KinFold v2.2

January 14, 2009

Joshua S. Martin
Alain Laederach
DOD-Wadsworth Center

The software and data examples that accompany this documentation may be downloaded from http://simtk.org/home/KinFold. This write-up may not be complete and should be used as a general guide and not a complete description.

1 Introduction

This package contains a series of matlab scripts to analyze and model time- progress curves measured with local probes of structure. Local probes of macromolecular structure are measurements that are sensitive to the environment of a relatively small region within a macromolecule. These include, but are not limited to, NMR deuterium exchange and shift perturbation analysis, Fluorescence Resonance Energy Transfer (FRET), and RNA/DNA protein footprinting. The separate transitions reported by individual probes yield unique insight into folding intermediates. While simultaneous acquisition of many unique local transitions provides a cornucopia of information, creating an accurate global de-scription of folding that remains faithful to local details is very challenging.

The package requires Matlab which one can obtain from the Mathworks (http://www.mathworks.com), although some of the scripts will also work with Octave (http://www.octave.org). This is not necessarily intended as "easy to use" software, but rather to provide the basic tools to carry out an analysis similar to the one reported in Laederach et al., "Local kinetic measures of macromolecular structure reveal partitioning among multiple parallel pathways from the earliest steps in the folding of a large RNA molecule," JMB 2006. **algorithms paper** This manuscript details the algorithms implemented in the accompanying code. A graphical user interface named kinfold.m has been included as a wrapper for some of these scripts.

Any questions, bug reports maybe reported to Josh Martin (jmartin at Wadsworth.org) or Alain Laederach(alain at wadsworth.org).

2 Methods:

Below are function definitions and usage examples for the main scripts to carry out a complete analysis of a data set. By inputting the commands in matlab that are shown by

»[output arguments]=matlabcommand(input arguments)

you can get through the example data provided here. The input and output arguments are explained for each command individually. The input and output arguments are separated by commas. A number of input arguments are optional for several commands as described in the commands individual description. The example data here can be processed up to the flux analysis without the need of a supercomputer however other data sets may require the use of a supercomputer at particular points as noted.

For more information on a particular function the actual file can be open to view the code or help can be run on the command as so:

»help matlabcommand

2.1 Data entry:

The Matlab function used for data entry is the readxlsfootprintdata.m . To load your data, type in "readxlsfootprintdata" followed by the filename of the data in parentheses immediately after "readxlsfootprintdata". The filename is optional. The function will however prompt you for filename if not given. The format of the data must be the excel file format (.xls). An example is shown:

» [time_bins, interp_data_ave, res_labels]= readxlsfootprintdata(filename, display);

The variables in the brackets are the outputs of the readxlsfootprintdata function. This function will read in an plot data from an excel spreadsheet. The data is histogramed and vectorized for k-means clustering. To see how the readxlsfootprintdata function works, type the following example into the command line in Matlab. Example:

» [time_bins, interp_data_ave, res_labels]= readxlsfootprintdata('dataeg.xls');

Again note the format of the data in the excel file dataeg.xls, it is important that this format if you are inputting your own data. If display is set to true, the program will show a number of figures corresponding to the number of nucleotides and the extrapolated data.

2.2 Gap Statistic:

The compute_Gap function is used to analyze the data by using the gap statistic. Gap Statistic is a published algorithm, see http://www-stat.stanford.edu/ tibs/ftp/gap.ps, to estimate the number of clusters in k-means clustering.

The format that compute_Gap function reads is the following.

khat=compute_Gap(t,sim_data,Krange,n_reps,B)

t is a vector containing the times of the time points in log format. sim_data is the data to be clustered as an txm matric. Krange is the range of k's you want to consider for k-means clustering. n_reps is the number of reps during k-means clustering of the random data, set at 10. B is as defined in the gap-statistic paper, it is a parameter for the number of random data sets to generate, set at 100.

An example of the function is shown below. To see how the function works, type the following in the command window.

» khat=compute_Gap(time_bins, interp_data_ave, [2 6], 10, 100)

This will take a couple of minutes to run, but will return a value of khat=3 as shown in the picture below.

The khat value of 3 means that the Gap statistic has estimated that the data has three clusters and this value should be used in the clustering, which is the next step. Note that for this and the next step you need to have the Statistics Toolbox installed. To check, if you can type the following:

» kmeans

??? Error using ==> kmeans
At least two input arguments required.

If that is the error you get, then the Toolbox is installed. If you get a different error message, it will have instructions on how to install that toolbox.

If you do not want to install this toolbox or are using Octave you will need to skip the clustering step. The results of the clustering have been stored so for this demonstration you can continue. The results are stored under "dataeg.mat".

