Documentation for Growth Curve Analysis Functions - Goodwin Gibbins

Summer 2010

The growth curve analysis functions (aka GCAF) (not yet completed) provide a centralized method for the organization and analysis of measured bacterial growth curves and associated metadata.

The functions are not ready for immediate widespread application; they currently only provide relatively basic analysis of growth curves and their use requires significant user participation and (possibly) error bugging. However, they have been constructed with an eye towards continued development and provide a strong framework for exploration of growth parameters. This document aims to explain the extent of the current functions and provide suggestions for further development.

1 The Vision

The ultimate goal of this project is a web-based pipeline for robust analysis of growth curves to provide biologists with meaningful numerical interpretation of their curves (designated "parameter"). The web application would also serve as a repository of past curve data/metadata to allow for cross-experiment comparison/analysis.

The programming language R has been suggested for implementation of the GCAF since it free and relatively easy to use with web-based analysis. With this in mind, the goal of the initial phase of the project has been to create a distributable pseudo-package in R of the growth curve analysis functions (GCAF).

A successful final product would be expected to have the following abilities:

- Be intuitively navigable and usable.
- Provide methods for the most commonly requested analysis techniques and allow for addition of other measurements.
- Derive measures of a growth curve which are useful to the biologists: that have some biological interpretation, that can successfully/consistently distinguish between different environmental/physical perturbations
- Provide opportunity for database-wide analysis to provide a wider view of the effects of changes on growth curves.
- Return raw data, derived parameters and visualizations for further analysis.

2 Using the Growth Curve Analysis Functions

At its current stage, the GCAF package is not always intuitive and contains limited error-handling facilities; this section seeks to outline proper techniques for both users and developers and should be used in conjunction with the package help files. The GCAF code is annotated and includes `# TODO` and `# USER` comments which idicate areas for developers or users to pay special attention to respectively.

2.1 Set-Up: Loading the Database and Function files

1. Acquire the ZIP file containing the database, functions and documentation, unzip, and save locally.
2. Open GrowthCurveAnalysis folder and the `gControl.r` file. Change the default `root.folder=''<some path>''` to the file path for the downloaded database folder in R formatting (i.e. forward slashes).
   ```
   > gControl <- function(..., root.folder = ''c:/Users/Documents/GrowthCurves/Database'', ...)
   ```
3. Until GCAF has been made into a package, open the function `gFunctions.r`, update `gcaf.folder`, and source the function in R. This will load GCAF functions and allow use of functions (without proper documentation help support).

2.2 Adding an Experiment to the Databases

The database is structured around file folders containing information for each experiment (ex: one run in one bioscreen machine) which are uniquely identified by `Date` and `Initials`. The experiments are initially described by the raw data and a database information file provided by the user. The current format accepts bioscreen results files (though this could be changed - other formats could be coerced into this, etc) and requires a user-made information file.

To add an experiment to the local database:

1. Choose `Initials` and `Date` to identify the experiment. If two growth curve experiments were run on the same day, treat as separate experiments and use consecutive dates. April 15, 2010 would be `20100415`. Initials should be consistent with past experiments (e.x. ``FYL'', ``LP'', ``ST'', ``KB'').
2. Create a folder in the local database folder for the new experiment, (e.x. ``...\Database\201000415ST''
3. Copy the bioscreen-style results file to the experiment folder with appropriate naming. e.x. ``...\Database\20100415ST \ResultsST20100415.csv''. (See appendix for formatting).
4. Open the layout file, select cells, copy to a new excel document and save as a .txt file (tab-delimited) with appropriate naming. e.x. ``...\Database\Layout\LayoutST20100415.txt'' The copying is because sometimes straight conversion to tab-delimited leaves empty cells. An error-check to remove empty cells would be a good addition. (See appendix for formatting).
5. Open `gControl.r` and confirm that the default `database.columns` vector contains the appropriate meta-information specifications for your experiment. You may need to add a section depending on perterbations (should become centralized in web pipeline).
6. Run `gPrepareDatabase` to create the bare-bones tab-delimited spreadsheet with `Intials`, `Date` and `Well.Name` (label) filled in. The file will be saved as the iformation file in the experiment folder, e.x. ``...\Database\20100415ST \InfoST20100415.csv''
7. Edit the file in Excel (or similar) and fill in information about each well by translating `Well.Name` or based on global experiment defaults (such as temperature or media). Naming of meta-information must match other experiments so refer to the appendix (reference) for explanations of naming and sections or other database files (this appendix should also be centralized). Blanks wells or information columns should be left as NA. Note: The cell-highlighting function in excel is has been useful to visually separate between "NA", already entered information and empty wells.

