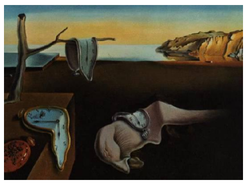

Journal of Statistical Software
June 2014, Volume 58, Issue 3. http://www.jstatsoft.org/

changepoint: An R Package for Changepoint Analysis

Rebecca Killick
Lancaster University

Idris A. Eckley
Lancaster University

Abstract

One of the key challenges in changepoint analysis is the ability to detect multiple changes within a given time series or sequence. The **changepoint** package has been developed to provide users with a choice of multiple changepoint search methods to use in conjunction with a given changepoint method and in particular provides an implementation of the recently proposed PELT algorithm. This article describes the search methods which are implemented in the package as well as some of the available test statistics whilst highlighting their application with simulated and practical examples. Particular emphasis is placed on the PELT algorithm and how results differ from the binary segmentation approach.

Keywords: segmentation, break points, search methods, bioinformatics, energy time series, R.

1. Introduction

There is a growing need to be able to identify the location of multiple change points within time series. However, as datasets increase in length the number of possible solutions to the multiple changepoint problem increases combinatorially. Over the years several multiple changepoint search algorithms have been proposed to overcome this challenge, most notably the binary segmentation algorithm (Scott and Knott 1974; Sen and Srivastava 1975); the segment neighborhood algorithm (Auger and Lawrence 1989; Bai and Perron 1998) and more recently the PELT algorithm (Killick, Fearnhead, and Eckley 2012a). This paper describes the **changepoint** package (Killick, Eckley, and Haynes 2014), available for R (R Core Team 2014) from the Comprehensive R Archive Network (CRAN) at `http://CRAN.R-project.org/package=changepoint`. Package **changepoint** makes each of these algorithms available, thus enabling users to select which method they would like to use for their analysis.

We are by no means the first to develop a changepoint package for the R environment. At the time of writing several such packages exist, including those which provide a single test statistic e.g., **sde** (Iacus 2009), **bcp** (Erdman and Emerson 2007) and/or are designed for a specific (typically genomic) application e.g., **cumSeg** (Muggeo 2012), **DNAcopy** (Seshan and Olshen 2008). More comprehensive R packages are also available such as **strucchange** (Zeileis, Leisch, Hornik, and Kleiber 2002) for changes in regression and **cpm** (Ross 2013) for online changepoint detection. However, all of the aforementioned packages implement a single search method for detecting multiple changepoints. In contrast, the **changepoint** package uniquely provides a choice of search algorithms for multiple changepoint detection in addition to a variety of test statistics. In particular the package implements the search algorithms for a selection of popular changepoint and penalty types. Specifically methods are implemented for the change in mean and/or variance settings with a similar argument structure where each function outputs an object of class '`cpt`'. Such an approach is deliberate to breed familiarity and ease of use. Whilst the package is driven from these core functions, part of our philosophy is to make it easier for others to use and adapt code snippets as appropriate. To this end we have deliberately coded each part of a method in an individual function which is also exported. Whilst several test statistics are included in the **changepoint** package there are currently some notable gaps which are covered by other software. These include changes in regression (see **strucchange**, Zeileis *et al.* 2002) and changes in autocorrelation (see **AutoPARM** available from Davis, Lee, and Rodriguez-Yam 2006). In addition there is currently no general software available whereby the user can supply their own cost function and this would be an interesting avenue to pursue. A list of general changepoint software, and indeed recent preprints in the area, are available from The Changepoint Repository (Killick, Nam, Aston, and Eckley 2012b, `http://changepoint.info`).

The remainder of the paper is structured as follows. A brief background to changepoint analysis is given in Section 2 before Section 3 describes the '`cpt`' class and its methods. Following this the three main functions; `cpt.mean`, `cpt.var` and `cpt.meanvar` are described and explored using simulated and practical examples. In these sections particular emphasis is placed on how to identify multiple changepoints and the difference between exact and approximate methods. The paper is summarized in Section 7, where we provide a discussion.

2. Changepoint detection

This section begins by introducing the reader to changepoints through the single changepoint problem before considering the extension to multiple changepoints. In its simplest form, changepoint detection is the name given to the problem of estimating the point at which the statistical properties of a sequence of observations change. Detecting such changes is important in many different application areas. Recent examples include climatology (Reeves, Chen, Wang, Lund, and Lu 2007), bioinformatic applications (Erdman and Emerson 2008), finance (Zeileis, Shah, and Patnaik 2010), oceanography (Killick, Eckley, Jonathan, and Ewans 2010) and medical imaging (Nam, Aston, and Johansen 2012).

More formally, let us assume we have an ordered sequence of data, $y_{1:n} = (y_1, \ldots, y_n)$. A changepoint is said to occur within this set when there exists a time, $\tau \in \{1, \ldots, n-1\}$, such that the statistical properties of $\{y_1, \ldots, y_\tau\}$ and $\{y_{\tau+1}, \ldots, y_n\}$ are different in some way. Extending this idea of a single changepoint to multiple changes, we will have a number of changepoints, m, together with their positions, $\tau_{1:m} = (\tau_1, \ldots, \tau_m)$. Each changepoint position is an integer between 1 and $n-1$ inclusive. We define $\tau_0 = 0$ and $\tau_{m+1} = n$, and assume that the changepoints are ordered so that $\tau_i < \tau_j$ if, and only if, $i < j$. Consequently the m changepoints will split the data into $m+1$ segments, with the ith segment containing data $y_{(\tau_{i-1}+1):\tau_i}$. Each segment will be summarized by a set of parameters. The parameters associated with the ith segment will be denoted $\{\theta_i, \phi_i\}$, where ϕ_i is a (possibly null) set of nuisance parameters and θ_i is the set of parameters that we believe may contain changes. Typically we want to test how many segments are needed to represent the data, i.e., how many changepoints are present and estimate the values of the parameters associated with each segment.

2.1. Single changepoint detection

Let us briefly recap the likelihood based framework for changepoint detection. Before considering the more general problem of identifying $\tau_{1:m}$ changepoint positions, we first consider the identification of a single changepoint. The detection of a single changepoint can be posed as a hypothesis test. The null hypothesis, H_0, corresponds to no changepoint ($m = 0$) and the alternative hypothesis, H_1, is a single changepoint ($m = 1$).

