Engineering Biology

Research Article

Debugging experiment machinery through time-course event sequence analysis

ISSN 2398-6182
Received on 9th March 2017
Revised on 10th May 2017
Accepted on 26th May 2017
doi: 10.1049/enb.2017.0008
www.ietdl.org

Christopher R. Reynolds ✉, *Kealan Exley*, *Matthieu A. Bultelle*, *Inaki Sainz de Murieta*, *Richard I. Kitney*

Centre for Synthetic Biology and Innovation and the Department of Bioengineering, Imperial College London, London, UK
✉ *E-mail: cr308@imperial.ac.uk*

Abstract: This application note describes an open-source web application software package for viewing and analysing time-course event sequences in the form of log files containing timestamps. Web pages allow the visualisation of time-course event sequences as time curves and the comparison of sequences against each other to visualise deviations between the timings of the sequences. A feature allows the analysis of the sequences by parsing selected sections with a support vector machine model that heuristically calculates a value for the likelihood of an error occurring based on the textual output in the log files. This allows quick analysis for errors in files with large numbers of log events. The software is written in ASP.NET with Visual Basic code-behind to allow it to be hosted on servers and integrated into web application frameworks.

1 Introduction

Synthetic biology uses a lot of different equipments (e.g. in the context of automation) that produces output which is logged. The size and number of log files produced by typical laboratory equipment (e.g. liquid handling robots) often become so large that the files are difficult to search by human operators. In the field of synthetic biology, the consistent replication of a workflow is crucial for the reliable characterisation of parts [1]. Enforcing consistency between workflows, detecting deviations from normal operation and identifying the root cause of deviations is important to ensure good data. The use case is intended to be the comparison and error analysis of large time-course event log files of the kind that are generated by machine hardware. This has applications in many fields of engineering that involve machinery, but synthetic biology introduces the additional problem of biological material, which can introduce unpredictable occurrences and effects into the system. This requires a flexible analysis tool which assists the user both in comparing sequences for differences and detecting errors using a non-deterministic method.

2 Time-course event sequences

A time-course event sequence (also referred to as a point event sequence, temporal event sequence or a categorical time series) is a sequence of discrete temporal events, each represented by a status and a timestamp. These sequences are generated across data analytics, particularly in the field of engineering and existing analytics platforms exist to analyse them [2, 3]. This paper describes a set of analytics (sequence comparison, error detection and time log averager) used to analyse time-course event sequences of outputs from synthetic biology experiments, specifically the characterisation of constitutive promoters. Four of our time logs have been included as example data in the github directory. The time log files can be thought of as time-course event sequences, with each event associated with a text description of the events.

In an ideal situation, each time-course event sequence for an experiment controlled by the same set of instructions would be identical with each event occurring at the same point in time from the start of the run. By comparing the deviation between where events have occurred in log files generated by the same set of instructions, deviations between the timings can be detected. These can provide a useful indication of where potential errors are occurring.

Detecting deviation between time-course event sequences can be useful for asynchronous workflows because of the way such workflows try to force all events to a pre-set timeline. The approach is also useful for synchronous workflows, where it can be used in the early debugging stage of workflows, by identifying where major deviations between runs are occurring.

3 Materials and methods

The time-course event software is written in ASP.NET with VB.NET code-behind. ASP.NET was used because it is a widely used open-source web application framework, which allows it to be hosted on a server and made available to a wide number of users on a network through any standard web browser interface.

The code utilises Microsoft Chart Controls for Microsoft.NET Framework 3.5, which can be downloaded [4], if not already installed.

The examples used in this paper relate to real characterisation experiments for constitutive promoters using a standard protocol [5]. The data shown in the plots is real data from a laboratory robot set-up that includes incubators, plate readers, a robot arm and a benchtop liquid handling device. However, it should be noted that the software can be applied to any equipment that produces text log files.

The programme accepts as input any plain text file that consists of a series of timestamps each followed by text. The text can be over multiple lines, but must not be on the same line as the timestamp.

4 Sequence comparison

The software is centred around two pages: (i) one page to view and compare time-course event sequences, and view the plots and (ii) the other page shows the combined time logs to produce a reference time-course event sequence.

