitting aborbance values that are lower than the 25th quantile for the lower baseline, to y = mx + b. Initial values for the enthalpy are determined using the $T_{0.5}$ and $T_{0.75}$ (in Kelvin) from first and second derivative curves. MeltR uses equation 16, equation 17, and equation 18, for heteroduplex, homoduplex, and monomolecular self-structured DNA and RNA melting curves.

$$dH = -0.007 * (\frac{1}{T_{0.5}} - \frac{1}{T_{0.75}}) \qquad (16)$$

$$dH = -0.0044 * (\frac{1}{T_{0.5}} - \frac{1}{T_{0.75}}) \qquad (17)$$

$$dH = -0.0032 * (\frac{1}{T_{0.5}} - \frac{1}{T_{0.75}}) \qquad (18)$$

The initial T_m is estimated from the $T_{0.5}$, determined in step 5.

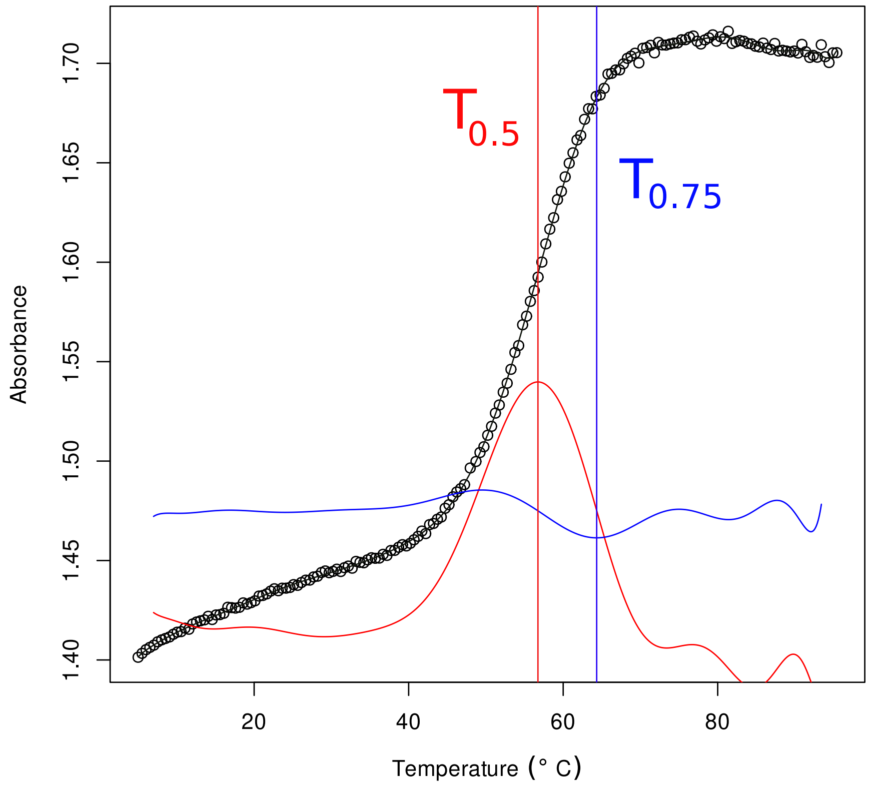


Figure 2: Absorbance melting curve derivative determined with polynomial regression.

3.2.2 Thermodynamic models for duplex formation

Thermo-dynamic parameters for helix formation are obtained using a Van't Hoff model:

$$lnK = \frac{dS}{R} - \frac{dH}{RT} \qquad (19)$$

where dS is the entropy change, dH is the enthalpy change, R is the gas constant in kcal/mol, T is the temperature in Kelvin, and K is the equillibrium constant given by Equation 20, Equation 21, and Equation 22 for heteroduplexes, homoduplexes, and monomolecular self-structured RNA respectively.

$$K = \frac{[AB]}{[A][B]} \qquad (20)$$

$$K = \frac{[AA]}{[A]^2} \qquad (21)$$

$$K = \frac{[F]}{[U]} \qquad (22)$$

For Equation 20 and Equation 21, [A] and [B] are the concentration of different strands, [AB] is the concentration of strand A in a duplex with strand B, and [AA] is the concentration of a self complementary strand A in a duplex with another self complementary strand A. For Equation 22, [F] is the concentration of a monomolecular self-structured RNA in the folded state and [U] is the concentration of a monomolecular self-structured RNA in the unfolded state.

MeltR uses three methods based on the Van't Hoff equation to calculate thermodynamic parameters: (1) fitting melting curves individually, (2) fitting the thermodenaturation point as a function of temperature, and (3) Global fitting melting curves.

3.2.3 Method 1 fitting melting curves individually Method 1 fits the absorption as a function of temperature for each sample individually. Base lines are modeled as y= mx + b for the absorbance of the unfolded-single stranded and folded-duplex states (Equation 23).

$$A = mT + b \qquad (23)$$

The absorption of each sample as a function of temperature is also a function of the fraction of RNA in the folded-duplex state (DS), as a function of temperature f(T), given by Equation 24. SS represents the RNA in the single-stranded state.

$$A = (m_{DS}T + b_{DS})f(T) + (m_{SS}T + b_{SS})(1 - f(T)) \qquad (24)$$

f(T) is variable, calculated by the analytic solution of the binding constant. MeltR uses Equation 25 for heteroduplexes, Equation 26 for homoduplexes, and Equation 27 monomolecular self-structured RNA.

$$f(T) = \frac{\frac{2}{K(T)*Ct} + 2 - \sqrt{(\frac{2}{K(T)*Ct} + 2)^2 - 4}}{2} \qquad (25)$$

$$f(T) = \frac{\frac{1}{2*K(T)*Ct} + 2 - \sqrt{(\frac{1}{2*K(T)*Ct} + 2)^2 - 4}}{2} \qquad (26)$$

$$f(T) = \frac{K(T)}{1 + K(T)} \qquad (27)$$

Where C_t is the total strand concentration. K(T) is the equilibrium constant as a function of temperature, given by Equation 28 for heteroduplexes, Equation 29 for homoduplexes, and Equation 30 for monomolecular self-structured RNA.

$$K(T) = \exp(\frac{dH}{R}(\frac{1}{T_m} - \frac{1}{T}) + ln(\frac{4}{Ct})) \qquad (28)$$

$$K(T) = \exp(\frac{dH}{R}(\frac{1}{T_m} - \frac{1}{T}) + ln(\frac{1}{Ct})) \qquad (29)$$

$$K(T) = \exp(\frac{dH}{R}(\frac{1}{T_m} - \frac{1}{T})) \qquad (30)$$

Note, K(T) is in terms of T_m and C_t, instead of the dS, to increase the ease of estimating initial parameters for non-linear regression and to increase the robustness of the nls algorithm. Briefly, the dS is replaced with T_m and C_t by solving Equation 19 for the dS at the T_m, where f(T) = 0.5.

Thus, method 1 fits absorbtion versus temperature for each sample to equations 31, 32, and 33 to determine thermodynamic prameters for heteroduplexes, homoduplexes, and monomolecular self-structured RNA respectively.

$$E = (m_{DS}T + b_{DS})\frac{\frac{2}{\exp(\frac{dH}{R}(\frac{1}{T_m} - \frac{1}{T}) + ln(\frac{4}{Ct}))*Ct} + 2 - \sqrt{(\frac{2}{\exp(\frac{dH}{R}(\frac{1}{T_m} - \frac{1}{T}) + ln(\frac{4}{Ct}))*Ct} + 2)^2 - 4}}{2} + (m_{SS}T + b_{SS})(1 -$$

$$E = (m_{DS}T+b_{DS})\frac{\frac{1}{2*\exp\left(\frac{dH}{R}(\frac{1}{T_m}-\frac{1}{T})+ln(\frac{1}{Ct})\right)*Ct}+2-\sqrt{(\frac{1}{2*\exp\left(\frac{dH}{R}(\frac{1}{T_m}-\frac{1}{T})+ln(\frac{1}{Ct})\right)*Ct}+2)^2-4}}{2}+(m_{SS}T+b_{SS})($$

$$E = (m_{DS}T+b_{DS})\frac{\exp\left(\frac{dH}{R}(\frac{1}{T_m}-\frac{1}{T})\right)}{1+\exp\left(\frac{dH}{R}(\frac{1}{T_m}-\frac{1}{T})\right)}+(m_{SS}T+b_{SS})(1-\frac{\exp\left(\frac{dH}{R}(\frac{1}{T_m}-\frac{1}{T})\right)}{1+\exp\left(\frac{dH}{R}(\frac{1}{T_m}-\frac{1}{T})\right)}) \qquad (33)$$

Free energy at 37 °C (dG) is calculated from the dH and entropy (dS) of helix formation

$$dG = dH - 310.15 * dS \qquad (34)$$

The dS of helix formation is calculated from the dH and the Tm.