We now introduce the general likelihood ratio based approach to test this hypothesis. The potential for using a likelihood based approach to detect changepoints was first proposed by Hinkley (1970) who derives the asymptotic distribution of the likelihood ratio test statistic for a change in the mean within normally distributed observations. The likelihood based approach was extended to changes in variance within normally distributed observations by Gupta and Tang (1987). The interested reader is referred to Silva and Teixeira (2008) and Eckley, Fearnhead, and Killick (2011) for a more comprehensive review.

A test statistic can be constructed which we will use to decide whether a change has occurred. The likelihood ratio method requires the calculation of the maximum log-likelihood under both null and alternative hypotheses. For the null hypothesis the maximum log-likelihood is $\log p(y_{1:n}|\hat{\theta})$, where $p(\cdot)$ is the probability density function associated with the distribution of the data and $\hat{\theta}$ is the maximum likelihood estimate of the parameters.

Under the alternative hypothesis, consider a model with a changepoint at τ_1, with $\tau_1 \in \{1, 2, \ldots, n-1\}$. Then the maximum log likelihood for a given τ_1 is

$$ML(\tau_1) = \log p(y_{1:\tau_1}|\hat{\theta}_1) + \log p(y_{(\tau_1+1):n}|\hat{\theta}_2). \tag{1}$$

Given the discrete nature of the changepoint location, the maximum log-likelihood value under the alternative is simply $\max_{\tau_1} ML(\tau_1)$, where the maximum is taken over all possible changepoint locations. The test statistic is thus

$$\lambda = 2\left[\max_{\tau_1} ML(\tau_1) - \log p(y_{1:n}|\hat{\theta})\right].$$

The test involves choosing a threshold, c, such that we reject the null hypothesis if $\lambda > c$. If we reject the null hypothesis, i.e., detect a changepoint, then we estimate its position as $\hat{\tau}_1$ the value of τ_1 that maximizes $ML(\tau_1)$. The appropriate value for this parameter c is still an open research question with several authors devising p values and other information criteria under different types of changes. We refer the interested reader to Guyon and Yao (1999); Chen and Gupta (2000); Lavielle (2005); Birge and Massart (2007) for interesting discussions and suggestions for c.

It is clear that the likelihood test statistic can be extended to multiple changes simply by summing the likelihood for each of the m segments. The problem becomes one of identifying the maximum of $ML(\tau_{1:m})$ over all possible combinations of $\tau_{1:m}$. The following section explores existing search methods that address this problem.

2.2. Multiple changepoint detection

With increased collection of time series and signal streams there is a growing need to be able to efficiently and accurately estimate the location of multiple changepoints. This section briefly introduces the main search methods available for identifying multiple changepoints within the **changepoint** package. Arguably the most common approach to identify multiple changepoints in the literature is to minimize

$$\sum_{i=1}^{m+1} \left[\mathcal{C}(y_{(\tau_{i-1}+1):\tau_i})\right] + \beta f(m) \tag{2}$$

where $\mathcal{C}$ is a cost function for a segment e.g., negative log-likelihood and $\beta f(m)$ is a penalty to guard against over fitting (a multiple changepoint version of the threshold c). This is the approach which we adopt in this paper and the accompanying package. A brute force approach to solve this minimization considers 2^{n-1} solutions reducing to $\binom{n-1}{m}$ if m is known. The **changepoint** package implements three multiple changepoint algorithms that minimize (2); binary segmentation (Edwards and Cavalli-Sforza 1965), segment neighborhoods (Auger and Lawrence 1989) and the recently proposed pruned exact linear time (PELT) (Killick *et al.* 2012a). Each of these algorithms is briefly described in the following paragraphs, for more information see the corresponding references.

At the time of writing binary segmentation is arguably the most widely used multiple changepoint search method and originates from the work of Edwards and Cavalli-Sforza (1965), Scott and Knott (1974) and Sen and Srivastava (1975). Briefly, binary segmentation first applies a single changepoint test statistic to the entire data, if a changepoint is identified the data is split into two at the changepoint location. The single changepoint procedure is repeated on the two new data sets, before and after the change. If changepoints are identified in either of the new data sets, they are split further. This process continues until no changepoints are found in any parts of the data. This procedure is an approximate minimization of (2) with $f(m) = m$ as any changepoint locations are conditional on changepoints identified previously. Binary segmentation is thus an *approximate algorithm* but is computationally fast as it only considers a subset of the 2^{n-1} possible solutions. The computational complexity of the algorithm is $\mathcal{O}(n \log n)$ but this speed can come at the expense of accuracy of the resulting changepoints (see Killick *et al.* 2012a, for details).

The segment neighborhood algorithm was proposed by Auger and Lawrence (1989) and further explored in Bai and Perron (1998). The algorithm minimizes the expression given by Equation 2 exactly using a dynamic programming technique to obtain the optimal segmentation for $m+1$ changepoints reusing the information that was calculated for m changepoints. This reduces the computational complexity from $\mathcal{O}(2^n)$ for a naive search to $\mathcal{O}(Qn^2)$ where Q is the maximum number of changepoints to identify. Whilst this algorithm is exact, the computational complexity is considerably higher than that of binary segmentation.

The binary segmentation and segment neighborhood algorithms would appear to indicate a trade-off between speed and accuracy however this need not be the case. The PELT algorithm proposed by Killick *et al.* (2012a) is similar to that of the segment neighborhood algorithm in that it provides an exact segmentation. However, due to the construction of the PELT algorithm, it can be shown to be more computationally efficient, due to its use of dynamic programming and pruning which can result in an $\mathcal{O}(n)$ search algorithm subject to certain assumptions being satisfied, the majority of which are not particularly onerous. Indeed the main assumption that controls the computational time is that the number of changepoints increases linearly as the data set grows, i.e., changepoints are spread throughout the data rather than confined to one portion.

All three search algorithms are available within the **changepoint** package. The following sections introduce the structure of the package, its S4 class – '`cpt`' and the core functions that enable quick and efficient analysis of changepoint problems.

3. Introduction to the package and the '`cpt`' class

The **changepoint** package introduces a new object class called '`cpt`' to store changepoint analysis objects. This section provides an introduction to the structure and methods associated with the '`cpt`' class, together with examples of its specific use.