The graph viewer has two modes. The first is to visualise a single point sequence from a time log, plotting the number of events against time. Regular expressions are used to parse a log file and extract all timestamps, together with associated text. Multiple selected sequences are plotted on the same graph, so that they can be compared (Fig. 1) (a maximum of five sequences can be plotted on the same graph).

In Fig. 1, three protocols are visualised. The magenta and teal plots use the same protocol, the navy plot is for a protocol running at a faster pace. The plots show an initial steep curve representing a high number of events over time as the equipment initialises, then a period of no events as the samples are incubated. At around 6000 s, the cyclical phases of plate reading, dilution and incubation start. There is a further high number of events over time as the reader and distillation machine initialise, and then a characteristic stepped line as the reading and distillation events occur between longer incubation periods. The differences between protocols and the phases of the process can be clearly visualised from the graph including possible deviations from consistency in the protocol such as a delay in events around 20,000 s on the magenta plot.

The second graph viewer mode measures and displays the cumulative deviation between graph plots (Fig. 2) and then plots

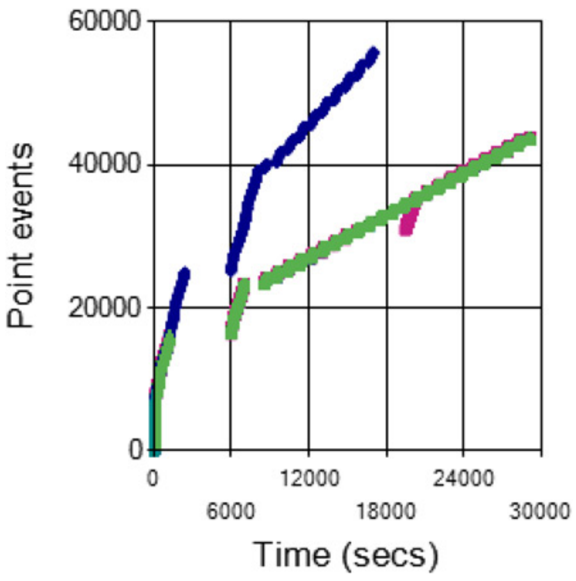


Fig. 1 *Time-course events versus time plots for three time logs, all for experiments characterising constitutive promoters*

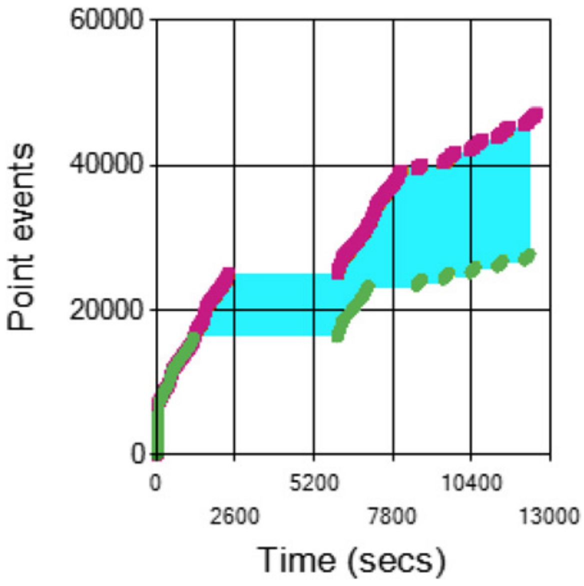


Fig. 2 *Calculation of the time deviation between two event sequences. The plots compare two time-course event sequences for the same experimental protocol characterising constitutive promoters. The area of cumulative deviation (shaded in cyan) is calculated*

the cumulative deviation against time (Fig. 3). This mode is limited to comparing two time-course event sequences against each other.

The deviation comparison of time-course event sequences is performed by the following method:

(i) The user selects two time-course event sequences: a *comparison sequence* and a *reference sequence*.
(ii) The start of both of the two time-course event sequences is synchronised.
(iii) The algorithm moves through the time-course events in both sequences chronologically.
(iv) If the time-course events are out of sequence, the cumulative time deviation is increased by the time difference between this event and the previous chronological event (in either sequence). This is equivalent to multiplying time difference on the *x*-axis by event number difference on the *y*-axis, and gives the area highlighted in cyan in Fig. 2.
(v) The cumulative time deviation is calculated up to the end of one of the sequences.
(vi) Cumulative deviation between the sequences is plotted against time, following synchronisation.