For heteroduplexes:

$$dS = \frac{dH}{Tm} + Rln(\frac{4}{Ct}) \qquad (35)$$

For homoduplexes:

$$dS = \frac{dH}{Tm} + Rln(\frac{1}{Ct}) \qquad (36)$$

For monomolecular self-structured RNA/DNA:

$$dS = \frac{dH}{Tm} \qquad (37)$$

Error in the dS and dG is calculated by propagating error in the fit terms dH and T_m.

$$SE_{dS} = |dS|\sqrt{(\frac{SE_{dH}}{dH})^2 + (\frac{SE_{Tm}}{Tm})^2 - 2\frac{Covar_{dH,Tm}}{dH*Tm}} \qquad (38)$$

$$SE_{dG} = \sqrt{{SE_{dH}}^2 + (310.15*\frac{SE_{dH}}{dH})^2 + (310.15*\frac{SE_{Tm}}{Tm})^2 - 2*310.15\frac{Covar_{dH,Tm}}{dH*Tm}} \qquad (39)$$

3.2.4 Method 2 fitting the $\mathrm{T_m}$ as a function of $\mathrm{C_t}$

Method 2 fits the relationship between $1/\mathrm{T_m}$ and the total strand concentration $\mathrm{C_t}$. To avoid inaccuracies in $\mathrm{T_m}$ determination from first derivative plots or covariation with the dH terms in method 1, $\mathrm{T_m}$s were determined for Method 2 using a semi-quantitative method. Slopes and intercepts from method 1 were used to calculate f(T) at each experimental temperature using the absorbance.

$$f(T) = \frac{A - (m_{SS}T + b_{SS})}{(m_{DS}T + b_{DS}) + (m_{SS}T + b_{SS})} \qquad (40)$$

F(T) is approimatelty linear in the range of 0.4 to 0.6. Thus, F(T in {0.4 to 0.6}) was fit with y = mT + b, and solved using y = 0.5 to accurately determine the melting temperature for each $\mathrm{C_t}$. To determine thermodynamic parameters, the relationship between $1/\mathrm{T_m}$ and the total strand concentration was then fit to Equations 41, and 42, for heteroduplexes and homoduplexes respectively. The $\mathrm{T_m}$ of monomolecular, self-structured RNA is independent of $\mathrm{C_t}$ so Method 2 cannot be used.

$$\frac{1}{T_m} = \frac{R}{dH} * lnC_t + \frac{dS}{dH} - R * ln(4) \qquad (41)$$

$$\frac{1}{T_m} = \frac{R}{dH} * lnC_t + \frac{dS}{dH} \qquad (42)$$

Free energy at 37 °C (dG) is calculated from the dH and entropy (dS) of helix formation directly from the fit.

$$dG = dH - 310.15 * dS \qquad (43)$$

Error in the dG is calculated by propagating error in the fit terms dH and dS.

$$SE_{dG} = \sqrt{{SE_{dH}}^2 + (310.15 * \frac{SE_{dS}}{dS})^2 - 2 * 310.15 * \frac{Covar_{dH,dS}}{dH * dS}} \qquad (44)$$

3.2.5 Method 3 Global fitting

Method 3 fits all curves to Equations 45, 46, and 47, simultaneously in a global fit, for heteroduplexes, homoduplexes, and mono-molecular self-structured RNA/DNA respectively. In this global fit, Equations 31, 32, and 33, are rearranged to be in terms of the dS instead of the Tm.

$$E = (m_{DS}T+b_{DS}) \frac{\frac{2}{\exp{(\frac{dS}{R}-\frac{dH}{RT})}*Ct} + 2 - \sqrt{(\frac{2}{\exp{(\frac{dS}{R}-\frac{dH}{RT})}*Ct} + 2)^2 - 4}}{2} + (m_{SS}T+b_{SS})(1- \frac{\frac{2}{\exp{(\frac{dS}{R}-\frac{dH}{RT})}*Ct} + 2 -}{}$$

$$E = (m_{DS}T + b_{DS}) \frac{\frac{1}{2\exp(\frac{dS}{R} - \frac{dH}{RT}) * Ct} + 2 - \sqrt{(\frac{1}{2*\exp(\frac{dS}{R} - \frac{dH}{RT}) * Ct} + 2)^2 - 4}}{2} + (m_{SS}T + b_{SS})(1 - \frac{\frac{1}{2*\exp(\frac{dS}{R} - \frac{dH}{RT}) * Ct} +}{}$$

$$E = (m_{DS}T + b_{DS}) \frac{\exp(\frac{H}{R*Tm} - \frac{1}{Tm})}{1 + \exp(\frac{dS}{R} - \frac{dH}{RT})} + (m_{SS}T + b_{SS})(1 - \frac{\exp(\frac{dS}{R} - \frac{dH}{RT})}{1 + \exp(\frac{dS}{R} - \frac{dH}{RT})}) \qquad (47)$$

The baselines are allowed to vary but dHs and dSs are constrained to a single value for all curves. For global fitting, the slopes and intercepts of the fits from Method 1 are used as initial parameter estimates for the slopes and intercepts of the global fit, and the average of the dHs and dSs from Method 1 are used as initial parameter estimates for the dH and dS.

The dG and error in the dG is calculated using the same equations as Method 2.

4 Running MeltR

In this section, we cover how to use MeltR in your R console. If you have not already, read section 3 to understand the theory underlying the results of MeltR. Section 4 will cover MeltR usage and how to avoid pitfalls. The most common error with MeltR is a user attempting to fit data that is inconsistent with the underlying model, either a fluorescence isotherm or a absorbance melting curve with a non-standard shape or specifying an incorrect molecular model. In the case of data with a non-standard shape, the aberrant data will need to be filtered out data set prior to fitting the set with MeltR. While MeltR has no dependencies other than base R 4.1.3, data wrangling and plotting functions in the "tidyverse" package[4] are highly recommended, along with the "ggrepel" package[5], for data quality checks and filtering. To begin, open a new R session in the proper directory. Load relevant packages into your memory.

```
library(MeltR)
library(tidyverse)
library(ggrepel)
```

Help documentation for MeltR functions can be pulled up using standard R commands.

```
?meltR.F
?meltR.A
```

4.1 Fitting fluorescence binding isotherms

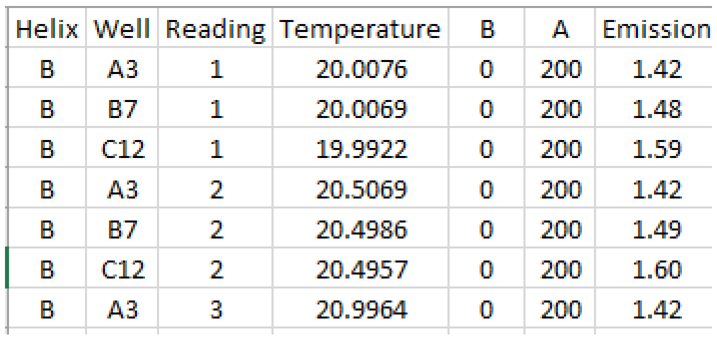
4.1.1 Formatting fluorescence data for MeltR

Data should be formatted into a comma separated value (".csv") text file with carefully labeled columns (Figure 3). There should be a "Well" column where numbers or character strings specify what well in a microplate the data came from, a "Reading" column that specifies the reading a data point comes from, a "Temperature" column that specifies the temperature where the data was recorded, a "B" column which specifies an approximate quencher labeled strand concentration in nM, an "A" column which specifies an approximate fluorophore labeled strand concentration in nM, and an "Emission" column containing the fluorescence emission intensity.

Two common pitfalls occur when formatting data for MeltR, usually in Excel. The first is incorrect column names, even incorporation of an extra space, so that MeltR cannot recognize relevant data when it is read into R. The second is incorporation of characters into data columns when values are missing. If a data point is missing, leave the cell blank. Do not write something like "NA".

4.1.2 Reading data into a R data frame

Comma separated value (".csv") data can be read into a R data frame for MeltR using the "read.csv" function that is included in base R.



Helix	Well	Reading	Temperature	B	A	Emission
B	A3	1	20.0076	0	200	1.42
B	B7	1	20.0069	0	200	1.48
B	C12	1	19.9922	0	200	1.59
B	A3	2	20.5069	0	200	1.42
B	B7	2	20.4986	0	200	1.49
B	C12	2	20.4957	0	200	1.60
B	A3	3	20.9964	0	200	1.42

Figure 3: Formatting fluorescence binding isotherm datat in a csv file for MeltR with Excel. B and A are the concentration of quencher and fluorophore labeled RNA strands, respectively, in nM. Temperature is in °C.

```
df = read.csv("path/file_name.csv")
```

Data can be checked with the "View" function.

```
View(df)
```

Note, the columns in "df" must be named correctly. If the columns are not named correctly, MeltR cannot recognize them. You can rename the columns of a data frame using:

```
colnames(df) = c("Helix", "Well", "Reading", "Temperature", "B", "A", "Emission")
```

4.1.3 Applying "meltR.F" to obtain thermodynamic parameters

For this tutorial, we will use sample data included in MeltR. First check that MeltR is in your memory and find out what is in the sample data.

```
?df.fluor.data
df = df.fluor.data
```

In general, it is a good idea to plot your data before you start fitting. First, I want to check the quality of low temperature isotherms. Using the tidyverse and ggrepel.

```
head(df)
ggplot(df %>% filter(Reading == 1), aes(x = B, y = Emission, label = Well)) +
  geom_point() +
  geom_text_repel()
```

The result is a really nice binding isotherm (Figure 4). One should check a few more readings by changing the value in the filter command, for example 20, 60, 80, etc. . .

