eSPC, an Online Data Analysis Platform for Molecular Biophysics

MoltenProt 1.0 User Documentation

March 2023

Table of Contents

Overview
MoltenProt has seven panels (Figure 1). Panels 1-3 contain the necessary steps to analyze user data.

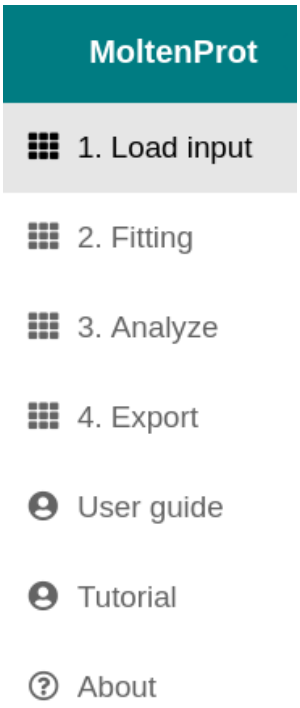

Figure 1. MoltenProt online tool panels.

1. Load input

1.1. Input file (raw data)

MoltenProt accepts as input four kinds 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 2), and four sheets called 'Ratio', '330nm', '350nm' and 'Scattering'. The first column of the signal sheet ('Ratio', '330nm', '350nm', 'Scattering') should be called 'Time [s]'. The second column should have the temperature data and all subsequent columns store the fluorescence data (Figure 3). The order of the fluorescence columns should match the order of the 'Sample ID' column in the 'Overview' sheet.

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

Figure 2. Example of the ‘SampleID’ column in the ‘Overview’ sheet required by 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 3. 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 4).

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

Figure 4. Example of the 'RFU' sheet required by 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 2), and one sheet called 'Data Export' where all the data is stored (Figure 5). The 'Data Export' sheet columns should have the following order:

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

Columns whose names include "Derivative" are not read.