Each of the core functions outputs an object of the '`cpt`' S4 class. The class has been constructed such that the '`cpt`' object contains the main features required for a changepoint analysis and future summaries. Each of these is stored within a slot entry in the '`cpt`' class. The slots within the class are,

- `data.set` – a time series ('`ts`') object containing the numeric values of the data;
- `cpttype` – characters describing the type of changepoint sought e.g., mean, variance;
- `method` – characters denoting the single or multiple changepoint search method applied;
- `test.stat` – characters denoting the test statistic, i.e., assumed distribution / distribution-free method;
- `pen.type` – characters denoting the penalty type, e.g., AIC, BIC, manual;
- `pen.value` – the numeric value of the penalty used in the analysis;
- `cpts` – a numeric vector giving the estimated changepoint locations always ending in n, the length of the time series in the `data.set` slot;
- `ncpts.max` – the numeric maximum number of changepoints searched for, e.g., 1, 5, `Inf` and denoted `Q` in Section 2;
- `param.est` – a list of parameters where each element in the list is a vector of the estimated numeric parameter values for each segment, denoted θ_i in Section 2;
- `date` – the system time / date when the analysis was performed.

Slots of an S4 object are typically accessed using the @ symbol (in contrast to the $ for S3 objects). Whilst this is still possible in the **changepoint** package, we have created accessor and replacement functions to control the access and replacement of slots. The accessor functions are simply the slot names. For example `data.set(x)` displays the vector of data contained within the '`cpt`' object `x`. The class slots are automatically populated with the correct information obtained from the completed analysis. Feedback from trials with the package users indicate that the accessor and replacement functions aid ease-of-use for those unfamiliar with S4 classes. Further demonstration of how the accessor and replacement functions work in practice are given in the examples within each section.

In addition to accessor and replacement functions, the **changepoint** package also contains a couple of extra functions that a user may find useful. The first of these is the `ncpts` function which, given a '`cpt`' object from a changepoint analysis, returns the number of identified changepoints. This can be particularly useful if the number of changepoints is expected to be large and/or users wish to quickly check whether the returned number of changepoints is equal to the maximum searched for when using the binary segmentation or segment neighborhood search algorithms. Similarly the second additional function, `seg.len`, returns the size of the segments, i.e., how many observations there are between consecutive changepoints. This may be useful when performing a changepoint analysis as short segments can be used as an indicator that the penalty function may be set too low.

All the functions described above are related to the '`cpt`' class within the **changepoint** package. The following section reviews the methods that act on the '`cpt`' class.

3.1. Methods within the '`cpt`' class

The methods associated with the '`cpt`' class are `summary`, `print`, `plot`, `coef` and `logLik`. The `summary` and `print` methods display standard information about the '`cpt`' object. The `summary` function displays a synopsis of the results from the analysis including number of changepoints and, where this number is small, the location of those changepoints. In contrast, the `print` function prints details pertaining to the S4 class including slot names and when the S4 object was created.

Having performed a changepoint analysis, it is often helpful to be able to plot the changepoints on the original data to visually inspect whether the estimated changepoints are reasonable. To this end we include a `plot` method for the '`cpt`' class. The method adapts to the assumed type of changepoint, providing a different output dependent on the type of change. For example, a change in variance is denoted by a vertical line at the changepoint location whereas a change in mean is indicated by horizontal lines depicting the mean value in different segments.

Similarly once a changepoint analysis has been conducted one may wish to retrieve the parameter values for each segment or the log likelihood for the fitted data. These can be obtained using the standard `coef` and `logLik` generics; examples are given in the code detailed below.

The following sections explore the use of the core functions within the **changepoint** package. We begin in Section 4 by demonstrating the key steps to a changepoint analysis via the `cpt.mean` function. Sections 5 and 6 utilize the steps in the change in mean analysis to explore changes in variance and both mean and variance respectively.

4. Changes in mean: The `cpt.mean` function

Early work on changepoint problems focused on identifying changes in mean and includes the work of Page (1954) and Hinkley (1970) who created the likelihood ratio and cumulative sum (CUSUM) test statistics respectively.

Within the **changepoint** package all change in mean methods are accessed using the `cpt.mean` function. The function is structured as follows:

```
cpt.mean(data, penalty = "SIC", pen.value = 0, method = "AMOC", Q = 5,
  test.stat = "Normal", class = TRUE, param.estimates = TRUE)
```

The arguments within this function are:

- `data` – A vector or 'codets' object containing the data within which to find a change in mean. If multiple datasets require to be analyzed, then this can be a matrix where each row is considered a separate dataset.
- `penalty` – Choice of `"None"`, `"SIC"`, `"BIC"`, `"AIC"`, `"Hannan-Quinn"`, `"Asymptotic"` and `"Manual"` penalties. If `"Manual"` is specified, the manual penalty is contained in `pen.value`. If `"Asymptotic"` is specified, the theoretical type I error is contained in `pen.value`. The predefined penalties listed do *not* count the changepoint as a parameter, postfix a 1 e.g., `"SIC1"` to count the changepoint as a parameter.
- `pen.value` – The theoretical type I error e.g., 0.05 when using the `"Asymptotic"` penalty. Alternatively when using the `"Manual"` penalty it is a numeric value or text which when evaluated results in a penalty value.
- `method` – Single or multiple changepoint method. Choice of `"AMOC"` (at most one change), `"PELT"`, `"SegNeigh"` or `"BinSeg"`. Default is `"AMOC"`. See Section 2 for further details of methods.
- `Q` – When using the `"BinSeg"` method this is the maximum number of changepoints to search for. When using the `"SegNeigh"` method this is the maximum number of segments (number of changepoints + 1) to search for. This is not required for the `"PELT"` method as this automatically selects the number of segments.
- `test.stat` – The test statistic, i.e., assumed distribution or distribution-free method for `data`. Choice of `"Normal"` or `"CUSUM"`. The test statistics behind the distributional options are contained within Hinkley (1970) for the `"Normal"` option and Page (1954) for the `"CUSUM"` option.
- `class` – Logical. If `TRUE` then an object of class '`cpt`' is returned.
- `param.estimates` – Logical. If `TRUE` and `class = TRUE` then parameter estimates are returned. If `FALSE` or `class = FALSE` no parameter estimates are returned.

Briefly the search options consist of exact methods: PELT ($\mathcal{O}(n)$ if assumptions are satisfied), segment neighborhoods ($\mathcal{O}(Qn^2)$); and approximate methods: binary segmentation ($\mathcal{O}(n \log n)$). Further details of the search options in the `method` argument are given in Section 2.

