Procedia Computer Science

Volume 53, 2015, Pages 39–47

2015 INNS Conference on Big Data

Segmentation of geophysical data: a big data friendly approach

David R.B. Stockwell[1], Ligang Zhang[1], and and Brijesh Verma[1]

Centre for Intelligent and Networked Systems at Central Queensland University, Australia
(d.stockwell@cqu.edu.au, ligzhang@gmail.com, b.verma@cqu.edu.au)

Abstract
A new scalable segmentation algorithm is proposed in this paper for the forensic determination of level shifts in geophysical time series. While a number of segmentation algorithms exist, they are generally not 'big data friendly' due either to quadratic scaling of computation time in the length of the series N or subjective penalty parameters. The proposed algorithm is called SumSeg as it collects a table of potential break points via iterative ternary splits on the extreme values of the scaled partial sums of the data. It then filters the break points on their statistical significance and peak shape. Our algorithm is linear in N and logarithmic in the number of breaks B, while returning a flexible nested segmentation model that can be objectively evaluated using the area under the receiver operator curve (AUC). We demonstrate the comparative performance of SumSeg against three other algorithms. SumSeg is available as an R package from the development site at http://github.com/davids99us/anomaly.

Keywords: data segmentation, data size, geophysical, change points

1 Introduction

Has the level of a time series changed due to natural variation or an external influence? Abrupt changes in level can be due to instrument faults or reconfiguration and so are necessary for QA/QC on data from weather stations [1] and automatic tide or stream level gauges. The level changes in a segmentation model may also represent gene expression in micro-array comparative genomic hybridization data [2], regime shifts in climate data [3], breakouts in stock prices, twitter or web service logs, or features of interest in weak machine learning classifiers [4]. Finding an optimal multi-segmentation is challenging as the number of potential segments grows exponentially in N (as the number of potential ways to segment a sequence is equal to the number of subsets of N, or 2^N). Thus segmentation is an example of a big data problem where larger data sets call for new approaches [5]. The following are key performance criteria:

1. Linear or better order of increase in the computational cost of data length N and number of breaks B. Quadratic growth in computation cost does not scale.

2. Reliable application to noisy real world data with gaps, missing values and errors. Some algorithms fail on missing data, while others become unreliable or produce biased statistical measures.

Selection and peer-review under responsibility of the Scientific Programme Committee of INNS-BigData2015

doi:10.1016/j.procs.2015.07.277

3. A well-known and transparent statistical framework. Type I/II errors (false positives and negatives) are unfortunately incorporated in only a few segmentation algorithms (CLAC and ACE) [2].

4. Objective model evaluation metrics like the Receiver Operating Characteristic (ROC) and optimization using the Area Under Curve (AUC).

5. Robust to non-normal data. Some geophysical data such as rainfall is highly non-normal, and most have strong periodic elements at a fine scale.

6. Flexible examination of models nested by confidence level. Output of a single model is not conducive to "drilling down" into sections with uncertainty.

This paper has three major contributions: 1) a novel ternary split segmentation algorithm available as a R package based on minimum and maximum extrema of the partial sums; 2) identification of linearity in length of data and number of breaks as crucial computational criteria for scaling segmentation; 3) use of the familiar learning statistical metric of the AUC as the criterion for breakpoints.

2 Related Work

Statistical methods for testing the homogeneity assumptions of linear models have a long history [6, 7]. An example of a simplistic algorithm would be to test each point for possible level changes using a goodness-if-fit test such as the standard normal homogeneity test (SNHT) [8]. As the goodness of fit of a segmented model increases without limit with additional breaks, additional penalties and measures such as AIC (Akaike Information Criterion) and BIC (Bayesian Information Criterion) have been used to control over-fitting [9]. Combining the goodness of fit and penalties for over-fitting, the problem becomes a minimization of a global cost function:

$$\sum_{j=1}^{B} \mathcal{C}(y_{j:next(j)}) + \beta(B) \tag{1}$$

where $\mathcal{C}$ is a goodness of fit function for each segment and β is the penalty function usually over the number of breaks B. While the brute-force search for the optimum of equation 1 is $\mathcal{O}(2^N)$ dynamic programming can reduce the exact solution to $\mathcal{O}(BN^2)$ [10]. Polynomial growth in computation cost still imposes practical limits on N, and so is undesirable for big data analysis. Users face a confusing choice of the ideal fit and penalty functions so introducing potential operator bias (see [11]).