2.3 Clustering:

Now we will cluster the data using kmeans clustering. The function used for clustering is kmeans. The following is the example of the format used for kmeans. The kmeans function is Matlab function. If you have any questions for

»[IDX, C, SUMD, D] = kmeans(X, k, varargin)

X stands for the matrix while k stands for the number of clusters. The varargin variable represents optional variables that can be added. Type in the following to see what the kmeans function does.

» [IDX, C, SUMD, D] = kmeans(interp_data_ave', khat, 'Distance', 'cityblock', 'Replicates', 100, 'EmptyAction', 'singleton')

In the example above, 'Distance', 'cityblock','Replicates', etc. are all optional variables. To learn more about these variables, see the Matlab help on kmeans. Overall kmeans(X,k) returns distances from each point to every centroid in the N-by-K matrix D. The optional input arguments have been optimized for this type of clustering.

The results of the clustering for this example can be seen using the following two commands:

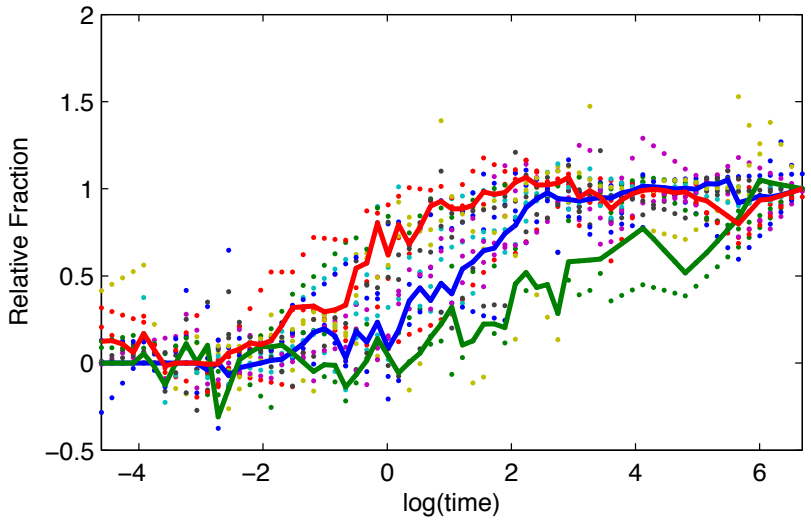

Figure 1: raw data

» load visualization.mat
» display_clusts(IDX, res_labels, C, exp(time_bins), 1, interp_data_ave, imagex, offset, residue_locations, 1, 2)

The results are 8 figures. Note that this is specific to the L-21 tetrahymena ribozyme, as this function displays the data on the secondary structure.

2.4 Kinetic Model Fitting:

There are two components to the Kinetic Model Fitting portion of KinFold v.2.2. The first component is accomplished through the search_reductions2 function. The search_reductions2 function test all possible models and determines which models are good and are to be fitted. To run the function, you must define 4 variables; C, time_bins, area_cutoff, and filename. C and time_bins were already determined earlier in the KinFold v2.0 program. The data you would like to evaluate is defined by giving the filename to the data. The area_cutoff component must be defined by the user. The area_cutoff variable is defined as a percentage or a value less than 1. If you do not define the area_cutoff, search_reductions2 will automatically define area_cutoff as 0.0. With area_cutoff defined as 0.0, any curve that falls below 0 will be eliminated along with all of the curves in that model. A good model has no curves that does not exceed the area_cutoff. The farther away the area_cutoff value is from zero, the more curves that will surpass the selection criteria.

Below is the format used for the search_reductions2 function.

» [new_C, time_bins, out]=search_reductions2(C, time_bins, area_cutoff, filename)

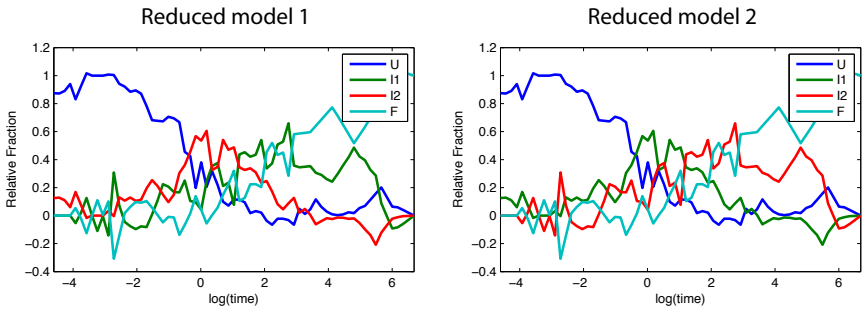

Figure 2: reduction models

The search_reductions2 function saves the good models. The models are saved in the following format 'filename-model-number.mat' unless the filename argument is not given then filename is set equal to 'data'.