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

Figure 5. 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 6). Comments should be at the beginning of the file and start with '*'. The header is the first line with more than 6 words, i.e. 'Well', 'Well Position', … , 'Target Name'. The column number 2 has the sample IDs. The columns number 4 and 5 have the signal and temperature data.

```
* 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 6. 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 7), and one sheet called 'Profiles_raw' where the fluorescence data is stored.

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

Figure 7. Example of the ‘Results’ sheet required by 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 8):

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

One row with the capillary numbers.

One row with the time, temperature and sample names.

The remaining rows store the 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 8. 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).

1.2. Normalization

There are 3 available options to normalize each fluorescence-based melting curve.

a. Divide by initial value: Divide by the median value of the signal corresponding to the first two degrees of temperature.

b. Max-min normalization: Transform the signal by applying

$$NormalizedSignal(Signal) = \frac{Signal - min(Signal)}{max(Signal) - min(Signal)}$$ Equation 1

c. Area normalization: Divide the signal by the area under the curve (calculated using the trapezoidal rule).

1.3. Median filter (smoothing)

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

1.4. Savitzky-Golay (SG) window size

This parameter, in degrees Celsius, is used to calculate the number of data points to apply the Savitzky-Golay filter corresponding to a polynomial of degree 4 before computing the first or second derivative as implemented in Scipy (scipy.signal.savgol_filter — SciPy v1.6.1 Reference Guide). For the second derivative, we add 5 degrees to the selected SG temperature window size.

The number of data points is obtained by computing

$$oddDataPoints(SavitzkyGolayWindowSize) = ceil(\frac{SavitzkyGolayWindowSize}{deltaTemperature}) // 2 * 2 + 1$$

Equation 2

where $SavitzkyGolayWindowSize$ is the SG parameter fixed by the user, $ceil(x)$ returns the smallest integer $i$such that $i >= x$, and $deltaTemperature$ corresponds to the average number of data points in one degree of temperature.

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

A non-model approach to estimate the melting temperature involves estimating the maximum or the minimum of the first derivative, depending on the way the signal changes with the temperature. In MoltenProt, the Tm values are estimated as follows. First, the median value of the first derivative in the interval $[min(temperature) + 6; min(temperature) + 11]$ and $[max(temperature) - 11; max(temperature) - 6]$ is calculated. Then, we obtain the mean of those two median values and add it (if it positive), or subtract it (if it is negative), to the first derivative in the interval $[min(temperature) + 6; max(temperature) - 6]$. This is done to shift the derivative baseline. Last, if the absolute value of the minimum (of the derivative) is higher than the absolute value of the maximum, we use the minimum to estimate the Tm. Otherwise, we use the maximum. If many curves are present, we always use the same option.

2. Fitting

2.1. Model selection

The models from the online version of MolteProt are based on the desktop application developed by Kotov et al. (Kotov *et al.*, 2021). All of them assume that the fluorescence signal can be expressed as the sum of the signal from different

protein states where the dependence of the signal to the temperature is given by a linear function. The difference in the models lies in establishing which are the possible protein states and how to calculate their concentration (Figure 9).

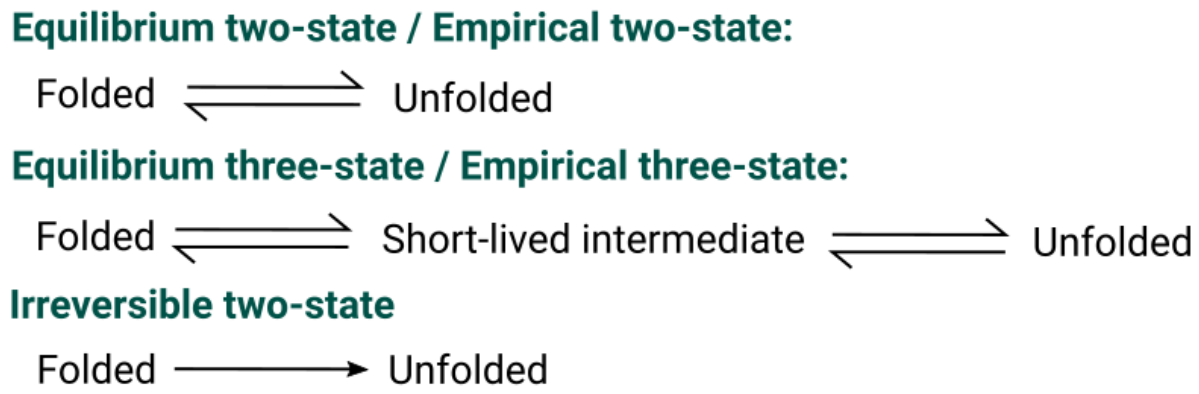

Figure 9. Five unfolding models have been implemented in the online version of MoltenProt. The different protein states are Folded, Unfolded, Short-lived intermediate, and the reaction(s) may be reversible or irreversible.

Equilibrium two-state[1,2]

This thermodynamic-based model presupposes that the protein only exists in the native (folded) or unfolded state and that there is an equilibrium between these two states given by the unfolding reaction N ⇆ U.The fluorescence signal $F(T)$ is described by the equation

$$F(T) = (k_n T + b_n + (k_u T + b_u) * e^{\frac{\Delta H_m}{R}(\frac{1}{T_m} - \frac{1}{T})})(1 + e^{\frac{\Delta H_m}{R}(\frac{1}{T_m} - \frac{1}{T})})^{-1} \quad (1)$$