If you observe one or two obvious outliers in the data set, it is reasonable to

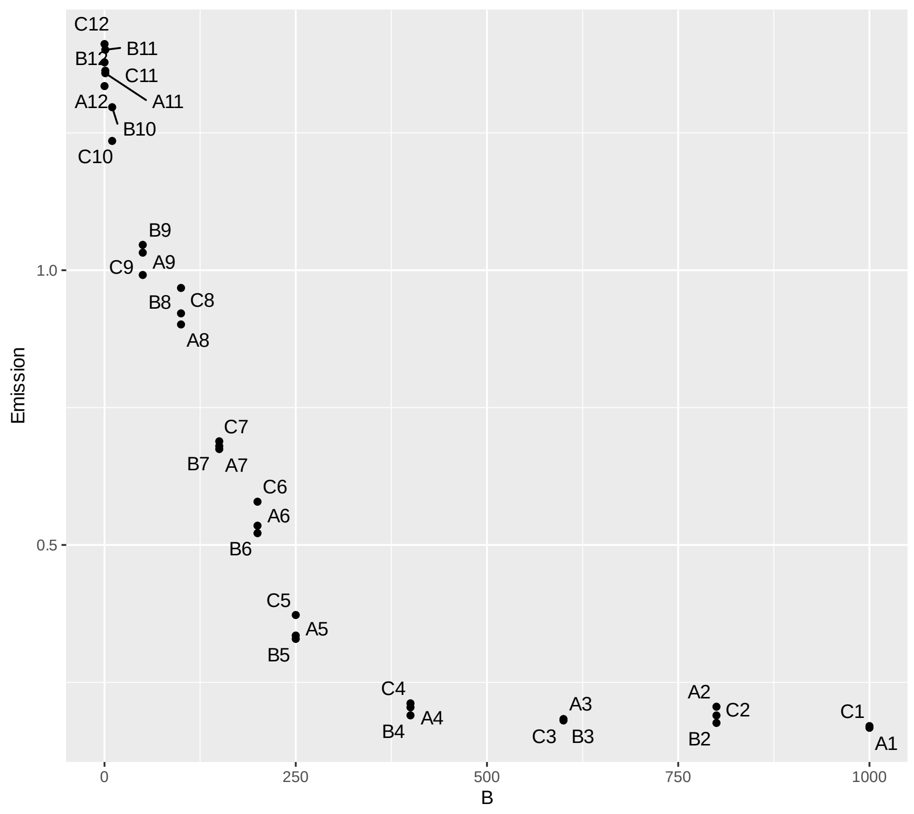


Figure 4: Fluorescence isotherm example from data included in MeltR. Labels represent the well in a 96 well plate.

remove them using the filter command. However, this data is excellent and needs no filtering.

```
df %>% filter(!Well %in% c("A1", "C3"))
```

If you are satisfied with the the data, we can move on to fitting. I will first inspect the help file for meltR.F.

```
?meltR.F
```

The only argument that needs set is the data frame.

```
meltR.F(df)
```

You will see the following output in your console.

```
[1] "Van't Hoff"
[1] "accurate Ks = 16"
        Method         H     SE.H         S      SE.S         G       SE.G
1    1 VH plot -48.37473 1.048500 -121.6157  3.357021 -10.65562 0.01033199
2 2 Global fit -48.88546 6.302188 -123.2438 20.157522 -10.66140 0.06564137
    K_error         R   Kd.opt
1 0.2916302 0.7797618 6.165553
2 0.2916302 0.7797618 6.165553
[1] "Fractional error between methods"
          H         S          G
1 0.01050236 0.013298 0.000542719
```

Note, the concentration optimization algorithm, by default allowing the Kd.opt to float, identified an optimal R of ~0.78. Since the FAM-RNA strand in each well is estimated at 200 nM, the concentration optimization algorithm will adjust the concentration to 200/0.78 = 256 nM. We will test the robustness of this estimate by constraining the Kd.opt to a range of K_Ds that are more than 10 times less than the FAM-RNA concentration, using the "low_K" argument. For example, 1, 0.5, 0.1, and 0.05 nM. Note, the mole ratio "R" was labeled "X" in the theory section to avoid confusion with the gas constant.

```
> meltR.F(df, low_K = 1)
[1] "Van't Hoff"
[1] "accurate Ks = 11"
        Method         H       SE.H         S      SE.S         G        SE.G
1    1 VH plot -60.55018  0.7291223 -160.1005  2.322061 -10.89501 0.009578498
2 2 Global fit -60.70845 12.4962966 -160.6024 39.782555 -10.89761 0.167951340
    K_error         R Kd.opt
1 0.3333168 0.7231654      1
2 0.3333168 0.7231654      1
[1] "Fractional error between methods"
            H           S            G
1 0.002610529 0.003130137 0.0002389311
> meltR.F(df, low_K = 0.5)
```

```
[1] "Van't Hoff"
[1] "Van't Hoff"
[1] "accurate Ks = 11"
        Method         H       SE.H         S      SE.S         G        SE.G
1    1 VH plot -61.48056  0.7017499 -163.0171  2.234887 -10.92082 0.009218896
2 2 Global fit -61.64895 12.7501200 -163.5512 40.589141 -10.92354 0.171775450
    K_error         R Kd.opt
1 0.3374478 0.7153701    0.5
2 0.3374478 0.7153701    0.5
[1] "Fractional error between methods"
           H           S            G
1 0.00273514 0.003271292 0.0002491787
> meltR.F(df, low_K = 0.1)
[1] "Van't Hoff"
[1] "accurate Ks = 14"
[1] "Van't Hoff"
[1] "accurate Ks = 10"
        Method         H       SE.H         S      SE.S         G        SE.G
1    1 VH plot -63.04998  0.6809662 -167.9641  2.167052 -10.95592 0.009325515
2 2 Global fit -63.18785 14.8873698 -168.4012 47.361747 -10.95822 0.207758608
    K_error         R Kd.opt
1 0.3446286 0.7080886    0.1
2 0.3446286 0.7080886    0.1
[1] "Fractional error between methods"
            H           S            G
1 0.002184357 0.002599086 0.0002100154
> meltR.F(df, low_K = 0.05)
        Method         H      SE.H         S      SE.S         G        SE.G
1    1 VH plot -63.17200  0.677299 -168.3465  2.155382 -10.95934 0.009275292
2 2 Global fit -63.31084 14.925896 -168.7867 47.484124 -10.96165 0.208349999
    K_error         R Kd.opt
1 0.3451437 0.7070892   0.05
2 0.3451437 0.7070892   0.05
[1] "Fractional error between methods"
            H           S            G
1 0.002195498 0.002611534 0.0002110328
```

The optimization algorithm, using a constrained K_D, adjusts the FAM-RNA concentration to at most 281, about 10% difference from the optimization algorithm using a floating K_D. Thus, the default concentration optimization is robust. Note that there is considerable variance in the enthalpies, entropies, and free energies from the various constrained K_Ds and the unconstrained fit (about 20 kcal/mole in terms of the enthalpy). We will demonstrate how to refine these thermodynamic parameters independent of the concentration optimization algorithm in section 4.1.5.

4.1.4 Saving meltR.F outputs

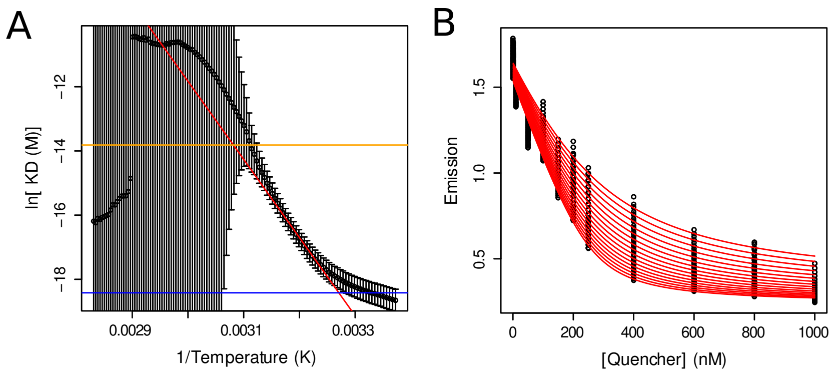
meltR.F results can be saved to the disk using the “Save_results” argument.