The search_kspace4 function fits all of the good models found by the search_reductions2 function. It will generate the best k values in the given amount of time. The search_kspace4 function randomly selects a random set of initial k values every time the function is run. For experimental data, we suggest the function be run for at least 16 CPU hours. The longer the function is run, the better results you will obtain. Because the function randomly selects these k values, the function can be split up over numerous days if desired. This is a very computationally intense function. One may wish to run this function on a supercomputer or a compute cloud such as the Amazon Elastic Compute Cloud found at this address (http://aws.amazon.com/ec2/). This function must be run for each of the models found by search_reductions2 function.

The format in which the search search_kspace4 function uses is shown below.

»search_kspace4(name,hours,display)

There are three input variables in which the search_kspace4 function reads. The name variable is the filename of the models must be given. If you do not give a filename, the function will prompt for the file. The hour and display input variables are also optional. If the hour variable is undefined, then the function will run for a total of 1 hour. The display function can either be turned on or off. If you do not type in " 'yes' " for the display variable, the function will not show the output. The function will run faster if the display variable is off.

The program search_kspace4 saves the resulting optimization of the models in a file called name-results.mat where name is the name of the file containing the model being looked up. If the file has already been created, it will add the other optimizations to the file. This allows for additional optimization runs just by running search_kspace4 on the same model.

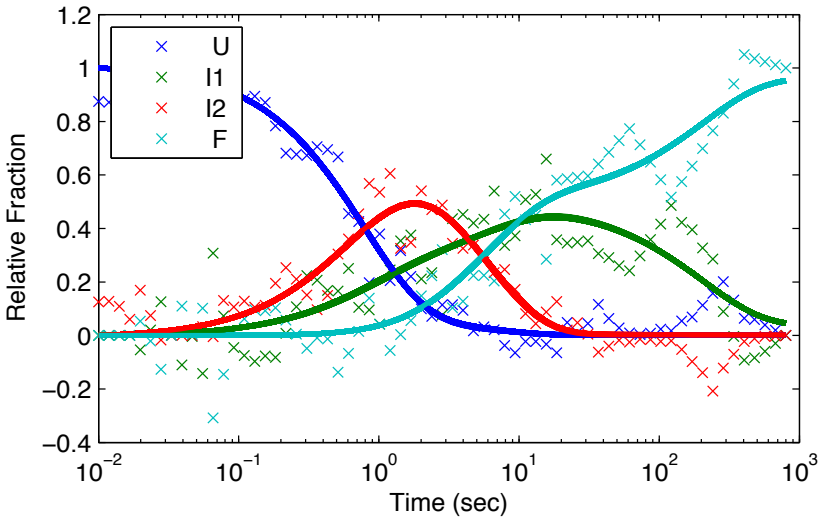

Figure 3: fitted model

2.5 Optimization visualization:

To next step in evaluating the data is to sort the fitted graphs determined from the search_kspace4 function. The "analyze_mat_dir2" function will sort the fitted models from the best to the worst. You can either input the filenames to the models individually or you can insert a directory name in which you set up that may contain all of the files. The format of the function is shown below. To use analyze_mat_dir2 function, type the following into the command window.

» tot_val=analyze_mat_dir2(filename)

The make_plots_interest2 function presents the best fit graphs to the data. However, before you see the best fit graphs, you must run the function to determine one of the function's input, the num_cut input. Without the num_cut value, the output values (mean_K) are the rate constants of the best fitting model with K(1,2) corresponding to the rate from U->I1, K(1,3)-> U->I2 etc. The num_cut variable is the last number that is less than 10 percent difference way from the best fit error (or the first tot_val value). The num_cut variable is an optional input so the function will initially run without it. The num_cut variable is determined by typing in the following command into the command window. The

»sum(tot_val<=tot_val(1)*1.1)

If you do not have tot_val saved as a variable, you can determine the num_cut value by analyzing figure 1. The num_cut value will be the value in which you the increase

in value from the baseline results.

Therefore, once num_cut is calculated, the best fit plots can be generated. The following is code for the make_plots_interest2 function. The filename input is optional. You will prompted for the filename if not given.

