# eSPC, an Online Data Analysis Platform for Molecular Biophysics

## CheMelt 1.0 User Documentation

September 2026

# Table of Contents

# 1. Data import and processing

## 1.1. Input file

CheMelt can parse several types of files:

A) The xlsx file (processed) generated by the Nanotemper Prometheus instrument that has one sheet called 'Overview' with a column called 'Sample ID' with the names of the samples (Figure 1), and four sheets called 'Ratio', '330nm', '350nm' and 'Scattering'. The first column of the signal sheet ('Ratio', '330nm', '350nm', 'Scattering') should be called 'Time [s]'. The second column should have the temperature data and all subsequent columns store the fluorescence data (Figure 2). The order of the fluorescence columns should match the order of the 'Sample ID' column in the 'Overview' sheet.

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

**Figure 1.** Example of the ‘SampleID’ column in the ‘Overview’ sheet required by MoltenProt to load the Nanotemper spreadsheet input file.

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

**Figure 2.** Example of the 'Ratio' sheet required by MoltenProt to load the Nanotemper spreadsheet input file.

B) The xls file generated by the ThermoFluor assay in a qPCR instrument. This file has one sheet called 'RFU' where the first row has the sample positions (header), the first column has the temperature data and all subsequent columns store the fluorescence data (Figure 3).

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

**Figure 3.** Example of the 'RFU' sheet required by MoltenProt to load ThermoFluor data.

C) The xlsx file generated by a Prometheus Panta instrument. This file has one sheet called 'Overview' with a column called 'Sample ID' with the names of the samples (Figure 1), and one sheet called 'Data Export' where all the data is stored (Figure 4). The 'Data Export' sheet columns should have the following order:

Temperature capillary 1 ; Ratio capillary 1 ; … ; Temperature capillary 1 ; 350 nm capillary 1 ; … ; Temperature capillary 1 ; 330 nm capillary 1 ; … ;

Temperature capillary 1 ; scattering capillary 1 ; … ; Temperature capillary 2 ; Ratio capillary 2; … ; Temperature capillary *n* ; Ratio capillary *n*.

Columns whose names include "Derivative" are not read.

| Temperature for Cap.1 (°C) | Ratio 350 nm / 330 nm for Cap.1 |
|---|---|
| 24.9993991851807 | 0.668331980705261 |
| 25.0000820159912 | 0.668272197246552 |
| 25.0013084411621 | 0.668575644493103 |
| 25.001501083374 | 0.66842645406723 |
| 25.0040893554687 | 0.667966544628143 |
| 25.0060691833496 | 0.668425559997559 |
| 25.0104427337646 | 0.668148577213287 |
| 25.0115489959717 | 0.668168842792511 |
| 25.016487121582 | 0.668507099151611 |

**Figure 4.** Example of the 'Data Export' sheet required by MoltenProt to load the Prometheus Panta spreadsheet input file.

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

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

[Melt Curve Raw Data]
Well    Well Position   Reading Temperature     Fluorescence    Derivative      Target Name
10      A10     1       24.913  4,828.486       -1,144.751      Target 1
10      A10     2       25.028  4,951.091       -905.159        Target 1
10      A10     3       25.142  4,722.069       -659.505        Target 1
10      A10     4       25.257  4,786.491       -469.428        Target 1
10      A10     5       25.371  5,124.026       -375.954        Target 1
10      A10     6       25.486  4,948.212       -373.484        Target 1
10      A10     7       25.601  4,832.502       -411.521        Target 1
10      A10     8       25.715  4,961.322       -422.667        Target 1
10      A10     9       25.830  4,779.467       -357.793        Target 1
10      A10     10      25.944  5,039.901       -208.366        Target 1
```

**Figure 5.** Example of the input file required by MoltenProt to load the QuantStudio™ 3 System instrument data.

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

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

**Figure 6.** Example of the 'Results' sheet required by MoltenProt to retrieve the sample names. This example file was generated by the Nanotemper Prometheus Tycho instrument.

The 'Profiles_raw' sheet columns should have the following structure (Figure 7):

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

One row with the capillary numbers.

One row with the time, temperature and sample names.

The remaining rows store the temperature and signal data.

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

**Figure 7.** Example of the 'Profiles_raw' sheet required by MoltenProt to load the temperature and signal data. This example file was generated by the Nanotemper Prometheus Tycho instrument.

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

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

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

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

The JSON file should have the following structure:

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

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

- 'Temperature'
- 'Signal'

Below you'll find a minimal example.


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