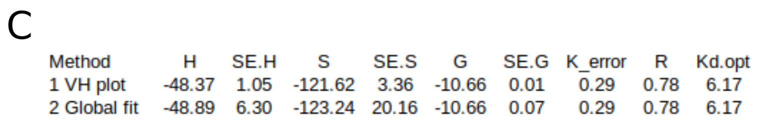
```
meltR.F(df,
        Save_results = "all")
```

The “file_prefix” argument can be used to add a custom file name to the outputs.

```
meltR.F(df,
        Save_results = "all",
        file_prefix = "Helix_J")
```

This will create three pre-canned outputs. The first output, corresponding to Method 1, is a Van’t Hoff plot (Figure 5A). Points represent the K_D and error from fitting isotherms individually. The red line represents the fit to Equation 6 that provides thermodynamic parameters. The blue line and orange line represents the lower and upper limit of the range of K_D values included in the fit. The second output, corresponding to Method 2, is a depiction of the global fit, where points represent raw data and red lines represent the global fit (Figure 5B). The third output is a .csv file containing the thermodynamic parameters from each method Figure 5 C. Note that the fit line (red) in Figure 5A does not follow the trend at high temperatures and the helix folding energies from this line are thus unreliable. We will cover refining the fit in section 4.1.5.




C

Method	H	SE.H	S	SE.S	G	SE.G	K_error	R	Kd.opt
1 VH plot	-48.37	1.05	-121.62	3.36	-10.66	0.01	0.29	0.78	6.17
2 Global fit	-48.89	6.30	-123.24	20.16	-10.66	0.07	0.29	0.78	6.17

Figure 5: Pre-canned meltR.F outputs.

4.1.5 Refining meltR.F fits

Two arguments are important for refining meltR.F fits. The first is, “Kd_range”, which is the range of K_Ds in nM that will be fit to obtain thermodynamic parameters. By default, the “Kd_range” is set to 10 to 1000. The second is, “Kd_error_quantile”, which controls the amount of error that is included in the K_Ds that will be fit to obtain thermodynamic parameters. By default, the “Kd_error_quantile” is 0.25, meaning only the 25% most accurate KDs in the “K_range” will be fit to obtain thermodynamic parameters.

As a first guess, the K_D range should start about 10 times less than the fluorophore labeled RNA strand concentration and end at about 10 times more than the fluorophore labeled RNA strand, and the “Kd_error_quantile” should be conservative, near 0.25. After this, the Van’t Hoff plot should be inspected, for how well the fit matches the linear range. In general, constraining the “Kd_range” and loostening the “Kd_error_quantile” results in the best fits. For example, a “Kd_range” of 40 to 500 nM and a “Kd_error_quantile” of 0.5 works well for the sample data (Figure 6).

```
meltR.F(df,
        Kd_range = c(40, 500),
        Kd_error_quantile = 0.5,
        Save_results = "all")
```

with an output of:

```
[1] "Van't Hoff"
[1] "accurate Ks = 18"
        Method         H     SE.H         S      SE.S         G       SE.G
1    1 VH plot -60.60235 0.964952 -160.5832  3.057177 -10.79747 0.01834264
2 2 Global fit -59.90460 6.321305 -158.3554 20.049234 -10.79066 0.11245376
   K_error         R   Kd.opt
1 6.182951 0.7797618 6.165553
2 6.182951 0.7797618 6.165553
[1] "Fractional error between methods"
           H          S            G
1 0.01158036 0.01397017 0.0006300081
>
```

Note two things: (1) the refined fit uses more isotherms than the original iterations of the program, where we were checking the concentration optimization algorithm in Section 4.1.3, with an “accurate Ks” count of 18 instead of 10 to 14. (2) The the thermodynamic parameters are closer to the fits with constrained “low_Ks”, with a difference of about 5% in terms of the enthalpy and 1% in terms of the free energy.

4.1.6 Advanced analysis of meltR.F outputs in R

meltR.F can pass a more extensive output to an object in R.

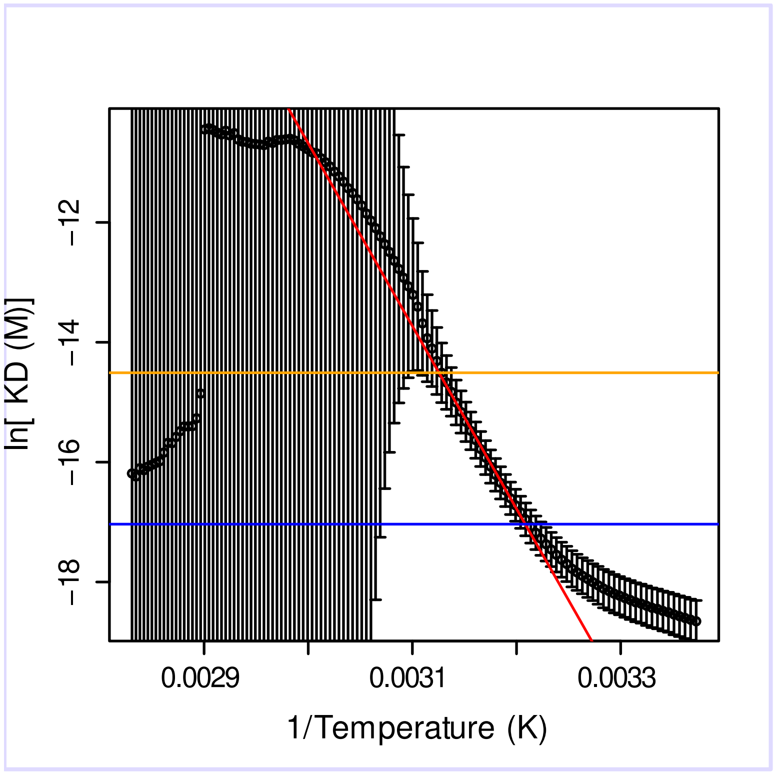


Figure 6: Refined Van't Hoff plot to determine the dH and dS.

```
MeltR.fit = meltR.F(df,
        Kd_range = c(40, 500),
        Kd_error_quantile = 0.5,
        Save_results = "all")
```

The object, "MeltR.fit" is now a list of objects that can be passed to plotting functions.

```
names(MeltR.fit)
```

Which will print the names of the list "MeltR.fit".

```
[1] "VantHoff"                          "K"
[3] "VH_method_1_fit"                   "VH_method_2_fit"
[5] "Raw_data"                          "First_derivative"
[7] "Tms"                               "R"
[9] "Fractional_error_between_methods"
```

[1] VantHoff: is a data frame containing the duflex formation energies. It can be called using:

```
MeltR.fit$VantHoff
```

[2] K: is a data frame containing the results from fitting each isotherm individually. It can be called using:

```
MeltR.fit$K
```

[3] VH_method_1_fit: is a nls object containing the fit obtained from the Van't Hoff plot. It can be called using:

```
MeltR.fit$VH_method_1_fit
```

[4] VH_method_2_fit: is a nls object containing the fit obtained from the global fit. It can be called using:

```
MeltR.fit$VH_method_2_fit
```

[5] Raw_data: The raw data passed back out of MeltR.F with no modifications. It can be called using:

```
MeltR.fit$Raw_data
```

[6] First derivative: The dirst derivative of each sample. Useful for qualitative comparison of data between conditions. It can be called using:

```
MeltR.fit$First_derivative
```

[7] Tms: The approximate T_m of each sample obtained from the maximum of the first derivative. Useful for qualitative comparison of data between solution conditions. It can be called using:

```
MeltR.fit$Tms
```

[8] R: the mole ratio of fluorophore and quencher labeled RNA, used in the concentration optimization algorithm. The mole ratio "R" was labeled "X" in the theory section to avoid confusion with the gas constant. It can be called using:

```
MeltR.fit$R
```

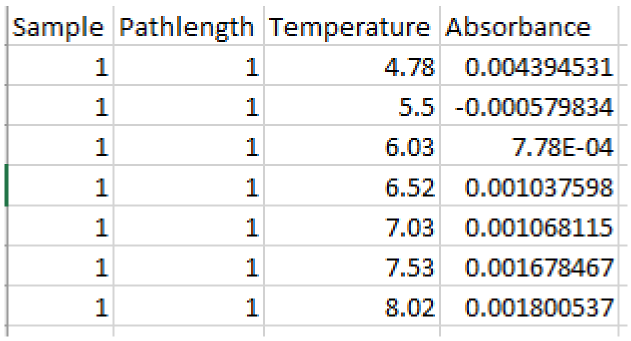
[9] Fractional error between Methods: The amount thermodynamic parameters vary between methods. It can be called using:

```
MeltR.fit$Fractional_error_between_methods
```

4.2 Fitting absorbance melting curves in MeltR

4.2.1 Formatting absorbance data for MeltR

Data should be formatted into a comma separated value (".csv") text file with carefully labeled columns (Figure 7). There should be a "Sample" column where numbers or character strings specify what sample the absorbance data was collected on. There should be one buffer blank in the data set for background subtraction. If no blank is available, add data with an absorbance of 0 recorded at each temperature. A "Pathlength" column specifies the pathlength in cm of the cuvette the data was collected in. A "Temperature" column specifies the temperature in °C where the data was recorded. Lastly, a "Absorbance" column specifies the absorbance collected at each temperature.