» [mean_K,error_K]=make_plots_interest2(filename,num_cut)

3 Flux analysis

Flux analysis also requires a significant amount of time to compute accurately, but for the purposes of this demonstration the following commands can be issued:

» get_fluxes_t(filename,1)

This command should be repeated at least 100 times to get better sampling, each time incrementing the argument by one. (About 100 CPU hours)

3.1 Pathway Analysis:

This command analyzes the result of all the pathways data stored in the pathways folder.

» pathways=analyze_state_pathways;

3.2 Pathway Visualization:

The most common pathways can be vizualized by issuing the following command.

» [clust_pathway,diff_pathway_vect]=pathway_analysis(pathway);

This will show a histogram of the fluxes through the major pathways.

The analysis presented above can be repeated with a different data set but will require the use of a super computer (between 200-300 nodes) for the computationally expensive steps. The scripts for generating massively parallel runs are nto included in this distribution as they are specific to the setup of the machine. For help with distributed computing setup, feel free to contact Alain Laederach (alain at helix.stanford.edu).

4 kinfold GUI

This is the kinfold graphical user interphase section for running readxlsfootprinting.m, compute_Gap.m, kmeans.m, search_reductions2.m, search_kspace4.m and make_plots_interest2.m. This gui may have some bugs however it has been tested with the dataeg.xls file included with this software. Each individual section of Kin Fold can be run separately however every piece is interconnected as shown in Figure 4.

Process XLS file

1)choose xls file from prompt by pressig choose xls file button

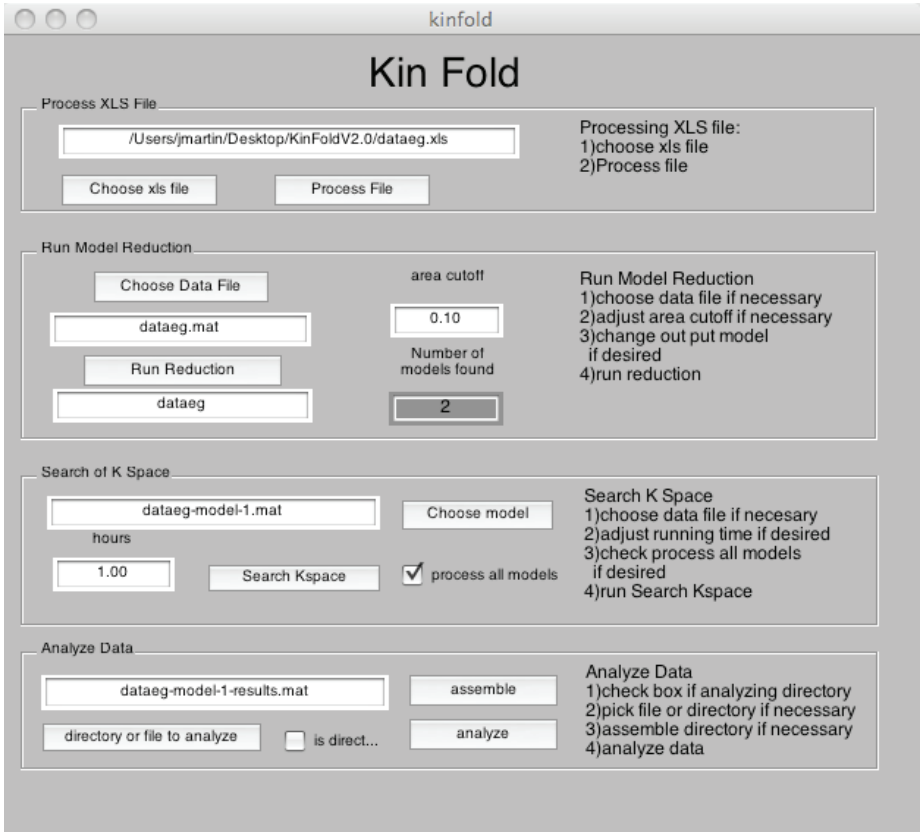

Figure 4: Snap shot of the kinfold.m gui for running readxlsfootprinting.m, compute_Gap.m, kmeans.m, search_reductions2.m, search_kspace4.m and make_plots_interest2.m. This is as it is shown run on OSX, the window will look different on other operating systems.

2)press process file button - this processing will take a while as the readxlsfootprinting.m, compute_Gap.m, and kmeans.m software

Run Model Reduction

1)choose data file

2)adjust area cutoff if necessary

3)run reduction button

Search of K Space

1)choose model to process

2)select to process all models - adjust number of models found

3)adjust number of hours to run search on each model to be processed

4)run search Kspace

Analyze Data

1)select if it is a directory to analyze

2)select file or directory to anlyze

3)a directory can be assembled from a series of processed models with assemble

4)analyze data will save a file starting K that contains the constants.