Several standard penalty functions used within changepoint analysis have been included in this function. These are: SIC (Schwarz information criterion), BIC (Bayesian information criterion), AIC (Akaike information criterion) and Hannan-Quinn. The authors will seek to include further penalty functions, such as minimum description length (MDL) (Davis *et al.* 2006), in future versions of the package. The user can also enter a manual penalty value by numeric value or formula. An example of using a manual penalty value with a formula is given in Section 4.1. In addition to the standard R functions, the following variables are available for the user to utilize:

- `tau` – the proposed changepoint location (only available when using `"AMOC"`);
- `null` – the likelihood under the null model of no changepoint (only available when using `"AMOC"`);
- `alt` – the likelihood under the alternative model of a single changepoint (only available when using `"AMOC"`);
- `diffparam` – the difference in the number of parameters between the no changepoint and single changepoint model e.g., for a Normal distribution, 1 for a change in mean or variance and 2 for a change in both mean and variance;
- `n` – the length of the data.

Thus if one wanted to use a penalty based on the ratio of the lengths of data before and after the change, then one may use `penalty = "Manual"`, `pen.value = "tau / (n - tau)"`. Note this is only possible using `"AMOC"`.

The remainder of this section gives a worked example exploring how to identify a change in mean.

4.1. Example: Changes in mean

We now describe the general structure of a changepoint analysis using the **changepoint** package. We begin by demonstrating the various possible stages within a change in mean analysis. To this end we simulate a dataset (`m.data`) of length 400 with multiple changepoints at 100, 200, 300. The sequence has four segments and the means for each segment are 0, 1, 0, 0.2.

```
R> library("changepoint")
R> set.seed(10)
R> m.data <- c(rnorm(100, 0, 1), rnorm(100, 1, 1), rnorm(100, 0, 1),
+    rnorm(100, 0.2, 1))
R> ts.plot(m.data, xlab = "Index")
```

Imagine that we have been presented with this dataset and are asked to perform a changepoint analysis. The first question we aim to answer is “Is there a change within the data?”. Our first choice in answering this question is whether we wish to consider a single change or whether multiple changes are plausible. From a visual inspection of the data in Figure 1(a), we suspect multiple changes in mean may exist.

The challenge in multiple changepoint detection is identifying the optimal number and location of changepoints as the number of solutions increases rapidly with the size of the data. In this example where $n = 400$, we have 399 possible solutions for a single changepoint, for two changes there are 79401 possible solutions and this is not taking into account that we do not know how many changes there are! As such it is clearly desirable to use an efficient method for searching the large solution space.

Any of the three search methods could be used to detect these changes. For this example we will compare the PELT and binary segmentation search methods as this provides a comparison between exact and alternative algorithms (see Section 2). For now we will assume that the dataset is independent and Normally distributed and consider an alternative towards the end of this section.

```
R> m.pelt <- cpt.mean(m.data, method = "PELT")
R> plot(m.pelt, type = "l", cpt.col = "blue", xlab = "Index",
+    cpt.width = 4)
R> cpts(m.pelt)

[1]  97 192 273 353 362 366

R> m.binseg <- cpt.mean(m.data, method = "BinSeg")
R> plot(m.binseg, type = "l", xlab = "Index", cpt.width = 4)
R> cpts(m.binseg)

[1]  79  99 192 273
```

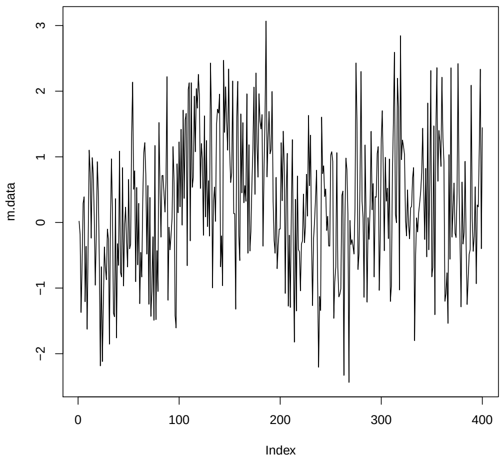
In this case, where we use the default SIC penalty, the `cpts` function returned 6 changepoints (97, 192, 273, 353, 362, 366) for PELT and 4 changepoints (79, 99, 192, 273) for binary segmentation. By construction we know that there are three changepoints within the dataset. We can either believe that there are six/four changes or consider that the method is too sensitive and try to compensate by increasing the penalty. The choice of appropriate penalty is still an open question and typically depends on many factors including the size of the changes and the length of segments, both of which are unknown prior to analysis (see Guyon and Yao 1999; Lavielle 2005; Birge and Massart 2007). As new approaches to penalty choice become available we will seek to include them within the **changepoint** package. In current practice, the choice of penalty is often assessed by plotting the data and changepoints to see if they seem reasonable.

Figure 1(b) shows the `m.pelt` changepoints. Note that there are two changes towards the end of the dataset which have very small segments. These are plausibly artifacts of the data rather than true changes in the underlying process. In an effort to remove these seemingly spurious changepoints we can increase the penalty to `1.5 * log(n)` rather than `log(n)` (SIC). This change is achieved by changing the penalty type to `"Manual"` and setting the value argument to `"1.5 * log(n)"`. Figure 1(d) shows the result which seem more plausible.

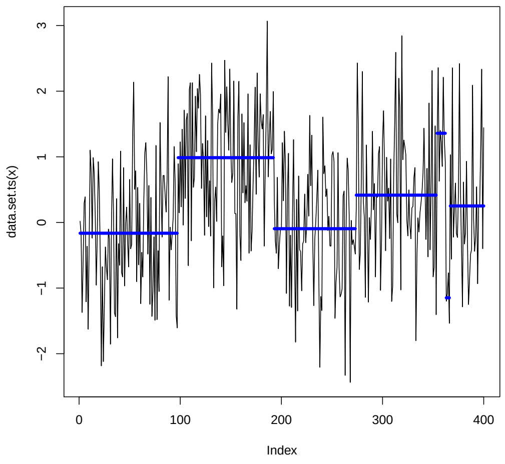
```
R> m.pm <- cpt.mean(m.data, penalty = "Manual", pen.value = "1.5 * log(n)",
+    method = "PELT")
R> plot(m.pm, type = "l", cpt.col = "blue", xlab = "Index", cpt.width = 4)
R> cpts(m.pm)

[1]  97 192 273
```

On the other hand, if we only consider the changepoints identified by the binary segmentation algorithm in Figure 1(c) then we may plausibly believe that there are four changes within the data as the spurious segment is much larger. However, for comparison we also perform the analysis with the increased penalty and find that the changepoints identified remain the same.