Sample	Pathlength	Temperature	Absorbance
1	1	4.78	0.004394531
1	1	5.5	-0.000579834
1	1	6.03	7.78E-04
1	1	6.52	0.001037598
1	1	7.03	0.001068115
1	1	7.53	0.001678467
1	1	8.02	0.001800537

Figure 7: Formatting absorbance melting curve data in a csv in Excel. Temperature is in °C and pathlength is in cm.

Two common pitfalls occur when formatting data for MeltR, usually in Excel. The first is incorrect column names, even incorporation of an extra space, so that MeltR cannot recognize data when it is read into R. The second is incorporation of characters into data columns when values are missing. If a data point is missing, leave the cell blank. Do not write something like "NA".

4.2.2 Reading data into a R data frame

Comma separated value (".csv") data can be read into a R data frame for MeltR using the "read.csv" function that is included in base R.

```
df = read.csv("path/file_name.csv")
```

Data can be checked with the "View" function.

```
View(df)
```

Note, the columns in "df" must be named correctly. If the columns are not named correctly, MeltR cannot recognize them. You can rename the columns of a data frame using:

```
colnames(df) = c("Sample", "Pathlength", "Temperature", "Absorbance")
```

4.2.3 Applying meltR.A to obtain thermodynamic parameters

Sample data is included in MeltR and can be loaded into your memory and checked.

```
?df.abs.data
df = df.abs.data
```

In general, it is a good idea to plot your data before you start fitting. First, I want to check the quality of low temperature isotherms. Using the tidyverse.

```
head(df)
ggplot(df, aes(x = Temperature, y = Absorbance, color = factor(Sample))) +
  geom_point() +
  theme_classic()
```

This data looks good (Figure 8), with a small and consistent blank absorbance and sigmoidal RNA melting curves. If one of the RNA melting curves does not resemble a sigmoid, it should be removed from the data set using:

```
df %>% filter(Sample != "Sample you want to remove")
```

Data can now be fit using meltR.A. First we need to define the nucleic acid sequence using a vector that specifies the RNA or DNA sequences used in the experiment.

```
helix = c("RNA", "CGAAAGGU", "ACCUUUCG")
```

meltR.A requires information from 4 user defined arguments to fit the data: the data frame the absorbance data is stored in, the blank sample, the nucleic acid sequences, and the molecular model. Three molecular models are availible in MeltR, "Monomolecular.2State", "Heteroduplex.2State", and "Homoduplex.2State". The absorbance sample data included in MeltR was collected on a heteroduplex.

```
?meltR.A
meltR.A(data_frame = df,
```

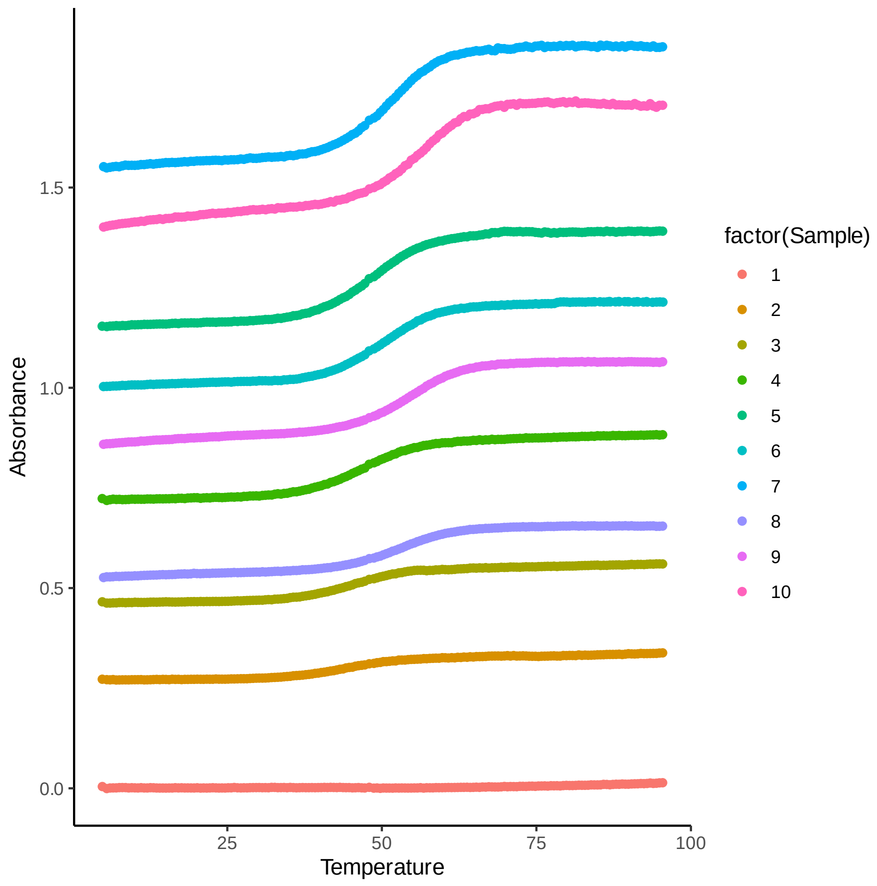


Figure 8: Absorbance data check.

```
        blank = 1,
        NucAcid = helix,
        Mmodel = "Heteroduplex.2State")
```

The output looks like this.

```
[1] "Individual curves"
  Sample           Ct     H       S     G   Tm
1      2 2.832690e-06 -63.5 -0.1725  -9.9 43.1
2      3 4.718659e-06 -71.2 -0.1969 -10.1 44.6
3      4 7.439046e-06 -64.8 -0.1764 -10.0 46.4
4      5 1.173670e-05 -64.5 -0.1752 -10.1 48.4
5      6 2.049582e-05 -70.9 -0.1952 -10.4 50.0
6      7 3.131297e-05 -74.3 -0.2055 -10.5 51.4
7      8 5.533500e-05 -71.6 -0.1974 -10.4 53.0
8      9 8.990554e-05 -74.8 -0.2069 -10.7 54.8
9     10 1.439218e-04 -76.7 -0.2121 -11.0 57.0
[1] "Summary"
             Method     H SE.H      S SE.S     G SE.G
1  1 individual fits -70.3  4.9 -193.1 14.9 -10.4  0.3
2 2 Tm versus ln[Ct] -60.4  1.2 -163.0  3.8  -9.9  0.1
3       3 Global fit -69.7  0.3 -191.3  1.0 -10.4  0.0
[1] "fractional error between methods"
          H         S          G
1 0.1482036 0.1649616 0.04885993
```

Note that the fractional error between the methods is high, over about 15% in terms of the enthalpy. Section 4.2.4 discusses how to refine the fits by trimming the absorbance baselines.

4.2.4 Saving meltR.A outputs

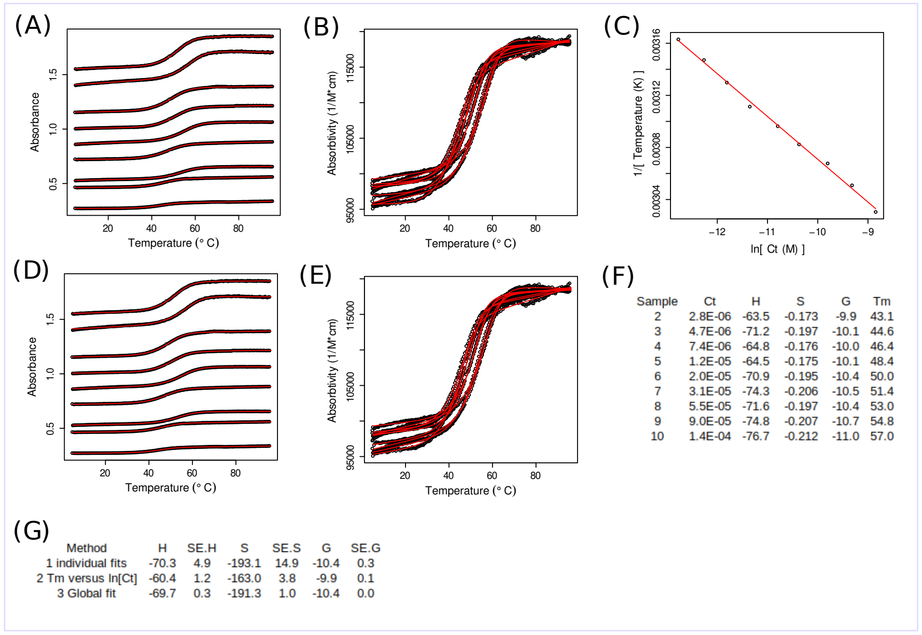
meltR.F results can be saved to the disk using the "Save_results" argument.

```
meltR.A(data_frame = df,
        blank = 1,
        NucAcid = helix,
        Mmodel = "Heteroduplex.2State",
        Save_results = "all")
```

The "file_prefix" argument can be used to add a custom file name to the outputs

```
meltR.A(data_frame = df,
        blank = 1,
        NucAcid = helix,
        Mmodel = "Heteroduplex.2State",
        Save_results = "all",
        file_prefix = "Helix")
```

This will create seven pre-canned outputs. The first and second outputs are depictions of the fits in method Method 1, with the individual fits (red lines) overlay-ed on raw data (Figure 9A) and normalized data generated by calculating the absorbtivity (extinction coefficient) at each temperature (Figure 9B). The third output is a depiction of the fit in Method 2, with the fit overlay-ed on the T_m data at each strand concentration (Figure 9C). The fourth and fifth outputs are depictions of the fit in method Method 3, with the global fits (red lines) overlay-ed on raw data (Figure 9D) and normalized data generated by calculating the absorbtivity (extinction coefficient) at each temperature (Figure 9E). The sixth output is a .csv file containing the thermodynamic parameters generated by individual fits (Figure 9F). The seventh output is a .csv file containing the thermodynamic parameters from each method (Figure 9G).