An example plot of a comparison is shown in Fig. 3.

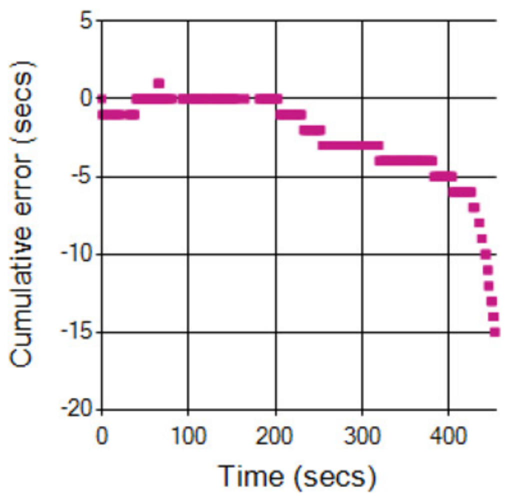


Fig. 3 *Plot displaying cumulative deviation between a reference sequence and a comparison sequence for an experiment that ended in failure after 400 s*

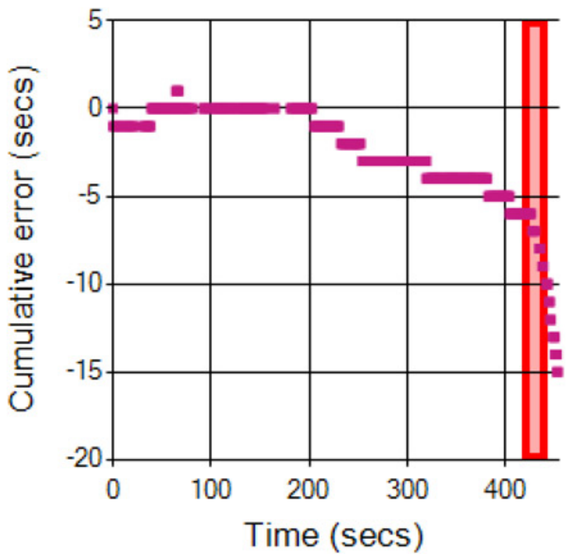
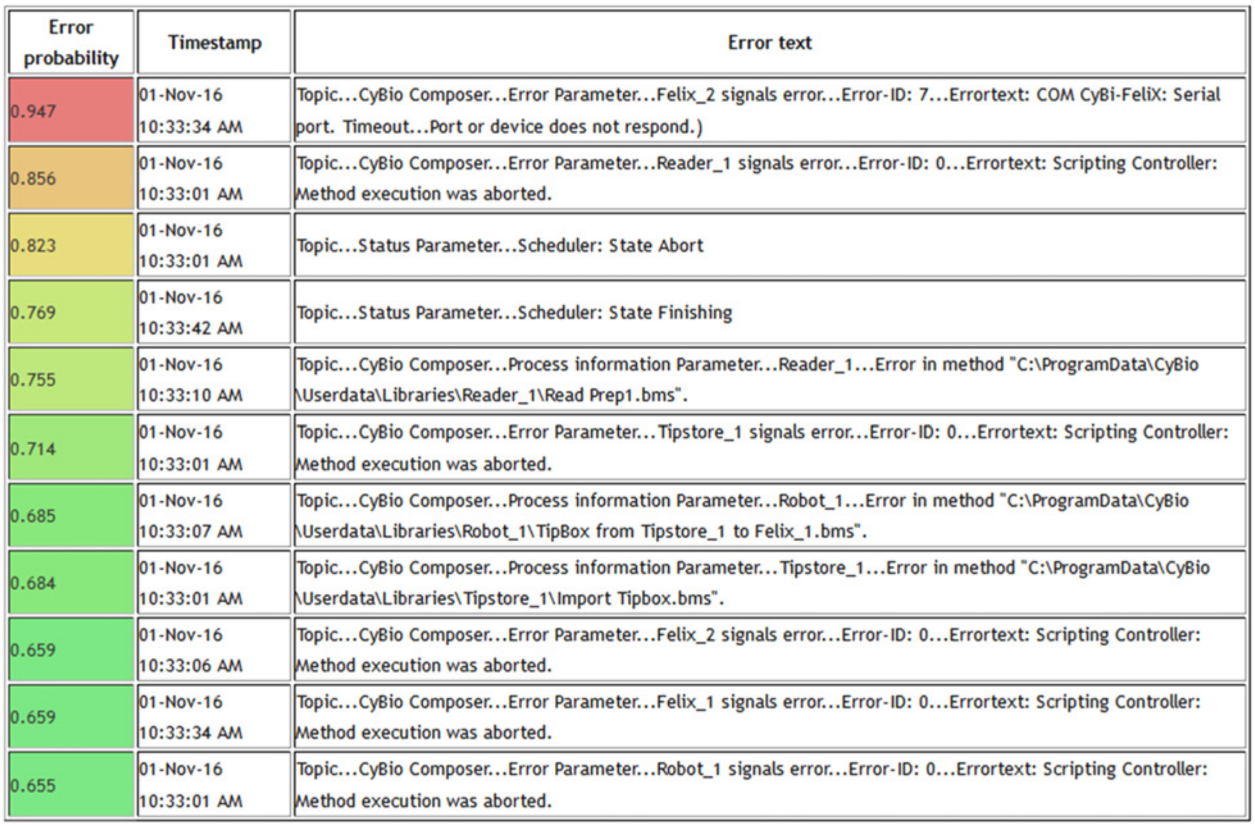