```
    311,
    312
  ]
}
```

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

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

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

**Figure 8.** Example of a csv file that can be imported into MoltenProt.

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

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

**Figure 9.** Example of one sheet from the UNCLE xlsx file.

## 1.2. Scaling

To aid the fitting algorithm, each denaturation curve $S_i$ can be rescaled by the global maximum $S_{max}$ as follows:

$$S_i = \frac{S_i}{S_{max}} \cdot 100 \qquad (1)$$

## 1.3. First derivative and apparent temperature of melting

Given a denaturation curve $S_i$, the derivative is taken with respect to temperature. Firstly, if the temperature step is not uniform, a new $S_{i,int}$ is calculated by linear interpolation (step of 0.1 °C). Secondly, the number of points *n* is calculated as:

$$n = ceil(\frac{8}{\Delta T}) \,//\, 2 * 2 + 1 \qquad (2)$$

where $ceil(x)$ returns the smallest integer $i$ such that $i >= x$, and $\Delta T$ is the temperature step. Lastly, the derivative is calculated using the *savgol_filter* function from the *Scipy* package, with a fourth-degree polynomial and filter window of length *n*.

Once the first derivative has been calculated, we use a non-model approach to estimate an apparent melting temperature ($T_m^{app}$). Firstly, the median value of the first derivative in the interval $[min(T) + 6; min(T) + 11]$ and $[max(T) - 11; max(T) - 6]$ is calculated. Then, we obtain the mean of those two median values and add it (if it positive), or subtract it (if it is negative), to the first derivative in the interval $[min(T) + 6; max(T) - 6]$. This is done to shift the derivative baseline. Lastly, if the absolute value of the minimum (of the derivative) is higher than the absolute value of the maximum, we use the minimum to estimate the $T_m^{app}$. Otherwise, we use the maximum. If many curves are present, we always use the same option.

# 2. Model

## 2.1 Theory

### 2.1.1. Combined chemical and thermal denaturation

We assume that the protein is a monomer that (un)folds reversibly via a two-state process. The fraction of protein in the native ($f_n$) and denatured ($f_u$) states depend on the unfolding equilibrium constant *K*.

$$f_n(K) = (1 + K(T,D))^{-1} \quad (3)$$

$$f_u(K) = 1 - f_n \quad (4)$$

where *T* is the temperature in Kelvin units and *D* is the denaturant concentration in molar units. The unfolding equilibrium constant *K* is related to the Gibbs free energy Δ*G*.

$$K(T,D) = e^{-\frac{\Delta G(T,D)}{RT}} \quad (5)$$

where *R* is the gas constant. If only chemical denaturation is considered, Δ*G* can be calculated using the Linear Extrapolation Model[1].

$$\Delta G_u(D) = \Delta G_{H_2O} - m \cdot D \quad (6)$$

where *m* is an empirical parameter that allows modelling Δ*G* with a linear dependence on denaturant concentration. $\Delta G_{H20}$ is the free energy of unfolding in the absence of a denaturant agent. On the other hand, if thermal denaturation is considered, the free energy of unfolding can be calculated as indicated below:

$$\Delta G_u(T) = \Delta H(1 - \frac{T}{T_m}) + \Delta C_p(T - T_m - Tln(\frac{T}{T_m})) \quad (7)$$

where Δ*H* is the enthalpy of unfolding, $T_m$ is the temperature of melting, and $\Delta C_p$ is the heat capacity of unfolding. The combination of both denaturation methods results in the expression[2]:

$$\Delta G_u(T,D) = \Delta H(1 - \frac{T}{T_m}) + \Delta C_p(T - T_m - Tln(\frac{T}{T_m})) - m \cdot D \quad (8)$$

### 2.1.2 Equation of the signal

The total signal is assumed to be a linear combination of the signal produced by the different protein states weighted by their mole fractions.

$$S(\Delta T, D) = f_n(T,D) \cdot S_n(\Delta T, D) + f_u(T,D) \cdot S_u(\Delta T, D) \quad (9)$$