Sample	Ct	H	S	G	Tm
2	2.8E-06	-63.5	-0.173	-9.9	43.1
3	4.7E-06	-71.2	-0.197	-10.1	44.6
4	7.4E-06	-64.8	-0.176	-10.0	46.4
5	1.2E-05	-64.5	-0.175	-10.1	48.4
6	2.0E-05	-70.9	-0.195	-10.4	50.0
7	3.1E-05	-74.3	-0.206	-10.5	51.4
8	5.5E-05	-71.6	-0.197	-10.4	53.0
9	9.0E-05	-74.8	-0.207	-10.7	54.8
10	1.4E-04	-76.7	-0.212	-11.0	57.0

Method	H	SE.H	S	SE.S	G	SE.G
1 individual fits	-70.3	4.9	-193.1	14.9	-10.4	0.3
2 Tm versus ln[Ct]	-60.4	1.2	-163.0	3.8	-9.9	0.1
3 Global fit	-69.7	0.3	-191.3	1.0	-10.4	0.0

Figure 9: Pre-canned meltR.A outputs

4.2.5 Specifying blanks with meltR.A

In this section, we will describe usage for the blank argument, which controls background subtraction. There is a single blank in the data set we fit in section 4.2.3. The identity of the blank sample specified in the "blank" argument, must match the identity of the blank sample in the data frame. If the blank sample is labeled with a character, the "blank" argument needs to be a character string, "Blank 1" for example. In the example in section 4.2.3, the blank sample is represented by a integer. We can check this using the "str" function in base R.

```
str(df.abs.data)
'data.frame':   1800 obs. of  4 variables:
 $ Sample     : int  1 1 1 1 1 1 1 1 1 1 ...
 $ Pathlength : num  1 1 1 1 1 1 1 1 1 1 ...
 $ Temperature: num  4.78 5.5 6.03 6.52 7.03 7.53 8.02 8.54 9.04 9.54 ...
 $ Absorbance : num  0.004395 -0.00058 0.000778 0.001038 0.001068 ...
```

As you can see, the Sample column in the data frame, which is why we specified the blank as:

```
blank = 1
```

You may have already performed the background subtraction on your data. In this case, blanks should be removed from the data set and background subtraction can be turned off in meltR.A by setting the blank argument to "none". There is a data set in MeltR that is already background subtracted.

```
?df.FAM.C.BHQ1.data
meltR.A(data_frame = df.FAM.C.BHQ1.data,
        blank = "none",
        NucAcid = c("RNA", "CUGAGUC", "GACUCAG"),
        Mmodel = "Heteroduplex.2State")
```

Thus, setting blank = "none" turns off background subtraction.

If there are multiple blanks in the data set, different blanks can be assigned to different blanks by supplying sample-blank pairs to the "blank" argument using a list of vectors. For example, the df.abs.ROX.data data set, included in MeltR, contains variable concentrations of ROX fluorophore. The ROX absorbance must be subtracted out. We first check the Sample identities.

```
?df.abs.ROX.data
unique(df.abs.ROX.data$Sample)
 [1] "0 uM ROX 0 uM Helix A"     "0 uM ROX 5 uM Helix A"
 [3] "0.032 uM ROX 5 uM Helix A" "0.1 uM ROX 5 uM Helix A"
 [5] "0.32 uM ROX 5 uM Helix A"  "3.2 uM ROX 5 uM Helix A"
 [7] "10 uM ROX 5 uM Helix A"    "32 uM ROX 5 uM Helix A"
 [9] "0.032 uM ROX 0 uM Helix A" "0.1 uM ROX 0 uM Helix A"
[11] "0.32 uM ROX 0 uM Helix A"  "3.2 uM ROX 0 uM Helix A"
[13] "10 uM ROX 0 uM Helix A"    "32 uM ROX 0 uM Helix A"
```

We then create a list of samples specifying sample-blank pairs. Samples are specified first and blanks are specified second.

```
list.blanks = list(c("0 uM ROX 5 uM Helix A", "0 uM ROX 0 uM Helix A"),
              c("0.032 uM ROX 5 uM Helix A", "0.032 uM ROX 0 uM Helix A"),
              c("0.1 uM ROX 5 uM Helix A", "0.1 uM ROX 0 uM Helix A"),
              c("0.32 uM ROX 5 uM Helix A", "0.32 uM ROX 0 uM Helix A"),
              c("3.2 uM ROX 5 uM Helix A", "3.2 uM ROX 0 uM Helix A"),
              c("10 uM ROX 5 uM Helix A", "10 uM ROX 0 uM Helix A"),
              c("32 uM ROX 5 uM Helix A", "32 uM ROX 0 uM Helix A"))
```

We can then supply the list "list.blanks" to the "blank" argument.

```
meltR.A(df.abs.ROX.data,
        blank = list.blanks,
        NucAcid = c("RNA", "CGAAAGGU", "ACCUUUCG"),
        methods = c(TRUE, FALSE, FALSE),
        Mmodel = "Heteroduplex.2State",
        fitTs = c(25, 75))
```

Note: I turned of methods 2 and 3 because the data set is not set up for these methods and it would likely cause the algorithm to fail because the data are inappropriate. Also, baselines are trimmed using the fitTs argument. This baseline trimming is discussed in more detail in section 4.2.7.

4.2.6 Specifying extinction coefficients

This section will describe how to specify extinction coefficients using the "NucAcid"" argument. In section 4.2.3, we supplied meltR.A with a nucleic acid sequence, and MeltR used extinction coefficient data to calculate an extinction coefficient at 260 nm, which meltR.A then uses to calculate C_t. MeltR contains data to calculate extinction coefficients at 300, 295, 290, 285, 280, 275, 270, 265, 260, 255, 250, 245, 240, 235, and 230 nm for RNA. MeltR contains data to calculate extinction coefficients at 260 nm for DNA. Extinction coefficient calculations assume pH 7.0.

You can change the wavelength that extinction coefficients are calculated at using the "wavelength" argument. We can test on the df.abs.ROX.data, even though this data was collected at 260 nm.

```
meltR.A(df.abs.ROX.data,
        wavelength = 280,
        blank = blanks,
        NucAcid = c("RNA", "CGAAAGGU", "ACCUUUCG"),
        methods = c(TRUE, FALSE, FALSE),
        Mmodel = "Heteroduplex.2State",
        fitTs = c(25, 75))
```

You can check what values meltR.A is calculating using "calc.extcoeff".

```
calc.extcoeff(c("RNA", "CGAAAGGU", "ACCUUUCG"))
$Total
[1] 115200

$CGAAAGGU
[1] 66120

$ACCUUUCG
```

```
[1] 49080
```

Custom molar extinction coefficients can be supplied to the NucAcid condition, if data is not included to calculate extinction coefficients for your specific nucleic acid. This is important for modified nucleic acids or extreme solution conditions. For example, the df.FAM.C.BHQ1.data set was collected on a fluorophore and quencher labeled helix. Thus, I need to use extinction coefficients that include the fluorophore and quencher.

```
?df.FAM.C.BHQ1.data
FAM.BHQ1 = c("Custom", 105860, 82300)
meltR.A(data_frame = df.FAM.C.BHQ1.data,
        blank = "none",
        NucAcid = FAM.BHQ1,
        Mmodel = "Heteroduplex.2State",
        concT = 75)
```

4.2.7 Refining meltR.A fits by trimming absorbance baselines.

MeltR approximates absorbance baselines using a line (y = mx+b). Real absorbance data deviates from this linear behavior over wide temperature ranges. MeltR fits are thus improved by trimming the baselines so that baselines better approximate a line. First, the raw data should be inspected for linear baselines one sample at a time using the tidyverse. Then, data are fit with meltR.A using a manually optimized baseline trim. After this, the manually optimized fit can be passed to an automated baseline trimmer, which tests combinations of randomly trimmed baselines and identifies an optimum set of trimmed baselines.

```
ggplot(df %>% filter(Sample == 10), #Only plot sample 10
aes(x = Temperature, y = Absorbance, color = factor(Sample))) +
  geom_point() +
  theme_classic() +
  geom_vline(xintercept = c(15, 70)) #will add horizontal lines to the plot at 15 and 70
```

Thus, for Sample 2, baselines are approximately linear above 80 °C and below 80 °C (Figure 10).