Fig. 4 *Area of the graph selected by a user becomes highlighted in red to indicate the area that the error likelihood analysis has been performed on*



Error probability	Timestamp	Error text
0.947	01-Nov-16 10:33:34 AM	Topic...CyBio Composer...Error Parameter...Felix_2 signals error...Error-ID: 7...Errortext: COM CyBi-FeliX: Serial port. Timeout...Port or device does not respond.)
0.856	01-Nov-16 10:33:01 AM	Topic...CyBio Composer...Error Parameter...Reader_1 signals error...Error-ID: 0...Errortext: Scripting Controller: Method execution was aborted.
0.823	01-Nov-16 10:33:01 AM	Topic...Status Parameter...Scheduler: State Abort
0.769	01-Nov-16 10:33:42 AM	Topic...Status Parameter...Scheduler: State Finishing
0.755	01-Nov-16 10:33:10 AM	Topic...CyBio Composer...Process information Parameter...Reader_1...Error in method "C:\ProgramData\CyBio \Userdata\Libraries\Reader_1\Read Prep1.bms".
0.714	01-Nov-16 10:33:01 AM	Topic...CyBio Composer...Error Parameter...Tipstore_1 signals error...Error-ID: 0...Errortext: Scripting Controller: Method execution was aborted.
0.685	01-Nov-16 10:33:07 AM	Topic...CyBio Composer...Process information Parameter...Robot_1...Error in method "C:\ProgramData\CyBio \Userdata\Libraries\Robot_1\TipBox from Tipstore_1 to Felix_1.bms".
0.684	01-Nov-16 10:33:01 AM	Topic...CyBio Composer...Process information Parameter...Tipstore_1...Error in method "C:\ProgramData\CyBio \Userdata\Libraries\Tipstore_1\Import Tipbox.bms".
0.659	01-Nov-16 10:33:06 AM	Topic...CyBio Composer...Error Parameter...Felix_2 signals error...Error-ID: 0...Errortext: Scripting Controller: Method execution was aborted.
0.659	01-Nov-16 10:33:34 AM	Topic...CyBio Composer...Error Parameter...Felix_1 signals error...Error-ID: 0...Errortext: Scripting Controller: Method execution was aborted.
0.655	01-Nov-16 10:33:01 AM	Topic...CyBio Composer...Error Parameter...Robot_1 signals error...Error-ID: 0...Errortext: Scripting Controller: Method execution was aborted.

Fig. 5 *Machine learning analyses topics from the generated log file within the analysed time period. The algorithm computes a measure of error likelihood and displays the top ranked topics from the log file in a table*

5 Error detection

The output graph of the deviation comparison (Fig. 3) can be analysed by an error detection algorithm. This allows the user to receive visual feedback from the cumulative deviation graph and then advantage of a machine learning approach that can ‘drill down’ into an area identified as problematic and sort the many events occurring in that area by their likelihood of being an error. Use of the error detection algorithm is intended to speedup the search for errors. Rather than having the user check through the messages in the log files around a time point, which usually involves thousands of messages, the algorithm uses a model to identify the most likely source of error.