---

[1] Greene, Raymond F., and C. Nick Pace. 1974. "Urea and Guanidine Hydrochloride Denaturation of Ribonuclease, Lysozyme, α-Chymotrypsin, and B-Lactoglobulin." *Journal of Biological Chemistry* 249 (17): 5388–5393.

[2] Hamborg, Louise, Emma Wenzel Horsted, Kristoffer Enøe Johansson, Martin Willemoës, Kresten Lindorff-Larsen, and Kaare Teilum. 2020. "Global Analysis of Protein Stability by Temperature and Chemical Denaturation." *Analytical Biochemistry* 605 (September): 113863.

where $S_n(T, D)$ and $S_u(T, D)$ are functions describing the signal dependence on temperature and denaturant concentration for the native and unfolded states, respectively. Here, the background signal is negligible. The dependence on temperature can be modelled with a quadratic equation:

$$S_x(\Delta T, D = 0) = a + b\Delta T + c\Delta T^2 \qquad (10)$$

where *a* is the intercept, *b* is the linear term, and *c* is the quadratic term. $\Delta T$ is the difference between the temperature and a reference temperature (298K). If *c* is zero, then the signal depends linearly on the temperature. If both *b* and *c* are zero, then there is no dependence on temperature. Alternatively, the baselines can be modelled with an exponential equation:

$$S_x(\Delta T, D = 0) = \gamma + \delta e^{-\alpha \cdot \Delta T} \qquad (11)$$

where $\gamma$ is the intercept, $\delta$ the pre-exponential term, and $\alpha$ the exponential coefficient. Lastly, the dependence on denaturant concentration can be assumed to be linear.

$$S_x(D, \Delta T = 0) = S_x(\Delta T = 0) + d \cdot D \qquad (12)$$

where $S_x$ evaluated at $\Delta T$ equal to zero implies that the intercept terms are shared. Here, *d* is the slope term for the dependence on denaturant concentration.

## 2.2. Limitations

The model presented here is based on several suppositions, which might not be true in many cases. To start with, the protein under study must be a monomer, the unfolding should be reversible and follow a two-state process, both when using a chemical agent and temperature. The heat capacity of unfolding $\Delta C_p$ and *m*-value are considered to be independent from temperature and denaturant concentration. The background signal is considered negligible and the folded and unfolded states should produce a different signal. In this regard, the ensemble of thermally-induced and denaturant-induced unfolded states are considered to be equal. Furthermore, the denaturant contribution to the free energy of unfolding for chemical denaturation is based on the empirical Linear Extrapolation Model, which may not adequately explain the unfolding process. Furthermore, the equations to model the baselines do not have a physical background and are based on empirical observations. Lastly, the data may not constrain the parameter space enough, turning the fitting into an ill-posed problem, where a set of unfolding curves can be explained by different sets of parameter values.

# 3. Fitting

## 3.1. Baselines

The fitting is done with global thermodynamic parameters and local or global baseline parameters. The different options are listed below, from more to less parameters.

- Local baselines and local slopes: The intercept terms are local, and the slope terms, such as *a*, *b* from Eq. 12, or $\gamma$ and $\delta$ from Eq. 12 are also local. There is no dependence on denaturant concentration.

- Local baselines and shared slopes: The intercept terms are local, and the slope terms are shared. There is no dependence on denaturant concentration.

- Shared baselines and shared slopes: The slope terms are shared. There is a linear dependence on denaturant concentration that allows having shared intercept terms.

- Shared baselines and shared slopes with scaling factor: Same as before, with one extra term (scaling factor) per denaturation curve.