where k_n, b_n are the slope and intercept of the pre-transition baseline (native), k_u and b_u are the slope and intercept of the post-transition (unfolded) baseline, R is the universal gas constant, ΔH_m is the enthalpy of unfolding at the melting temperature T_m. This model assumes that the heat capacity ΔC_p of unfolding equals zero. If the user has knowledge about these values, it can be used to correct later the calculated Gibbs energy of unfolding at the standard temperature ($25°C$, 298.15 K) by using

$$\Delta G_{298.15}(\Delta H_m, T_m, C_p) = \Delta H_m * (1 - \frac{298.15\ K}{T_m}) - C_p * (dC_{p,Component}) \quad (2)$$

[1] Santoro, M. M., & Bolen, D. W. (1988). Unfolding free energy changes determined by the linear extrapolation method. 1. Unfolding of phenylmethanesulfonyl. alpha.-chymotrypsin using different denaturants. *Biochemistry*, *27*(21), 8063-8068.

[2] Bedouelle, H. (2016). Principles and equations for measuring and interpreting protein stability: From monomer to tetramer. *Biochimie*, *121*, 29-37.

where

$$dC_{p,Component}(T_m) = T_m - 298.15\,K + 298.15\,K * ln(\frac{298.15\,K}{T_m}) \quad (3)$$

Empirical two-state[3]

This model is similar to the Equilibrium two-state, but instead of enthalpy of unfolding, it uses the descriptive parameter T_{onset} to describe the steepness of the fluorescence curve. The signal is described by the equation:

$$F(T) = (k_n T + b_n + (k_u T + b_u) * A1)(1 + A1)^{-1} \quad (4)$$

where

$$A1 = exp(\frac{(T - T_m) * ln(0.01 / 0.99)}{T_{Onset} - Tm}) \quad (5)$$

Equilibrium three-state[4]

This model adds the presence of one short-lived protein state: native (N), intermediate (I) and unfolded (U). The signal is described by the equation:

$$F(T) = (k_n T + b_n + k_1 A1 + (k_u T + b_u) * A1 * A2)(1 + A1 * A2)^{-1} \quad (6)$$

where

$$A1 = exp(\frac{\Delta H_{m1}}{R}(\frac{1}{T_1} - \frac{1}{T})) \quad (7)$$

and,

$$A2 = exp(\frac{\Delta H_{m2}}{R}(\frac{1}{T_2} - \frac{1}{T})) \quad (8)$$

---

[3] Kotov, V., Bartels, K., Veith, K., Josts, I., Subhramanyam, U. K. T., Günther, C., ... & Garcia-Alai, M. M. (2019). High-throughput stability screening for detergent-solubilized membrane proteins. *Scientific reports*, *9*(1), 1-19.

[4] Mazurenko, S., Kunka, A., Beerens, K., Johnson, C. M., Damborsky, J., & Prokop, Z. (2017). Exploration of protein unfolding by modelling calorimetry data from reheating. *Scientific reports*, *7*(1), 1-14.

where k_1 is the signal slope for the short-lived I state, ΔH_{m1} and ΔH_{m2} are the enthalpy of unfolding at melting temperatures T1 (N ⇆ I) and T2 (I ⇆ U).

Empirical three-state[5]

This model is similar to the Equilibrium three-state, but instead of enthalpy of unfolding, it uses the descriptive parameters T_{onset1} and T_{onset2} to describe the steepness of the fluorescence curve. The signal is described by the equation:

$$F(T) = (k_n T + b_n + k_1 A1 + (k_u T + b_u) A1 * A2)(1 + A1 * A2)^{-1} \quad (9)$$

where

$$A1 = exp(\frac{T - T_1}{T_{Onset1} - T_1} ln(0.01/0.99)) \quad (10)$$

and,

$$A2 = exp(\frac{T - T_2}{T_{onset2} - T_2} ln(0.01/0.99)) \quad (11)$$

Irreversible two-state[6],[7]

This model assumes that the protein only exists in the native (N) and unfolded (U) state and that the unfolding reaction is irreversible. The signal is described by the equation:

$$F(T) = k_u T + b_u + (k_n T + b_n) * x_n(T) \quad (12)$$

where $x_n(T)$ is the fraction of natively folded molecules as a function of temperature and can be obtained via numerical integration:

$$x_n(T) = \int_{Tmin}^{Tmax} \frac{-1}{v} * exp(\frac{-E_a}{R}(\frac{1}{T} - \frac{1}{T_f}) * x_n) \quad (13)$$

---

[5] Kotov, V., Bartels, K., Veith, K., Josts, I., Subhramanyam, U. K. T., Günther, C., ... & Garcia-Alai, M. M. (2019). High-throughput stability screening for detergent-solubilized membrane proteins. *Scientific reports*, *9*(1), 1-19.

[6] Mazurenko, S., Kunka, A., Beerens, K., Johnson, C. M., Damborsky, J., & Prokop, Z. (2017). Exploration of protein unfolding by modelling calorimetry data from reheating. *Scientific reports*, *7*(1), 1-14.

[7] Sanchez-Ruiz, J. M. (1992). Theoretical analysis of Lumry-Eyring models in differential scanning calorimetry. *Biophysical journal*, *61*(4), 921-935.

where $Tmax$ and $Tmin$are the start and end temperatures of the measurement, $v$is the scan rate in degrees/minutes, E_a is the activation energy of unfolding, $T_f$is the temperature where the reaction rate constant of unfolding equals 1. For simplicity, x_n is assumed to be 1 at $Tmin$.

2.2. Temperature range for baseline estimation

All models require the parameters k_u, b_u, $k_n$and b_n. The initial values of these parameters are estimated by fitting the equation of a line to the first or last n-degrees (selected by the user).

2.3. Curve fitting

Each curve is fitted individually using the Levenberg Marquardt (damped least-squares) algorithm as implemented in the curve_fit function from the Scipy package. The initial estimates of k_u, b_u, $k_n$and $b_n$are used to provide fitting boundaries (± 40 % of the initial values). For the other parameters, the fitting boundaries are described in the following Table.