A support vector machine (SVM) was used to analyse error likelihood. The training data for the error likelihood model was a set of 200 text sentences, a mix of system error messages and common normal system status messages. Error messages were compiled from a set of 15 log files generated by the equipment, with the addition of Linux system error messages. Normal system status messages were compiled from the most common messages that appeared within those same log files. Each sentence was assigned a heuristic weighting between zero and one based on how much the message looked like a crucial error. The plain text training data is included in the github repository as TrainingData.csv.

The words in each error message were tokenised and stemmed (stripped of suffixes [6]). The stemmed words become feature vectors, and the assigned values vector magnitudes, to form a kernel for an SVM. SVM-Light [7] was used to generate an SV model from this data. An ‘error likelihood’ value can then be produced for any sentence by tokenising and stemming the words and analysing them with the computed model. Common differences between British and American English spellings are standardised to American English before being analysed.

Clicking on a point on a graph allows analysis of the log data in an area around the click: a period in time set to 5% of the plot width and centred on the point of the click are taken, and all topics from the log files that fall within this time period are passed to the error detection algorithm for analysis. The algorithm is used to compute an ‘error likelihood’ for all text associated with timestamps in this area of the graph. Fig. 4 shows the portion of the plot being analysed highlighted after being clicked on by the user. Fig. 5 shows a

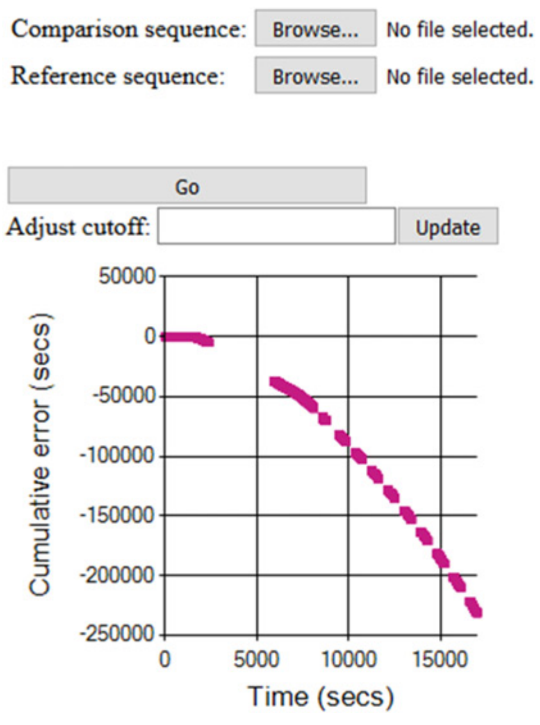
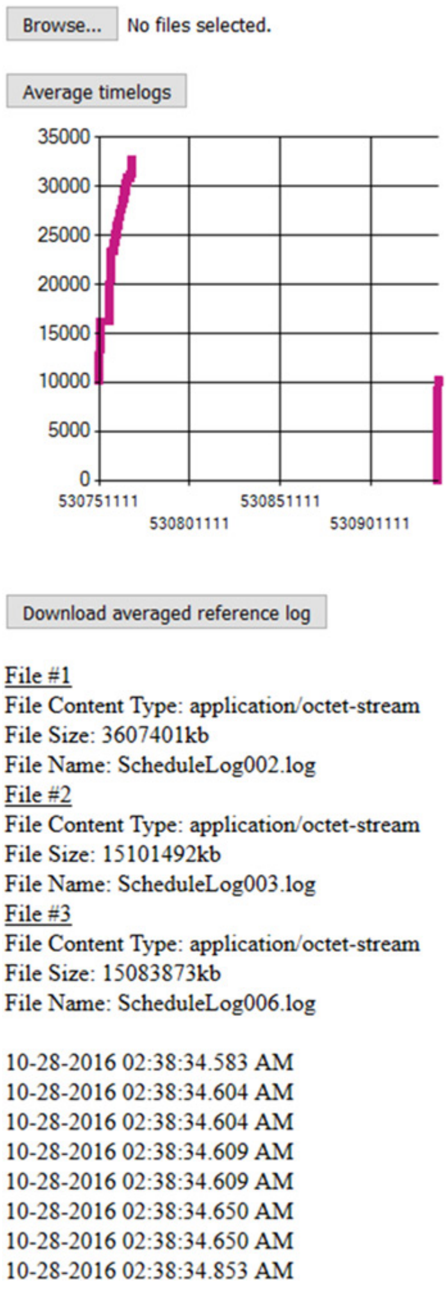