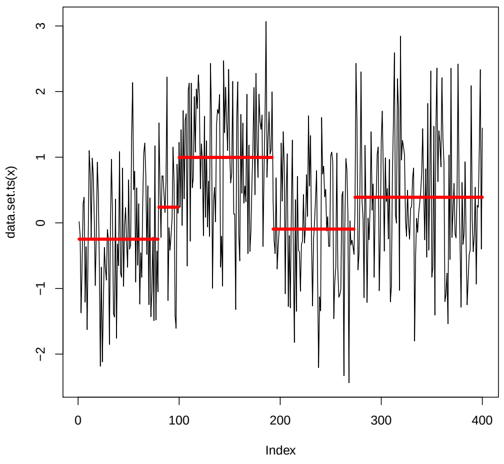
```
R> m.bsm <- cpt.mean(m.data, "Manual", pen.value = "1.5 * log(n)",
+    method = "BinSeg")
R> cpts(m.bsm)
```



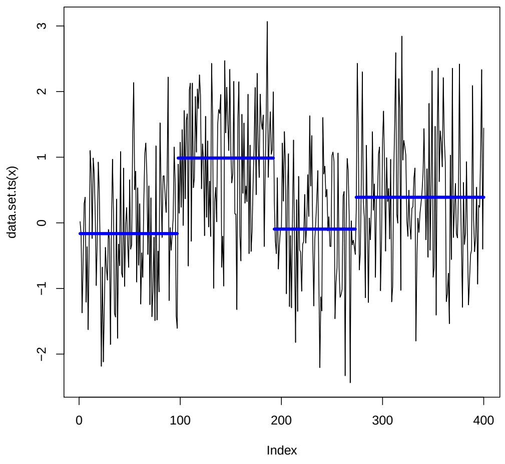
(a) `m.data`.



(b) PELT changepoints with default penalty.



(c) Binary segmentation changepoints with default penalty.



(d) PELT changepoints with manual penalty.

Figure 1: Plot of the simulated dataset `m.data` along with horizontal lines for the underlying (fitted) mean.

```
[1]  79  99 192 273
```

Recall from Section 2 that both the segment neighborhood and PELT algorithms are exact. Thus, for a linear penalty, the only difference between them is their computational time. A user can use the below commands on their own computer to identify their personal speedup for this example.

```
R> system.time(cpt.mean(m.data, method = "SegNeigh"))
R> system.time(cpt.mean(m.data, method = "PELT"))
```

Using modern computers for this example PELT will return a time needed of 0.001 or 0.002 seconds compared to segment neighborhoods where the authors have seen a range from 0.4 to 1.1 seconds for the time needed.

As a final note on this example, if the Normal assumption made at the start of the analysis is questionable then the CUSUM method, which has no distributional assumptions, can be used by adding the argument `test.stat = "CUSUM"`.

Thus far we have only considered a simulated example. In the next section we apply the `cpt.mean` function to some Glioblastoma data previously analyzed by Lai, Johnson, Kucherlapati, and Park (2005).

4.2. Case study: Glioblastoma

Lai *et al.* (2005) compare different methods for segmenting array comparative genomic hybridization (aCGH) data from Glioblastoma multiforme (GBM), a type of brain tumor. These arrays were developed to identify DNA copy number alteration corresponding to chromosomal aberrations. High-throughput aCGH data are intensity ratios of diseased vs. control samples indexed by the location on the genome. Values greater than 1 indicate diseased samples have additional chromosomes and values less than 1 indicate fewer chromosomes. Detection of these aberrations can aid future screening and treatments of diseases.

The example we consider is from Figure 4 in Lai *et al.* (2005), the data is replicated in the **changepoint** package for ease. Following Lai *et al.* (2005) we fit a Normal distribution with a piecewise constant mean using a likelihood criterion. Figure 2 demonstrates that PELT (with default penalty) gives the same segmentation as the CGHseg method from Lai *et al.* (2005).

```
R> data("Lai2005fig4", package = "changepoint")
R> Lai.default <- cpt.mean(Lai2005fig4[, 5], method = "PELT")
R> plot(Lai.default, pch = 20, col = "grey", cpt.col = "black", type = "p",
+    xlab = "Index")
R> cpts(Lai.default)

[1]  81  85  89  96 123 133

R> coef(Lai.default)

$mean
[1] 0.2468910 4.6699210 0.4495538 4.5902489 0.2079891 4.2913844 0.2291286
```

5. Changes in variance: The `cpt.var` function

Whilst considerable research effort has been given to the change in mean problem, Chen and Gupta (1997) observe that the detection of changes in variance has received comparatively little attention. Much of the work in this area builds on the foundational work of Hinkley (1970) in the change in mean setting. See for example Hsu (1979), Horvath (1993) and Chen and Gupta (1997) who extend Hinkley's ideas to the change in variance setting. Existing methods within the change in variance literature find it hard to detect subtle changes in variability, see Killick *et al.* (2010).

Within the **changepoint** package all change in variance methods are accessed using the `cpt.var` function. The function is structured as follows:

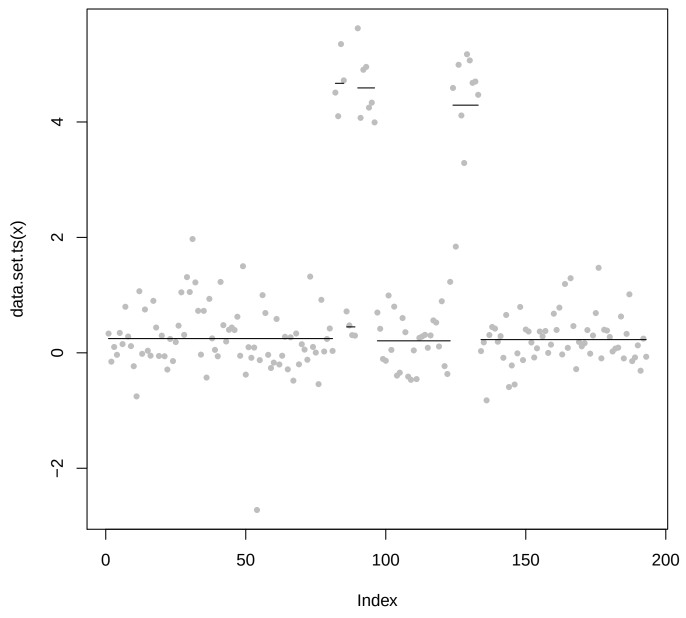


Figure 2: Plot of the GBM data along with horizontal lines for the underlying mean.

```
cpt.var(data, penalty, pen.value, know.mean = FALSE, mu = NA, method, Q,
  test.stat = "Normal", class, param.estimates)
```

The `data`, `penalty`, `pen.value`, `method`, `Q`, `class` and `param.estimates` arguments are the same as for the `cpt.mean` function (see Section 4). The three remaining arguments are interpreted as follows.