2.3 Analyzing a Set of Growth Curves

The analysis of growth curves follows a simple path of searching for desired growth curves, analyzing the curves then exporting or displaying the reusulting information. The steps are as follows:

1. Create `matches` and `not.matches` objects and provide to `gChoose()`. Refer to documentation for specifications.
```
> matches <- list(Media=``CM'', Background=``NRC-1'', Temp=c(``37'', ``NA''))
> not.matches <- list(Date=``20100707'')
> chosen.wells <- gChoosematches, not.matches
```
2. Run `gAnalysis` on `chosen.wells`. Note: Checking the `length(chosen.wells)` first is recommended for time estimate.
```
> cm.nrc1.37 <- gAnalysis(chosen.wells)
```
3. To summarize the information of the wells analyzed, use `gSummary(cm.nrc1.37)`. This can be useful for checking that no unwanted wells are in analysis.
4. To view plots of individual wells, use the plot function. See documentation for further options.
```
> plot(cm.nrc1.37, 1)
> plot(cm.nrc1.37, ``all'', graph.log = T)
```
5. To view the distribution of wells against parameters, use `gSlice(cm.nrc1.37)`. See documentation for further options.
6. To save a spreadsheet containing meta-information and analyzed parameters, use `gExport(cm.nrc1.37, name=``cmNRC137'')`.

2.4 Adding Parameter Estimator Functions

One of the strengths of the GCAF framework is that it facilitates easy addition of algorithms for parameter estimates. To add a parameter to `gAnalysis`:

1. Write a function which uses already defined parameters, spline-fitted data, time or derivatives or the raw data and time information to define a numerical parameter. If the function is of appreciable length, save in the `gFunctions` folder with naming `gFitXXXXX.r` where the XXX help identify the parameter and add path to `gFunctions.r`.
2. Open `gSplineFit.r` and and above the line `# USER: ADD PARAMETER ESTIMATE FUNCTIONS HERE`. Also add the parameter to the list of parameters returned by the function.
3. Open `gControl.r` and add new parameter to end of `...parameter.columns=c(...., XXXX)`.

3 Wishlist

Develop better parameter estimators - Currently, parameter estimation is unsuccessful at capturing meaningful biological quantities reliably - often noise, length of experiment, growth the the second phase, etc cloud results. The bulk of further work on this project will probably be directed towards design and implementation of "smart" analysis techniques such by using Hill Functions (idea from Alex Ratushny), as has been experimented with in gFitHill (see In Progress folder). Another possibility is to continue to develop the package and streamline the process of adding estimators, providing a vehicle for future parameter developers.

Error Checks - Depending on the use of these programs as skeletons for further development or a distributable package, error checks will need to be incorporated. One useful one would be for empty cells in layout files.

Expand Database - Save and use `gAnalysis` objects to speed up graphing of certain sets of data.

Develop Visualization - The plotting function has been developed as an aid to the design of parameters and is not completely user-friendly. The plot itself could be enhanced, and other options could be added, such as graphs of distribution of parameters in different sets of curves (with coloring like gSlice, but one graph per parameter), a graph showing all plots using rows of boxes colored based on OD, and basic SVG/Log plots which are missing from the current plot function.

Renaming Variables - Due to the pace of development of this set of functions, naming is not yet ideal, both of individual functions and variables and of the package itself. Possible advancements include centralizing variables such as `g.analysis` or `num.wells`, renaming GCAF itself, renaming Database files to Information files (more accurate since they are only the information piece. This is not a priority but would be a boone for ease of future development.

Website Design - Obviously, this project would benefit from work on an online web pipeline for more widespread and organized access to the to-be-developed parameter analysis.

Clean Up - Clean up and complete this and other documentation files. A list of acceptable info needs to be added to the InfoandParameters file, documentation needs to be edited, and in code comments also need work.

Summary Diagram - A flow diagram of the functions and their interactions might be useful to users of the GCAF package.