App Introduction

This app detects events in ShapeSeq experimental data. Events are detected down each column in the data matrix. Detected events come in two categories: swing events and linear ramp events. Swing events are sudden changes in ShapeSeq measurement either going from low to high (upswing event) or high to low (downswing event). Linear ramps, in contrast, are long events that persist for multiple positions. The swing events are detected using an adaptation of the PID controller; swing events are detected when the value increases/decreases above three thresholds: proportional, integrative, and differential. The ramp events are detected by conducting linear regression and testing for noise, steepness, and uniformity.

The provided example data illustrates this app’s use. The provided data comes from a ShapeSeq experiment that measures RNA folding as a function of RNA length. See Example Data Descriptions below.

This app is interactive meaning any change you make on the front should immediately update on the user interface. If you make a change that breaks the code/figure, you can either reload the app (losing all progress) or revert the change.

App Startup

To start the app, make sure R and RStudio are installed. Both can be found for free online. Double click on ui.R and RStudio should pop up. Press Ctrl-Shift-Enter to run the app and a user interface should pop up with the examples already displayed. To exit or restart the app, simply exit the RStudio pop-up (RStudio does not need to be restarted).

File Inputs and Outputs

For the file input, look at the example\_data.csv file. This comma-separated file has 125 columns and 96 rows. In principal, any 2D matrix can be loaded. Note that column names are ignored and row names should not exist in the csv file

An input data file can be loaded using the browse button in Input data file. If you do not input anything, then the example\_data.csv file will be loaded instead. Similarly, you can specify your output file name, and the default name will be example\_output if you specifying nothing. Once you have created your plot (explained below), you can export results as a table .csv file or export the figures in a pdf.

Plotting Parameters

First, press Update plot at the bottom of the right-most panel. This will update the figure and should take a few seconds. There are two figures in this app: first is a heatmap showing the data with events highlighted in red/blue and the second are details of each column shown as a scatterplot. Upswing/downswings are shown in red/blue boxes and linear ramps are red/blue lines.

This app supports basic visual changes such as sizing, y-axis range, and showing column details. Figure width and figure height adjust the figure sizes in inches. y-axis range specifies the y-axis in the column details (specify two numbers, the min and max of the y-axis), Columns to display allows the user to specify what columns to show in the details, and Show all columns automatically shows all columns regardless of the box above.

In the default example (created on startup), the figure width and height are both 10 inches, the y-axis is set to a min of 0 and max of 15, and columns 1, 11, 16, and 26 are displayed below. To specify columns, use numbers only and separate with a comma (without spaces)

PID Parameters

The PID parameters are responsible for detecting swing events. A swing event is detected if all three PID parameters are satisfied.

*Pre-processing*

Before describing the parameters, some data processing is required to attenuate noise. We take the mean of a sliding window down each column and set aside these new values as “pre-event values.” The values are stored in the last position of the sliding window. Taking the mean smooths out noisy events, and the original data is still retained as “post-event” data. For all three parameters, the pre-event values will be compared to the post-event data. The size of the sliding window is specified with Window size.

*Differential (D)*

The differential (D) parameter is the absolute change from pre-event data to post-event data, shifted by one position. For example, if the pre-event value is 1.2, and the next row has a post-event value of 1.6, then the differential change is 0.3. The D-parameter is necessary to filter out low magnitude changes that are due to noise. For instance, column 16 (shown in details below) has only noise and is entirely filtered out by the D-parameter.

*Proportional (P)*

Proportional (P) is the proportional change and is the same as D except divided by the magnitude of the pre-event value. For example, if the pre-event value is 4, and the next position has a post-event value of 5, then the proportional change is 0.2. The P-parameter is necessary to filter out proportionally-low changes in high magnitude measurements. For example, column 1 has high magnitude values which appear to be mostly noise (though this is subjective and changeable with the P-parameter). If we only included the D-parameter, then every position would be considered a swing event purely because of high magnitude. By including the Proportional change, most positions are not events.

There is another wrinkle with proportional change. An upswing and a downswing event of equal magnitudes do not appear the same with proportional change. For example, going from 5 to 10 is P change of 1, but going from 10 to 5 is a P change of (negative) 0.5. Instead, the P-parameter is specified in the upswing direction and is automatically adjusted for the downswing direction. The downswing P-parameter is:

*Integral (I)*

Integral (I) change integrates the differential change across multiple positions. The number of positions that are integrated over is the same as in Window size. The I-parameter helps to remove sharp transient changes or anomalies that persist over very few positions. For example, column 26 has two events that may seem anomalous and pass both D and P parameters. A higher I-parameter filters out these events because it spreads the D and P change across multiple positions.