Fig. 6 *Screenshot of the ProcessMachineData.aspx page*



File #1
File Content Type: application/octet-stream
File Size: 3607401kb
File Name: ScheduleLog002.log
File #2
File Content Type: application/octet-stream
File Size: 15101492kb
File Name: ScheduleLog003.log
File #3
File Content Type: application/octet-stream
File Size: 15083873kb
File Name: ScheduleLog006.log

```
10-28-2016 02:38:34.583 AM
10-28-2016 02:38:34.604 AM
10-28-2016 02:38:34.604 AM
10-28-2016 02:38:34.609 AM
10-28-2016 02:38:34.609 AM
10-28-2016 02:38:34.650 AM
10-28-2016 02:38:34.650 AM
10-28-2016 02:38:34.853 AM
```

Fig. 7 *Screenshot of the TimelogAverager.aspx page*

typical output of this section of the programme, demonstrating that an error message indicating an unresponsive port or device and a message indicating an aborted method are ranked as having the two highest error likelihoods.

6 Time log averager

Time logs can be compared against each other to compare the deviation between them. It will be common to compare experiments against a 'reference' log of an experiment known to be successful and error free. It is useful to be able to generate a time log that is averaged from multiple successful time logs to remove unique random error that may occur in individual experiments.

This is used to generate a reference sequence by averaging the timings of events over several time logs. The TimelogAverager.aspx page allows the selection of successive time logs that are averaged as they are added. The resultant plot of time-course events against time is displayed (as shown in Fig. 1), and the user can download the resultant log as to use as a reference file.

7 File listing

In its initial commit, the software package consists of:

- DateProcessor.vb – a module to read in dates and timestamps from log files.
- PorterStemmerAlgorithm.vb – a module containing an implementation of the Porter–Stemmer algorithm [6], written by Stemmer.
- SVMClassification.vb – a module for applying an SV model file generated by SVM-Light [7] to a feature vector.
- ProcessMachineData.aspx (see Fig. 6).
- ProcessMachineData.aspx.vb.
- TimelogAverager.aspx (see Fig. 7).
- TimelogAverager.aspx.vb.
- TrainingData.csv – data used for training the SVM.
- svm_model – data file for the SV model.
- words.dat – data file.
- Web.config – configuration file.
- README.md – readme text file.
- Example time logs – a folder containing time logs for four experiments we have carried out, characterising constitutive promoters.

8 Acknowledgment

This work was supported by two EPSRC grants: EPL011573/1 and EPJ02175X/1.

9 References

1 Andrianantoandro, E., Basu, S., Karig, D.K., *et al.*: 'Synthetic biology: new engineering rules for an emerging discipline', *Mol. Syst. Biol.*, 2006, **2**, p. 2006.0028
2 Gay, D., Guigourès, R., Boullé, M., *et al.*: 'TESS: temporal event sequence summarization'. 2015 IEEE Int. Conf. on Data Science and Advanced Analytics (DSAA), 2015, pp. 1–10
3 Monroe, M., Lan, R., Lee, H., *et al.*: 'Temporal event sequence simplification', *IEEE Trans. Vis. Comput. Graph.*, 2013, **19**, (12), pp. 2227–2236
4 'Microsoft chart controls for Microsoft.NET framework 3.5', *Microsoft download center*. Available at https://www.microsoft.com/en-gb/download/details.aspx?id=14422, accessed 03 March 2017
5 Hirst, C.D.: 'Automated BioPart characterisation for synthetic biology' (Imperial College London, 2014)
6 Porter, M.F.: 'An algorithm for suffix stripping', *Program*, 1980, **14**, (3), pp. 130–137
7 Joachims, T.: 'SVM-Light', University of Dortmund, Informatik, AI-Unit collaborative research center, 2008