Internal algorithms of **qFeature**

Landon Sego

4 September 2015

The following is a mathematical description of the algorithms in the **fitQ** and **discFeatures** functions in the **qFeature** package:

Extracting features from continuous time series using **fitQ**

A quadratic regression model is fit to a moving window of the time series. To simplify the solution of the least square estimators, and thus avoid complicated (and expensive) expressions and/or matrix inversion, the linear and quadratic predictors are centered so that they are both orthogonal to the intercept, and if possible, orthogonal to one another. The usual quadratic regression model can be written as:

$$y_i = a + bx_i + cx_i^2 + e_i \tag{1.1}$$

We can create an equivalent model by centering the linear and quadratic predictors as follows:

$$y_i = a' + b'x_i^* + c'x_i^{**} + e_i' \tag{1.2}$$

where

$$x_i^* = x_i - \frac{1}{n}\sum_{i=1}^{n} x_i \ , \quad x_i^{**} = \left(x_i^*\right)^2 - \frac{1}{n}\sum_{i=1}^{n}\left(x_i^*\right)^2 \ , \tag{1.3}$$

and n is the number of data points in the window. Note that $\sum_{i=1}^{n} x_i^* = 0$ and $\sum_{i=1}^{n} x_i^{**} = 0$, which implies that x_i^* and x_i^{**} are orthogonal to the intercept. Furthermore, if the original x_i are evenly spaced, then $\sum_{i=1}^{n} x_i^* x_i^{**} = 0$. However, in the event there are missing data points in the window, the original x_i are not evenly spaced. Using (1.3) and some algebra, we can rewrite (1.2) as:

$$y_i = \left(a' - b'\bar{x} + c'\left(\bar{x}^2 - \bar{\bar{x}}\right)\right) + \left(b' - 2c'\bar{x}\right)x_i + c'x_i^2 + e_i' \tag{1.4}$$

where

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad \text{and} \quad \bar{\bar{x}} = \frac{1}{n}\sum_{i=1}^{n}\left(x_i^*\right)^2 \tag{1.5}$$

Using (1.4), we can equate the original parameters in (1.1) to those in (1.2):

$$
\begin{aligned}
a &= a' - b'\bar{x} + c'\left(\bar{x}^2 - \bar{\bar{x}}\right) \\
b &= b' - 2c'\bar{x} \\
c &= c' \\
e_i &= e_i' \;, \quad \forall i
\end{aligned}
\tag{1.6}
$$

The advantage of the parameterization given in (1.2) is that it simplifies the computation of the least squares estimates of the model parameters. To begin with, suppose the x_i are evenly spaced so that x_i^* and x_i^{**} are orthogonal. It can be readily shown that the least squares estimates are then given by:

$$
\begin{aligned}
\hat{a}' &= \frac{1}{n}\sum_{i=1}^{n} y_i \\
\hat{b}' &= \sum_{i=1}^{n} x_i^* y_i \Big/ \sum_{i=1}^{n}\left(x_i^*\right)^2 \\
\hat{c}' &= \sum_{i=1}^{n} x_i^{**} y_i \Big/ \sum_{i=1}^{n}\left(x_i^{**}\right)^2
\end{aligned}
\tag{1.7}
$$

And the residual is calculated in the usual fashion:

$$
e_i' = y_i - \hat{a}' - \hat{b}' x_i^* - \hat{c}' x_i^{**} \tag{1.8}
$$

If x_i are not evenly spaced so that x_i^* and x_i^{**} are not orthogonal (which would occur if the window contained missing data), it can be shown that the least squares estimates are given by:

$$
\begin{aligned}
\hat{a}' &= \frac{1}{n}\sum_{i=1}^{n} y_i \\
\hat{b}' &= \frac{1}{D}\left[\left(\sum_{i=1}^{n}\left(x_i^{**}\right)^2\right)\left(\sum_{i=1}^{n} x_i^* y_i\right) - \left(\sum_{i=1}^{n} x_i^* x_i^{**}\right)\left(\sum_{i=1}^{n} x_i^{**} y_i\right)\right] \\
\hat{c}' &= \frac{1}{D}\left[\left(\sum_{i=1}^{n}\left(x_i^{*}\right)^2\right)\left(\sum_{i=1}^{n} x_i^{**} y_i\right) - \left(\sum_{i=1}^{n} x_i^* x_i^{**}\right)\left(\sum_{i=1}^{n} x_i^{*} y_i\right)\right]
\end{aligned}
\tag{1.9}
$$

where

$$
D = \left[\sum_{i=1}^{n}\left(x_i^*\right)^2\right]\left[\sum_{i=1}^{n}\left(x_i^{**}\right)^2\right] - \left[\sum_{i=1}^{n} x_i^* x_i^{**}\right]^2 \tag{1.10}
$$

Note that (1.9) reduces to (1.7) when $\sum_{i=1}^{n} x_i^* x_i^{**} = 0$, i.e., when x_i^* and x_i^{**} are orthogonal. The **fitQ** function fits the quadratic model shown in (1.1) by calculating the parameter estimates for each window using (1.7) or (1.9) and then "backtransforming" to the original parameterization using (1.6).

If the **linear.only** argument of **fitQ** is set to **TRUE**, a similar approach to the one described above is used, except the quadratic term in (1.1) is omited, and the calculations are simpler.

Extracting features from a discrete time series using **discFeatures**

We will refer to the distinct values of a discrete time series as states, where the value of the state may change (or remain the same) for each time point in the series. The **discFeatures** function produces a summary of the percentage of time spent in each state, along with the number of transitions that occurred between the various states. Calculating the percentage of time that is spent within each state is a relatively simple matter and will not be discussed here.

The strategy of the algorithm is to create a mapping of the state labels to a set of integers such that first difference of those integers will uniquely identify all possible transitions between the states at a given point in time. The approach is easiest to explain using a simple example:

Suppose we have a discrete vector that has four states labeled 0, 1, 2, and 3. We found that the mapping $m(x) = 2^x$ will produce unique differences among each of the states. That is, $m(x) - m(y)$ will be unique for every combination of $x \neq y$, where *x* and *y* correspond to state labels 0, 1, 2, …. The approach works for 2 to 55 unique states. Beyond 55, some combinations of $m(x) - m(y)$ begin to repeat due to what appears to be machine error. The transition matrix below shows the values of $m(x) - m(y)$ for each of the 16 possible ways to move from state *x* (on the rows) to state *y* (on the columns).

	0	**1**	**2**	**3**
0	0	-1	-3	-7
1	1	0	-2	-6
2	3	2	0	-4
3	7	6	4	0

The diagonal elements correspond to the situation where the state does not change, which is why their matrix elements are zero. For purposes of this discussion, these diagonal elements are not of concern to us since they are easily summarized by the percentage of time spent in that particular state.

The off-diagonal elements of the matrix are unique. For example, if the discrete parameter transitions from state 3 to state 1, then that transition is uniquely identified in the matrix by $m(3) - m(1) = 6$. And the transition from state 1 to state 2 is represented by $m(1) - m(2) = -2$.

To count the number transitions from one state to another for a discrete time series *Y*, the first (or successive) differences of *m*(*Y*) are tabulated. The number of "6's" that occur will indicate the number of times the variable transitions from state 3 to state 1. Likewise, the number of "-2's"

that occur in a given phase will indicate the number of times the variable transitions from state 1 to state 2, etc.

This approach helps to minimize the number of passes that need to be made through the vector of the discrete variable in order to count all the possible transitions.