As a reference, to analyse ten unfolding curves with **exponential baselines**, the number of parameters for the different models is presented in the Table below.

| **Method** | **# Parameters** |
|---|---|
| Local baselines and local slopes | 64 |
| Local baselines and shared slopes | 28 |
| Shared baselines and shared slopes (no scaling factor) | 12 |
| Shared baselines and shared slopes (with scaling factor) | Between 12 and 21.* |

*Scaling factors that are almost equal to one are automatically removed.

## 3.2. Fitting algorithm

Once a model is selected, the parameters are estimated by minimizing the residual sum of squares.

## 3.3. Parameters errors

The default errors provided in the 'Fitted params' Table are obtained using the square root of diagonal values from the fit parameter covariance matrix. These errors are an underestimation of the true errors and assume a linear model.

For a better estimate of the parameter errors two more methods are available, namely the profile-likelihood method and the leave-one-out method. The former consists of fixing a certain parameter around the vicinity of the fitted value and reoptimizing the other parameters values until the residual-sum-of-squares changes significantly. The critical value is found using an F-test with 95 % confidence. The latter consists of performing *n* fits, where *n* corresponds to the total number of unfolding curves. For each fit, one unfolding curve is systematically excluded from the analysis. Afterwards, for each parameter, such as $\Delta C_p$, the mean and standard deviation derived from the *n* fits are provided.

## 3.4. Statistical criteria

If the option "compare models" is used in the "Fitting" tab, several statistical criteria are provided to the User to guide the selection of the "best" model. The criteria are the Akaike information Criterion (AIC), the Bayesian Information Criterion (BIC), and the extended Bayesian Information Criterion (eBIC) using a gamma of one. Models are ranked according to the eBIC values. The formula of each criterion is provided below:

$$\text{AIC:} \quad 2k - 2ln(\widehat{L}) \tag{13}$$

$$\text{BIC:} \quad k \cdot ln(n) - 2ln(\widehat{L}) \tag{14}$$

$$\text{eBIC:} \quad k \cdot ln(n) - 2ln(\widehat{L}) + 2\gamma \cdot ln(\frac{p!}{k!(p-k)!}) \tag{15}$$

where *k* is the number of parameters, $\widehat{L}$ is the maximized value of the likelihood function, *n* is the number of data points, and *p* is the number of parameters. The value of $\gamma$ is set to one in CheMelt. The value of $\widehat{L}$ is estimated using a simplified version that removes a constant term shared by the different fitted models:

$$\widehat{L} = \frac{\chi^2}{n} \tag{16}$$

where $\chi^2$ is the chi-squared value of the fitting.

## 3.5. Interpretation

We recommend discarding a certain fit if the thermodynamic parameters have high errors, if the parameters are physically unreasonable, or if the model predicts certain

phenomena that are not observed, such as denaturation at unexpected temperatures (e.g., 15 ºC). Furthermore, we recommend visualizing the fitted curves and disregarding a fit if the fitted curves do not match the experimental raw data.

Moreover, utilising the statistical criteria to select a model is no guarantee that the selected model corresponds to the truth underlying data generating process. A reasonable alternative is to report thermodynamic parameters only if the “best” two (or three) ranked models provide reasonably similar parameter values.

## Contact details

For further assistance, please contact us:


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

## Packages

**CheMelt is possible thanks to:**


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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Python package openpyxl: https://openpyxl.readthedocs.io/en/stable/

Python package pychemelt: https://github.com/osvalB/pychemelt