- `know.mean` – This logical argument is only required for `test.stat = "Normal"`. If `TRUE` then the mean is assumed known and `mu` is taken as its value. If `FALSE` and `mu = NA` (default value) then the mean is estimated via maximum likelihood. If `FALSE` and the value of `mu` is supplied, `mu` is not estimated but is counted as an estimated parameter for decisions.

- `mu` – Only required for `test.stat = "Normal"`. Numerical value of the true mean of the data (if known). Either single value or vector of length `nrow(data)`. If `data` is a matrix and `mu` is a single value, the same mean is used for each row.

- `test.stat` – The test statistic, i.e., assumed distribution or distribution-free method for `data`. Choice of `"Normal"` or `"CSS"`. The test statistics behind the distributional options are contained within Chen and Gupta (2000) for the `"Normal"` option and Chen and Gupta (1997) for the `"CSS"` option.

The remainder of this section is a worked example considering changes in variability within wind speeds.

5.1. Case study: Irish wind speeds

With the increase of wind based renewables in the power grid, there has become great interest in forecasting wind speeds. Often modelers assume a constant dependence structure when modeling the existing data before producing a forecast. Here we conduct a naive changepoint analysis of wind speed data which are available in the R package **gstat** (Pebesma 2004). The data provided are daily wind speeds from 12 meteorological stations in the Republic of Ireland. The data has previously been analyzed by several authors including Haslett and Raftery (1989) and Gneiting, Genton, and Guttorp (2007). These analyses were concerned with a spatial-temporal model for 11 of the 12 sites. Here we consider a single site, Claremorris depicted in Figure 3.

```
R> data("wind", package = "gstat")
R> ts.plot(wind[, 11], xlab = "Index")
```

The variability of the data appears smaller in some sections and larger in others, this motivates a search for changes in variability. Wind speeds are by nature diurnal and thus have a periodic mean. The change in variance approaches within the `cpt.var` function require the data to have a fixed value mean over time and thus this periodic mean must be removed prior to analysis. Whilst there are a range of options for removing this mean, we choose to take first differences as this does not require any modeling assumptions. Following this we assume that the differences follow a Normal distribution with changing variance and thus use the `cpt.var` function. Again we compare the analyses provided by the PELT and binary segmentation algorithms.

```
R> wind.pelt <- cpt.var(diff(wind[, 11]), method = "PELT")
R> plot(wind.pelt, xlab = "Index")
R> logLik(wind.pelt)

   -like -likepen
37328.68 37856.13

R> wind.bs <- cpt.var(diff(wind[, 11]), method = "BinSeg")

Warning message:
In binseg.var.norm(coredata(data), Q, pen.value, know.mean, mu) :
The number of changepoints identified is Q, it is advised to increase Q to
make sure changepoints have not been missed.

R> ncpts(wind.bs)

[1] 5
```

Note that unlike the PELT algorithm, the binary segmentation algorithm has only found 5 changepoints. This is because we used the default value of the parameters that set `Q = 5` which results in a maximum of 5 changepoints identified. Whilst a warning message is produced, when performing an analysis using binary segmentation this should always be checked and the default increased if necessary.

```
R> wind.bs <- cpt.var(diff(wind[, 11]), method = "BinSeg", Q = 60)
R> plot(wind.bs, xlab = "Index")
R> ncpts(wind.bs)

[1] 8

R> logLik(wind.bs)

   -like -likepen
37998.37 38068.69
```

As we are considering the negative log-likelihood the smaller value provided by PELT is preferred. Even when eye-balling the results, it would appear that the PELT segmentation is more appropriate than that of the binary segmentation analysis, see Figure 3.

6. Changes in mean and variance: The `cpt.meanvar` function

The **changepoint** package contains four distributional choices for a change in both the mean and variance; Exponential, Gamma, Poisson and Normal. The Exponential, Gamma and Poisson distributional choices only require a change in a single parameter to change both the mean and the variance. In contrast, the Normal distribution requires a change in two parameters. The multiple parameter changepoint problem has been considered by many authors including Horvath (1993) and Picard, Robin, Lavielle, Vaisse, and Daudin (2005).

Each distributional option is available within the `cpt.meanvar` function which has a similar structure to the `cpt.mean` and `cpt.var` functions from previous sections. The basic call format is as follows:

```
cpt.meanvar(data, penalty, pen.value, method, Q, test.stat = "Normal", class,
  param.estimates, shape = 1)
```

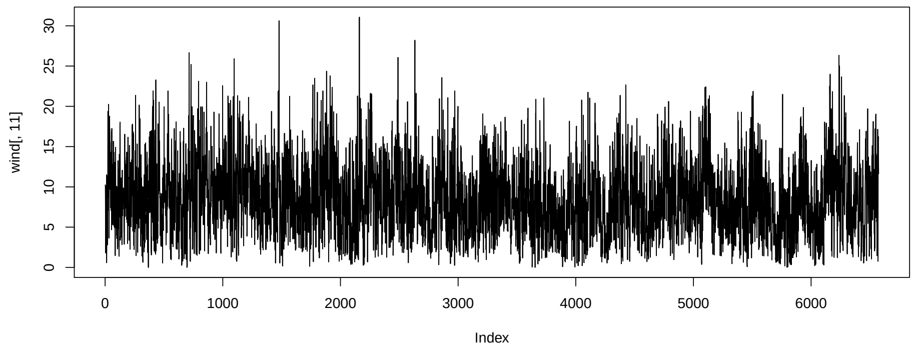
The `data`, `penalty`, `pen.value`, `method`, `Q`, `class` and `param.estimates` arguments are the same as those described for the `cpt.mean` function (see Section 4). The remaining arguments are interpreted as follows.

- `test.stat` – The test statistic, i.e., assumed distribution of `data`. Choice of `"Normal"`, `"Gamma"`, `"Exponential"` or `"Poisson"`.
- `shape` – Value of the known `shape` parameter required when `test.stat = "Gamma"`.