Noise Parameters

These noise parameters are included in the PID parameters panel and can be used for final noise filtering (applied only to swing events). From experience, even with meticulous PID parameter tuning, there will always be some events that are misrepresented due to noise.

First, some real event may have gaps that were not detected. Instead of a continuous series of detected events, there will be gaps. We can include an assumption about our events: noise can cause real events to occasionally have gaps in an otherwise continuous series of detected events. This is solved by the event gap parameter, which fills in gaps of the specified length between detected events. To disable, simply set the parameter to 0.

Second, some false events appear as detected swing events over a few positions. We can include another assumption: noise can occasional create short instances of detected events, but these are rare and would not be affected by the previous event gap parameter. We can solve this by removing any event series that have a length at or below a specified parameter. This parameter is noise length and can be disabled by setting to 0.

Linear Ramp Parameters

The linear ramp parameters controlling detection of linear ramp events. Linear regression is conducted over a specified ramp window and if the fitted line passes three conditions (p-value, beta value, and DW p-value), then the entire window is marked with a linear ramp. For linear ramps, the y-value is the measured experimental values and the x-value is always 1-to-ramp size.

*Ramp length*

Ramp length controls the window size to fit the linear ramp. This parameter should generally be long to distinguish it from swing events detected by the PID parameters. A rule of thumb is to set the ramp length to your expectations of the shortest ramp in the data. For example, column 57 has a distinctive ramp that is approximately 40 positions long. It should be noted that the longer ramp lengths suffer from over-elongating the ramps; a short ramp will be detected as longer because a large proportion will still allow it to pass the three conditions below.

*Ramp p-value*

Ramp p-value asks if the noise around the linear fit is too high. A lower p-value means there is less variance around the line and that the linear ramp is more likely to truly exist, rather than as a result of pure noise. Illustrated with a toy example:

|  |
| --- |
|  |

The left scenario is not statistically significant because the values are too spread out, but the right scenario passes.

Because many lines are being fitted, it is suggested that you choose a lower p-value by using multiple hypothesis correction, like the highly conservative Bonferroni correction. A Bonferroni correction would divide your original p-value (usually 0.05) by:

But less conservative corrections also exist. In the example data file, there are many empty data values, which cuts down on the number of linear regression and would further adjust the p-value (omitted here).

*Ramp beta coefficient*

The Beta coefficient sets the minimum absolute value of the slope. Naturally, linear ramps with a zero slope are not events and the minimum beta-value should be set according to expectations.

*Ramp Durbin-Watson p-value*

The Durbin-Watson p-value accounts for the shape of the data values around the line. Not only do we expect a Gaussian distribution around the fitted lines, but we also expect this distribution to persist down the length of the line. This is better described visually:

|  |
| --- |
|  |

In this toy example, both scenarios are linear ramps, but the left should not qualify because the residuals (distance between data and line) are not uniformly distributed down the line. The Durbin-Watson statistic tests for scenarios on the left (specifics: it looks for autocorrelation in residuals). When applying this test,higher confidence of uniformly distributed residualsresult in *higher* p-values. In this case, the higher we choose the p-value, the stricter our Durbin-Watson test. It is suggested to leave this value at the standard 0.05 or 0.01. Also, I am not sure if multiple-hypothesis testing applies so the p-value probably does not need corrections.

Example Data Descriptions

The example data file is “example\_data.csv”. The data is from the published manuscript:

Kyle E Watters, Eric J. Strobel, Angela M. Yu, John T. Lis, & Julius B. Lucks. Cotranscriptional folding of a riboswitch at nucleotide resolution. *Nature Structural and Molecular Biology*. (2016)

The ShapeSeq experiment measures RNA folding by exposing different lengths of RNA to ShapeSeq. ShapeSeq preferentially binds to open regions of RNA and subsequent assays can measure the location and presence of ShapeSeq. Computational corrections and other analysis were done to generate the provided example data.

The data is a 2D matrix where columns are RNA positions and rows describe the different RNA lengths. As we go down the rows, the RNA length increases and folding behavior can change. The first row has the minimum RNA length of 30 and each row adds the next position, up to the max of 125. High values imply “open” RNA regions that are not folded, and low values suggest folded or otherwise inaccessible RNA regions. However, low values are not necessarily indicative of inaccessibility as other explanations exist.

Author Information

This app was created by Albert Xue of the Bagheri Lab at Northwestern University.

Current version updated on 08/08/17

You can reach him at [xuebert@u.northwestern.edu](mailto:xuebert@u.northwestern.edu)