Parameter	Lower bound	Upper bound
T_m	Lowest temperature + 6 degrees	Highest temperature - 6 degrees
T_1*	Lowest temperature + 5 degrees	Middle temperature
T_2*	Middle temperature	Highest temperature - 5 degrees
T_onset	Lowest temperature + 1 degrees	Highest temperature - 11 degrees
T_onset1	T_1 lower bound - 5 degrees	T_1 higher bound - 5 degrees
T_onset2	T_2 lower bound - 5 degrees	T_2 higher bound - 5 degrees
dHm, dHm1, dHm2	2.5 kcal/mol	750 kcal/mol
Ki	1e-3	1e3

* Values can be changed by the user.

2.4. Fitting errors

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

2.5. Fitting residuals

The residuals of the fitting are normalized by dividing them by the standard error.

3. Analyze

3.1. Baseline separation factor

This value is useful to compare the height of the unfolding transition for the equilibrium or empirical two-state unfolding models and is calculated as

$$BS(S, k_u, T_m, b_u, k_n, b_n) = 1 - \frac{6*S}{k_u T_m + b_u - k_n * T_m - b_n} \quad (14)$$

where S is the standard error of the estimation.

3.2. Protein stability score

After fitting a model, a protein stability score is provided which can be used to sort the conditions.

Model	Score
Equilibrium two-state	$\Delta G\ of\ unfolding\ at\ 298.15\ K$
Empirical two-state	$distance((T_m; T_{Onset}), (0; 0))$
Equilibrium three-state	$\Delta G\ of\ unfolding\ at\ 298.15\ K$ $(\Delta G\ of\ reaction\ N \leftrightarrows I\ +\ \Delta G\ of\ reaction\ I \leftrightarrows U)$
Empirical two-state	$distance((T_{m1}; T_{Onset1}), (0; 0))\ +$ $distance((T_{m2}; T_{Onset2}), (0; 0))$
Irreversible two-state	$-\ log(k_{irrev}(298.15\ K))$

References

Kotov, V., Mlynek, G., Vesper, O., Pletzer, M., Wald, J., Teixeira-Duarte, C. M., Celia, H., Garcia-Alai, M., Nussberger, S., Buchanan, S. K., Morais-Cabral, J. H., Loew, C., Djinovic-Carugo, K. & Marlovits, T. C. (2021). *Protein Sci.* **30**, 201–217.

Contact details

For further assistance, please contact us:

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

Packages

MoltenProt is possible thanks to:

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

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

R package viridis: Simon Garnier (2018). viridis: Default Color Maps from 'matplotlib'. R package version 0.5.1. https://CRAN.R-project.org/package=viridis

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

R package pracma: Hans W. Borchers (2019). pracma: Practical Numerical Math Functions. R package version 2.2.9. https://CRAN.R-project.org/package=pracma

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

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

R package xlsx: Adrian Dragulescu and Cole Arendt (2020). xlsx: Read, Write, Format Excel 2007 and Excel 97/2000/XP/2003 Files. R package version 0.6.3. https://CRAN.R-project.org/package=xlsx

R package reshape2: Hadley Wickham (2007). Reshaping Data with the reshape Package. Journal of Statistical Software, 21(12), 1-20. URL http://www.jstatsoft.org/v21/i12/.

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

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

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

R package tableHTML: Theo Boutaris, Clemens Zauchner and Dana Jomar (2019). tableHTML: A Tool to Create HTML Tables. R package version 2.0.0. https://CRAN.R-project.org/package=tableHTML

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

R package remotes: Jim Hester, Gábor Csárdi, Hadley Wickham, Winston Chang, Martin Morgan and Dan Tenenbaum (2020). remotes: R Package Installation from Remote Repositories, Including 'GitHub'. R package version 2.1.1. https://CRAN.R-project.org/package=remotes

R package devtools: Hadley Wickham, Jim Hester and Winston Chang (2020). devtools: Tools to Make Developing R Packages Easier. R package version 2.3.0. https://CRAN.R-project.org/package=devtools

R package shinyjs: Dean Attali (2020). shinyjs: Easily Improve the User Experience of Your Shiny Apps in Seconds. R package version 1.1. https://CRAN.R-project.org/package=shinyjs

R package data.table: Matt Dowle and Arun Srinivasan (2019). data.table: Extension of data.frame. R package version 1.12.8. https://CRAN.R-project.org/package=data.table

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

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

Baptiste Auguie (2019). egg: Extensions for 'ggplot2': Custom Geom, Custom Themes, Plot Alignment, Labelled Panels, Symmetric Scales, and Fixed Panel Size. R package version 0.4.5. https://CRAN.R-project.org/package=egg

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

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

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

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

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

Python package natsort: https://natsort.readthedocs.io/en/master/