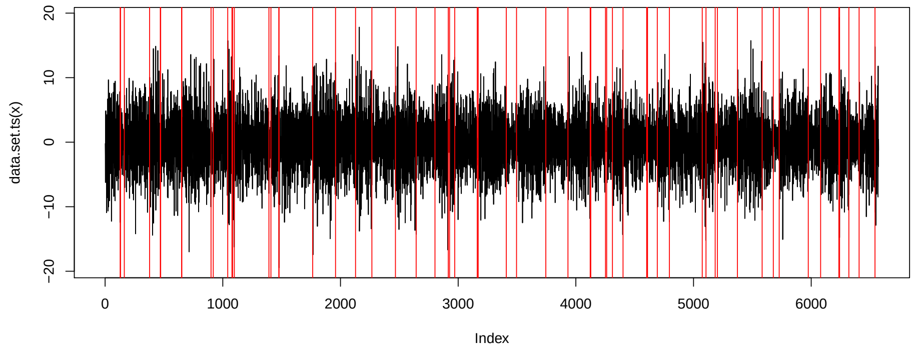
Following the format of previous sections we briefly describe a case study using data on notable inventions / discoveries.

6.1. Case study: Discoveries

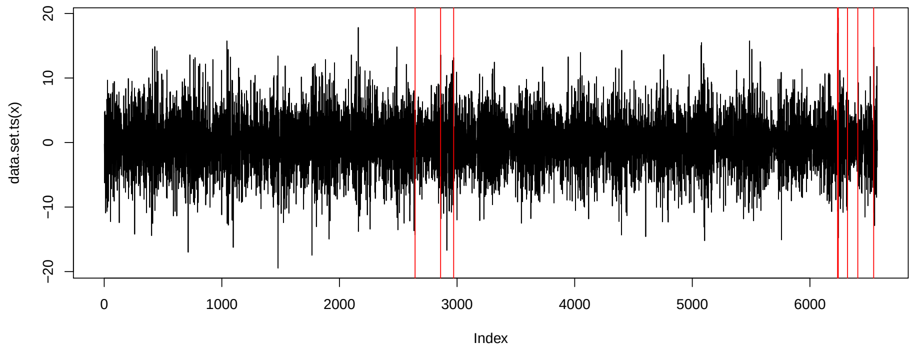
This section considers the dataset called `discoveries` available within the **datasets** package in the base distribution of R. The data are the counts of the number of "great" inventions and/or scientific discoveries in each year from 1860 to 1959. Our approach models each segment as following a Poisson distribution with its own rate parameter. Again we compare the results for both PELT and binary segmentation search methods.



(a)



(b)



(c)

Figure 3: (a) Republic of Ireland hourly wind speeds, (b) and (c) show the first differences of (a) with vertical lines depicting changepoints identified by (b) PELT and (c) binary segmentation.

```
R> data("discoveries", package = "datasets")
R> dis.pelt <- cpt.meanvar(discoveries, test.stat = "Poisson",
+    method = "PELT")
R> plot(dis.pelt, cpt.width = 3)
R> cpts.ts(dis.pelt)
```

```
[1] 1883 1888 1932 1952
```

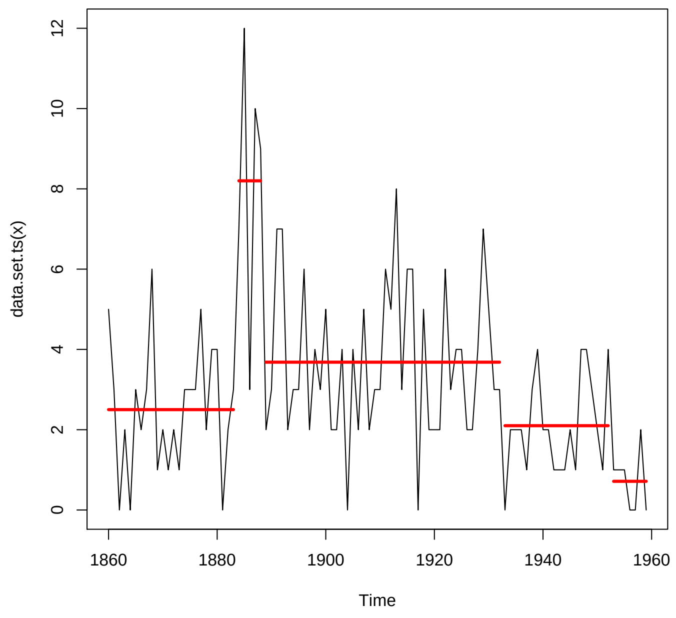


Figure 4: Discoveries dataset with identified changepoints.

```
R> dis.bs <- cpt.meanvar(discoveries, test.stat = "Poisson",
+    method = "BinSeg")
R> cpts.ts(dis.bs)

[1] 1883 1888 1932 1952
```

The number and year of the changepoints identified by both methods are the same. Here we have used the `cpts.ts` function to return the date of the changepoints rather than their position within the sequence of data.

7. Summary

The unique contribution of the **changepoint** package is that the user has the ability to select the multiple changepoint search method for analysis. The package contains three such methods: segment neighborhood; binary segmentation and PELT and this paper has described and demonstrated some differences between these approaches. The multiple changepoint search methods are available both for changes in mean and/or variance using distributional or distribution-free assumptions utilizing both established and novel methods. As such the **changepoint** package is useful both for practitioners to implement existing methods and for researchers to compare the performance of new approaches against the established literature.

Acknowledgments

The authors wish to thank Paul Fearnhead for helpful discussions and encouragement as they developed this work as well as the editor and anonymous referees for helpful feedback on earlier versions of this manuscript. R. Killick and I.A. Eckley acknowledge financial support from Shell Research Limited and the Engineering and Physical Sciences Research Council (EPSRC).

References

Auger IE, Lawrence CE (1989). "Algorithms for the Optimal Identification of Segment Neighborhoods." *Bulletin of Mathematical Biology*, **51**(1), 39–54.

Bai J, Perron P (1998). "Estimating and Testing Linear Models with Multiple Structural Changes." *Econometrica*, **66**(1), 47–78.

Birge L, Massart P (2007). "Minimal Penalties for Gaussian Model Selection." *Probability Theory and Related Fields*, **138**(1), 33–73.

Chen J, Gupta AK (1997). "Testing and Locating Variance Changepoints with Application to Stock Prices." *Journal of the American Statistical Association*, **92**(438), 739–747.

Chen J, Gupta AK (2000). *Parametric Statistical Change Point Analysis*. Birkhauser.

