**Static Full Growth Model Structure**

**Directory**

FullStaticGrowthModel

**Class**

* Gompertz
* Huang
* Baranyi
* Buchanan

Structure

ModelName(maxCount, InitialCount, Rate, LagParameter, time)

Properties

InitialCount: double, single value

maxCount: double, single value

Rate: double, single value

LagParameter: double, single value

time: double, array

**Data Flow and Logic**

***Frontend***

* Enter data (X, Y)
* Create two buttons, one for submitting data, and the other for clearing data
  + For clearing data, add an alert to warn the user and need a confirmation (Y/N).
* Click to submit data
* Check alphanumerical entries
  + Write a JavaScript to check if all data are numerical (Y/N).
  + If Y, pass. If No, send an alert if the data contain non-numeric values
    - “Warning(!) The data contain at least one non-numeric entry. Please verify!” OK (click to disappear).
    - Column1 and row 1 get focus
* Check Y for log or ln values
  + Y can be log CFU or ln CFU, but they must be checked sending to the backend for calculation. The calculation must be in ln CFU for Huang and Baranyi models. It is not necessary for Gompertz and Buchanan models.
  + Write a JavaScript code to check if Y is in log or ln format – (this can be used to all growth models)
    - If min(Y) is < 6.0 and max(Y) is < 10.0, send an alert box message: “Warning (!) - The colony counts appear to be in log CFU. They must be converted to ln CFU for data analysis.
    - Do you want to convert the data to ln CFU?
    - Yes/No
    - If Y, convert
    - If no, keep data format.
    - Create a “state” parameter called “YState” (Boolean) to register Y as “log CFU” or “ln CFU”, which can appear on the corner of the plot to be created.
    - Get data as an array (X, Y)
* Plot data for visual inspection (the data in ln CFU for Huang and Baranyi models, in long or ln CFU for Gompertz and Buchanan models)
* Create a UI for model selection (Drop down menu, dialog, or tree, mutually exclusive)
  + Once a model is changed, plot is clear, and Raw data are replotted.
* Create a JavaScript code for four full growth models (to be called when needed).
  + The same model structure in JavaScript
  + ModelName(maxCount, InitialCount, Rate, LagParameter, time)
  + Accept min(Y) as InitialCount, fixed
  + Accept max(Y) as maxCount, fixed
  + Accept max(X) as maxX; use linear interpolation to create an array of X\_plot, containing 500 values, X\_plot (array, 500 values) = [0., maxX]. This is for plotting purpose.
* Select a full model
  + Pop-up a dialog for the full model
  + Create to UIs for Rate and LagParameter
  + For each UI, use a slider to adjust value; use a spinner to show value
    - For each change in the slider, a change will be made to spinner, vice versa
      * For rate (min, initial, max) = 0., intialGuessValue, maxValue, step:
        + initialGuessValue = (maxY-minY)/maxTime
        + maxValue = 20\*initialGuessValue
        + Step = maxValue/100.0
      * For LagParameter (min, initial, max) = intialGuessValue, maxValue, step:
        + initialGuessValue = maxTime/10.0
        + maxValue = maxTime
        + step = maxValue/100.0
  + For each change in the parameter, call the model (frontend) will be called to calculate the values, call model- ModelName(InitialCount, maxCount, Rate, LagParameter, time)
    - calculate X\_plot and Y\_plot, and plot (X\_plot, Y\_plot) along with raw data (X, Y).
    - Submit Model
  + A submit model button is needed
  + Once it is clicked, get the last values of (Rate, and LagParameter) and put them in an array or list Parameter [Rate, and LagParameter]
  + Send Parameter, Raw Data, and model choice to the backend for calculation [P, rawData, model], P: array, rawData: array, model: String. All these can be packed in an array.

Backend

* Get data passed from the Frontend, calling them incoming\_data
* Unpack incoming\_data. Incoming\_data[0] = [parameters]; Incoming\_data[1] = [rawData]
* Process data analysis using each wrapper
* Send results to Frontend for presentation