This should be repeated for each sample that contains RNA. After identifying ranges where baselines are linear, ranges can be supplied to the “fitTs” argument as a list of vectors (in the order of the samples in the data set). Note, do not provide a temperature range for the blank.

```
list.T.range = list(c(15, 70), #Sample 2
                    c(15, 70), #Sample 3
                    c(15, 80), #Sample 4
                    c(15, 85), #Sample 5
                    c(20, 78), #Sample 6
                    c(20, 85), #Sample 7
                    c(20, 85), #Sample 8
```

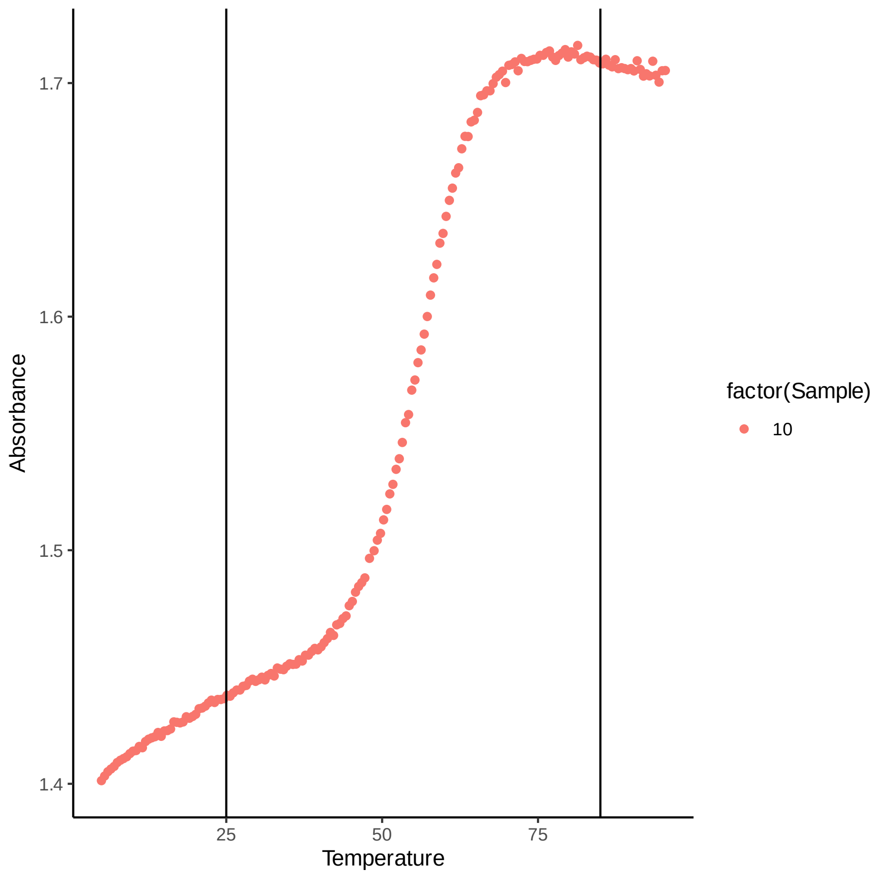


Figure 10: Baseline trimming on Sample 10. Vertical lines represent 25 and 80 °C.

```
                    c(25, 85), #Sample 9
                    c(25, 85)) #Sample 10

fit = meltR.A(data_frame = df,
        blank = 1,
        NucAcid = helix,
        Mmodel = "Heteroduplex.2State",
        Save_results = "all",
        file_prefix = "Trimmed",
        fitTs = list.T.range)
```

Which will produce:

```
[1] "Individual curves"
  Sample           Ct     H       S     G   Tm
1      2 2.913343e-06 -67.2 -0.1850  -9.9 42.4
2      3 4.853010e-06 -71.6 -0.1985 -10.1 44.5
3      4 7.650852e-06 -65.3 -0.1784 -10.0 46.1
4      5 1.207087e-05 -61.9 -0.1676  -9.9 47.9
5      6 2.107938e-05 -66.3 -0.1814 -10.0 49.3
6      7 3.220452e-05 -71.2 -0.1964 -10.3 50.8
7      8 5.691051e-05 -68.9 -0.1894 -10.1 52.3
8      9 9.246535e-05 -68.4 -0.1879 -10.1 53.9
9     10 1.480195e-04 -71.0 -0.1953 -10.4 56.1
[1] "Summary"
              Method     H SE.H      S SE.S     G SE.G Tm_at_0.1mM SE.Tm_at_0.1mM
1  1 individual fits -68.0  3.2 -186.7  9.9 -10.1  0.2    54.15446      3.8393625
2 2 Tm versus ln[Ct] -62.1  2.8 -168.9  8.7  -9.8  0.1    53.76500      3.6805415
3       3 Global fit -67.4  0.3 -184.8  0.9 -10.1  0.0    54.26074      0.3579972
[1] "fractional error between methods"
           H          S    G
1 0.08962025 0.09881569 0.03
[1] "dH and dG are in kcal/mol and dS is in cal/mol/K. Tms are in deg Celsius"
```

Fitting manually trimmed baselines reduces the fractional error between the methods from 15% to 9% in terms of the enthalpy.

Next, the “fit” from meltR.A can be fit using the “BLTrimmer”.

```
BLTrimmer(fit)
```

Which will yeild something like this:

```
[1] "You are trying to test 100 baseline combinations"
[1] "Do you think this is possible?"
[1] "Fitting 100 combinations of 5 different baselines per sample"
  |======================================================================| 100%[1] "Testi
[1] "Using autotrimmed baselines in MeltR"
[1] "Individual curves"
```

```
  Sample           Ct     H       S     G   Tm
1      2 2.913343e-06 -65.8 -0.1802  -9.9 42.6
2      3 4.853010e-06 -59.2 -0.1588  -9.9 45.2
3      4 7.650852e-06 -63.7 -0.1735  -9.9 45.7
4      5 1.207087e-05 -67.4 -0.1852 -10.0 47.2
5      6 2.107938e-05 -43.1 -0.1090  -9.3 50.5
6      7 3.220452e-05 -72.6 -0.2010 -10.3 50.6
7      8 5.691051e-05 -66.8 -0.1836  -9.9 51.7
8      9 9.246535e-05 -71.9 -0.1987 -10.3 53.9
9     10 1.480195e-04 -70.0 -0.1926 -10.3 55.8
[1] "Summary"
              Method     H SE.H      S SE.S     G SE.G Tm_at_0.1mM SE.Tm_at_0.1mM
1  1 individual fits -64.5  9.0 -175.8 28.2 -10.0  0.3    54.49795     11.5865784
2 2 Tm versus ln[Ct] -65.4  3.6 -179.2 11.0  -9.9  0.1    53.42931      4.4052573
3       3 Global fit -68.2  0.4 -187.3  1.2 -10.1  0.0    54.17182      0.4705376
[1] "fractional error between methods"
           H          S    G
1 0.05603231 0.06361792 0.02
[1] "dH and dG are in kcal/mol and dS is in cal/mol/K. Tms are in deg Celsius"
```

In this example, the BLTrimmer reduces the fractional error between methods from 9% to 5.6%. Note, your answer when you run this code will be different, because the BLTrimmer tests 100 random baseline combinations. By, default, there are n.ranges^n.samples = 5^9 baseline combinations (5 different trimmed baselines per sample and 9 samples). If you want to get a more similar answer, you will need to do a more exaustive testing of baseline combinations. For example,

```
BLTrimmer(fit, n.combinations = 1000)
[1] "You are trying to test 1000 baseline combinations"
[1] "Do you think this is possible?"
[1] "Fitting 1000 combinations of 5 different baselines per sample"
  |======================================================================| 100%[1] "Testi
[1] "Using autotrimmed baselines in MeltR"
[1] "Individual curves"
  Sample           Ct     H       S     G   Tm
1      2 2.913343e-06 -65.8 -0.1802  -9.9 42.6
2      3 4.853010e-06 -59.2 -0.1588  -9.9 45.2
3      4 7.650852e-06 -65.5 -0.1792  -9.9 45.8
4      5 1.207087e-05 -66.5 -0.1822  -9.9 47.2
5      6 2.107938e-05 -43.1 -0.1090  -9.3 50.5
6      7 3.220452e-05 -72.3 -0.1998 -10.3 50.8
7      8 5.691051e-05 -69.3 -0.1911 -10.1 51.9
8      9 9.246535e-05 -71.7 -0.1980 -10.2 53.7
9     10 1.480195e-04 -70.0 -0.1926 -10.3 55.8
[1] "Summary"
              Method     H SE.H      S SE.S     G SE.G Tm_at_0.1mM SE.Tm_at_0.1mM
```

```
1  1 individual fits -64.8  9.1 -176.8 28.3 -10.0  0.3    54.35821     11.5749687
2 2 Tm versus ln[Ct] -65.4  3.4 -179.0 10.6  -9.9  0.1    53.75580      4.2359722
3       3 Global fit -65.8  0.5 -180.0  1.4 -10.0  0.0    54.11935      0.5884726
[1] "fractional error between methods"
           H          S          G
1 0.01530612 0.01791713 0.01003344
[1] "dH and dG are in kcal/mol and dS is in cal/mol/K. Tms are in deg Celsius"
```

Note, that this produces a different answer, that is more reproducible, but requires more computational time, in this case 36.1 seconds. If you have a multicore machine, you can test baselines in parallel. First, you must install the "doParallel" package and its dependencies, and load them into your memory.

```
install.packages("doParallel")
library(doParallel)
```

Now, you can turn on the parallelization and specify the number of cores.