Davis RA, Lee TC, Rodriguez-Yam GA (2006). "Structural Break Estimation for Nonstationary Time Series Models." *Journal of the American Statistical Association*, **101**(473), 223–239.

Eckley IA, Fearnhead P, Killick R (2011). "Analysis of Changepoint Models." In D Barber, AT Cemgil, S Chiappa (eds.), *Bayesian Time Series Models*. Cambridge University Press.

Edwards AWF, Cavalli-Sforza LL (1965). "A Method for Cluster Analysis." *Biometrics*, **21**(2), 362–375.

Erdman C, Emerson JW (2007). "**bcp**: An R Package for Performing a Bayesian Analysis of Change Point Problems." *Journal of Statistical Software*, **23**(3), 1–13. URL http://www.jstatsoft.org/v23/i03/.

Erdman C, Emerson JW (2008). "A Fast Bayesian Change Point Analysis for the Segmentation of Microarray Data." *Bioinformatics*, **24**(19), 2143–2148.

Gneiting T, Genton MG, Guttorp P (2007). "Geostatistical Space-Time Models, Stationarity, Separability and Full Symmetry." In *Statistical Methods for Spatio-Temporal Systems*, pp. 151–175. Chapman & Hall/CRC.

Gupta AK, Tang J (1987). "On Testing Homogeneity of Variances for Gaussian Models." *Journal of Statistical Computation and Simulation*, **27**(2), 155–173.

Guyon X, Yao J (1999). "On the Underfitting and Overfitting Sets of Models Chosen by Order Selection Criteria." *Journal of Multivariate Analysis*, **70**(2), 221–249.

Haslett J, Raftery AE (1989). "Space-Time Modelling with Long-Memory Dependence: Assessing Ireland's Wind Power Resource." *Journal of the Royal Statistical Society C*, **38**(1), 1–50.

Hinkley DV (1970). "Inference about the Change-Point in a Sequence of Random Variables." *Biometrika*, **57**(1), 1–17.

Horvath L (1993). "The Maximum Likelihood Method of Testing Changes in the Parameters of Normal Observations." *The Annals of Statistics*, **21**(2), 671–680.

Hsu DA (1979). "Detecting Shifts of Parameter in Gamma Sequences with Applications to Stock Price and Air Traffic Flow Analysis." *Journal of the American Statistical Association*, **74**(365), 31–40.

Iacus SM (2009). ***sde**: Simulation and Inference for Stochastic Differential Equations.* R package version 2.0.10, URL http://CRAN.R-project.org/package=sde.

Killick R, Eckley I, Haynes K (2014). ***changepoint**: An R Package for Changepoint Analysis.* R package version 1.1.5, URL http://CRAN.R-project.org/package=changepoint.

Killick R, Eckley IA, Jonathan P, Ewans K (2010). "Detection of Changes in the Characteristics of Oceanographic Time-Series using Statistical Change Point Analysis." *Ocean Engineering*, **37**(13), 1120–1126.

Killick R, Fearnhead P, Eckley IA (2012a). "Optimal Detection of Changepoints with a Linear Computational Cost." *Journal of the American Statistical Association*, **107**(500), 1590–1598.

Killick R, Nam CFH, Aston JAD, Eckley IA (2012b). "changepoint.info: The Changepoint Repository." URL http://changepoint.info/.

Lai WR, Johnson MD, Kucherlapati R, Park PJ (2005). "Comparative Analysis of Algorithms for Identifying Amplifications and Deletions in Array CGH Data." *Bioinformatics*, **21**(19), 3763–3770.

Lavielle M (2005). "Using Penalized Contrasts for the Change-Point Problem." *Signal Processing*, **85**(8), 1501–1510.

Muggeo VMR (2012). ***cumSeg**: Change Point Detection in Genomic Sequences.* R package version 1.1, URL http://CRAN.R-project.org/package=cumSeg.

Nam CFH, Aston JAD, Johansen AM (2012). "Quantifying the Uncertainty in Change Points." *Journal of Time Series Analysis*, **33**(5), 807–823.

Page ES (1954). "Continuous Inspection Schemes." *Biometrika*, **41**(1–2), 100–115.

Pebesma EJ (2004). "Multivariable Geostatistics in S: The **gstat** Package." *Computers & Geosciences*, **30**(7), 683–691.

Picard F, Robin S, Lavielle M, Vaisse C, Daudin JJ (2005). "A Statistical Approach for Array CGH Data Analysis." *BMC Bioinformatics*, **6**(27), 1–14.

R Core Team (2014). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Reeves J, Chen J, Wang XL, Lund R, Lu Q (2007). "A Review and Comparison of Changepoint Detection Techniques for Climate Data." *Journal of Applied Meteorology and Climatology*, **46**(6), 900–915.

Ross GJ (2013). ***cpm**: Sequential Parametric and Nonparametric Change Detection.* R package version 1.1, URL http://CRAN.R-project.org/package=cpm.

Scott AJ, Knott M (1974). "A Cluster Analysis Method for Grouping Means in the Analysis of Variance." *Biometrics*, **30**(3), 507–512.

Sen A, Srivastava MS (1975). "On Tests for Detecting Change in Mean." *The Annals of Statistics*, **3**(1), 98–108.

Seshan VE, Olshen A (2008). ***DNAcopy**: DNA Copy Number Data Analysis.* R package version 1.24.0, URL http://www.Bioconductor.org/packages/release/bioc/html/DNAcopy.html.

Silva EG, Teixeira AAC (2008). "Surveying Structural Change: Seminal Contributions and a Bibliometric Account." *Structural Change and Economic Dynamics*, **19**(4), 273–300.

Zeileis A, Leisch F, Hornik K, Kleiber C (2002). "**strucchange**: An R Package for Testing for Structural Change in Linear Regression Models." *Journal of Statistical Software*, **7**(2), 1–38. URL http://www.jstatsoft.org/v07/i02/.

Zeileis A, Shah A, Patnaik I (2010). "Testing, Monitoring, and Dating Structural Changes in Exchange Rate Regimes." *Computational Statistics & Data Analysis*, **54**(6), 1696–1706.

Affiliation:

Rebecca Killick
Department of Mathematics & Statistics
Lancaster University
LA1 4YF, United Kingdom
E-mail: r.killick@lancs.ac.uk
URL: http://www.lancs.ac.uk/~killick/

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/

Volume 58, Issue 3 *Submitted:* 2013-01-10
June 2014 *Accepted:* 2014-02-23