!!!!!Warning. Copying and pasting this code into your console could hurt your computer. Make sure you know how many cores your computer has and make sure you know how many cores can be designated to run this task!!!!!

```
BLTrimmer(fit, n.combinations = 1000,
          parallel = "on", n.core = 3,
          Save_results = "all")
[1] "You are trying to test 1000 baseline combinations"
[1] "Do you think this is possible?"
[1] "Fitting 1000 combinations of 5 different baselines per sample"
[1] "Testing baselines took 13.3319999999994 seconds"
[1] "Using autotrimmed baselines in MeltR"
[1] "Individual curves"
  Sample           Ct     H       S     G   Tm
1      2 2.913343e-06 -65.8 -0.1802  -9.9 42.6
2      3 4.853010e-06 -59.2 -0.1588  -9.9 45.2
3      4 7.650852e-06 -65.0 -0.1777  -9.9 45.9
4      5 1.207087e-05 -66.5 -0.1822  -9.9 47.2
5      6 2.107938e-05 -43.1 -0.1090  -9.3 50.5
6      7 3.220452e-05 -72.3 -0.1998 -10.3 50.8
7      8 5.691051e-05 -68.4 -0.1880 -10.1 52.1
8      9 9.246535e-05 -68.8 -0.1892 -10.1 53.8
9     10 1.480195e-04 -70.9 -0.1951 -10.4 55.9
[1] "Summary"
              Method     H SE.H      S SE.S     G SE.G Tm_at_0.1mM SE.Tm_at_0.1mM
1  1 individual fits -64.4  8.9 -175.6 27.6 -10.0  0.3    54.32267     11.3692822
2 2 Tm versus ln[Ct] -65.6  3.5 -179.8 10.7  -9.9  0.1    53.44949      4.2719888
3       3 Global fit -66.7  0.3 -182.7  1.0 -10.0  0.0    54.19970      0.3839714
[1] "fractional error between methods"
          H          S          G
```

```
1 0.0350788 0.03958372 0.01003344
[1] "dH and dG are in kcal/mol and dS is in cal/mol/K. Tms are in deg Celsius"
```

Note, this obtained a similar answer to running the code with no parallelization but required 13.6 seconds instead of 36.1 seconds.

Lastly, we can check the quality of the BLTrimmer results using some precanned outputs which are generated by setting Save_results = “all”. See section 4.2.4 for details on saving results.

The BLTrimmer produces, three precanned plots. The first is a histogram of the difference in enthalpies obtained by method 1 and method 2 obtained by fitting all of the baseline (Figure 11A). The peak of this distribution is occurring at about 6% enthalpy error, indicating that the helices are likely two state. The second is the enthalpy produced by methods 1 and 2 versus the fractional error between the two methods (Figure 11B). The enthalpies should converge on a single enthalpy value, plus or minus 1 kcal/mol, at low fractional error values. The third is a depiction of the trimmed baselines, where the blue points are the no trim range, red points are the baselines that are fit, and black points are points that are removed from the data set by the BLTrimmer (Figure 11C). The baselines produced by this example make sense.

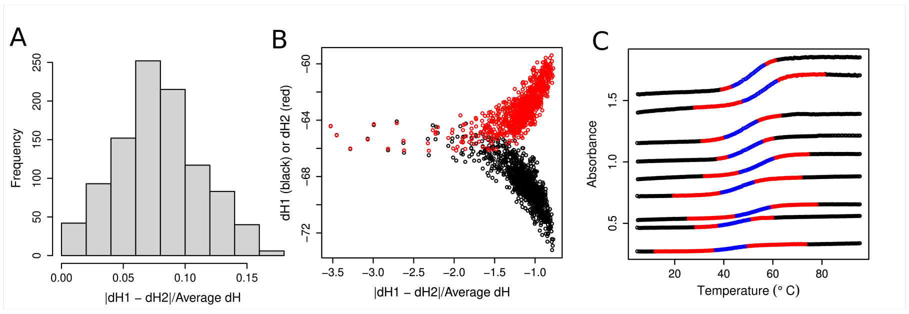


Figure 11: Precanned outputs of the BL trimmer. Enthalpy units are in kcal/mol.

4.2.8 Advanced plotting meltR.A outputs using the “tidyverse”

meltR.A can pass a more extensive output to an object in R. This also applies to the BLTrimmer.

```
MeltR.fit = meltR.A(data_frame = df,
        blank = 1,
        NucAcid = helix,
        Mmodel = "Heteroduplex.2State",
        Save_results = "all",
        file_prefix = "Trimmed",
        fitTs = list.T.range)
```

The object, “MeltR.fit” is now a list of objects that can be passed to plotting functions.

```
names(MeltR.fit)
```

Which will print a list of fit names:

```
 [1] "Summary"           "Method.1.indvfits" "Range"
 [4] "Derivatives.data"  "Method.1.data"     "Method.1.fit"
 [7] "Method.2.data"     "Method.2.fit"      "Method.3.data"
[10] "Method.3.fit"
```

[1] Summary: A data frame containing the thermodynamic parameters from each method. It can be called using:

```
MeltR.fit$Summary
```

[2] Method.1.indfits: A data frame containint the thermodynamic parameters from the individual fits.It can be called using:

```
MeltR.fit$Method.1.indvfits
```

[3] Fractional error between Method 1, 2, and 3. It can be called using:

```
MeltR.fit$Range
```

[4] Derivatives.data: Contains the first and second derivatives for each sample containing RNA. It can be called and plotted using:

```
MeltR.fit$Derivatives.data
ggplot(MeltR.fit$Derivatives.data, aes(x = Temperature, y = dA.dT/(Pathlength*Ct), color = f
  geom_point() +
  theme_classic()
```

[5] Method.1.data: Contains the raw data from method 1 and the model. It can be called and plotted using:

```
MeltR.fit$Method.1.data
ggplot(MeltR.fit$Method.1.data, aes(x = Temperature, color = factor(Sample), group = factor(
  geom_point(mapping = aes(y = Absorbance/(Pathlength*Ct))) +
  geom_line(mapping = aes(y = Model/(Pathlength*Ct)), color = "black") +
  theme_classic()
```

[6] MeltR.fit$Method.1.fit: A list of nls objects containing the fits obtained from fitting melting curves individually. It can be called using:

```
MeltR.fit$Method.1.fit
```

[7] MeltR.fit$Method.2.data: Contains the raw data from method 1 and the model. It can be called and plotted using:

```
MeltR.fit$Method.2.data
ggplot(MeltR.fit$Method.2.data, aes(x = lnCt)) +
  geom_point(mapping = aes(y = invT)) +
  geom_line(mapping = aes(y = Model)) +
  theme_classic()
```

[8] Method.2.fit: A nls object containing the fit obtained from fitting the relationship of Tm and Ct. It can be called using:

```
MeltR.fit$Method.2.fit
```

[9] Method 3 data: Contains the raw data from method 2 and the model. It can be called and plotted using:

```
MeltR.fit$Method.3.data
ggplot(MeltR.fit$Method.3.data, aes(x = Temperature, color = factor(Sample), group = factor(
  geom_point(mapping = aes(y = Absorbance/(Pathlength*Ct))) +
  geom_line(mapping = aes(y = Model/(Pathlength*Ct)), color = "black") +
  theme_classic()
```

[10] Method.3.fit: A nls object containing the fit obtained from fitting the raw data. It can be called using:

```
MeltR.fit$Method.3.fit
```

5 References

[1] Biochemistry 1996, 35 (45), 14077–14089. [2] Methods in Enzymology, 1989; Vol. 180, pp 304-325. [3] Biochemistry 1998, 37 (42), 14719–14735. [4] Welcome | R for Data Science https://r4ds.had.co.nz/ (accessed 2022 -05 -02). [5] ggrepel: An R package https://ggrepel.slowkow.com/index.html (accessed 2022 -05 -02).