Segmentation algorithms are widely used in the detection and correction of meteorological data sets as station moves or reconfigurations often cause step changes in temperature. The International Surface Temperature Initiative (ISTI) is building global homogeneous temperature products [12] from a network of inhomogenous meteorological station data. Such projects need a very reliable segmentation method in the analysis chain. This is because segmentation is applied to a contrast with regional climatology, using either a weighted average of neighbors [13] or an exhaustive pairwise comparison [14]. While this reduces the noise from common climate variations, biases and errors may be introduced from the comparators [15]. Analysis of the original data collected at daily and shorter intervals is generally preferable as additional steps in the analysis chain such as monthly or annual aggregation can also introduce bias [16].

Here we compare the algorithm SumSeg with three representative approaches to multiple change-point detection algorithms supplied in the R package **changepoint** [11]. BinSeg, a binary segmentation algorithm, performs a recursive descent on binary splits blocking on segments

where no split is found. BinSeg is an approximate minimization of equation 1 due to the truncated search [17], as optimization of equation 1 would require descent into unitary sections to find outliers or short breakouts. The segment neighborhood algorithm SegNeigh [10] minimizes equation 1 exactly using a dynamic programming technique, but the computational complexity is $\mathcal{O}(BN^2)$ and so is unsuitable for longer sequences. The Pruned Exact Linear Time (PELT) algorithm finds the optimum of equation 1 [11] using a modern bottom-up dynamic programming and pruning approach, and is $\mathcal{O}(Nlog(N))$, or $\mathcal{O}(N)$ subject to certain assumptions such as the number of change points increases linearly as the data set grows.

3 Proposed Algorithm

The algorithm is motivated by the relationship between a sequence and its integral and derivative. Consider an original sequence y_i where $i = 1...N$ is transformed into a partial sum $S_i = \sum y_i$ where $i = 1...N$. In a scaled sequence with a change in level like a square wave, the derivative consists of two spikes, one positive and one negative. The integral has the form of a rotated 'Z' where the turning points coincide with the level shifts of the original series, or the spikes of the derivative. Thus, exhaustive comparison of means around every possible break in the original series, as performed by exact methods, is not needed.

The integral (or partial sum) is more sensitive than the derivative due to supression of random noise and robustness to non-normal distributions, a consequence of the central limit theorem [9]. A drawback of the integral is that both level shifts and trends will produce extrema. Gradual trends may be distinguished from level shifts by the peak height/width ratio in the partial sum as level shifts produce sharper peaks than gradual trends. This will be addressed in our future work.

The critical value for a standardized sequence is $crit = p\sqrt{n}$ where p is the significance level and n is the length of an individual segment. The *crit* can be tested in the presence of gaps and missing values. Listing 1 shows the essential R code for algorithm, using an extension of the base R construct data.frame called **data.table** [18], employing a powerful SQL-like $DT[where, select|update, groupby]$ idiom suited to large data applications.

Listing 1: Listing of the main function in SumSeg in the R language data.frame idiom.

```
model.ss<-function(D,max.breaks=4) {
#Initialise cut table with variables and start and end cuts
cuts=\pkg{data.table}(value=NA,F1=factor(NA),F2=factor(NA),
    cusum=NA,crit=NA,index=D[,c(index[1],index[.N]+1)],width=NA)
for (i in 1:max.breaks) \{
    cuts[,F1:=F2] # replace old groups
    D[,F2:=cut(index,cuts$index),by=F1] # new groups
    D[,cusum:=cumsum(scale(value)),by=F1] # calculate partial sum
    D[,crit:=crit(.N,1),by=F1] # calculate critical value
    # rows with maximum cusum by each group
    max=D1[,.(index=index[which.max(cusum)],width=.N),by=F2]
    # rows with minimum cusum by each group
    min=D1[,.(index=index[which.min(cusum)],width=.N),by=F2]
    cuts=rbind(cuts,D[max],D[min]) # append new cuts
    cuts=cuts[,.SD[1],by=index] # only unique cuts
}
cuts[,prob:=pnorm(abs(cusum),0,crit)] # probability
cuts[,peak:=(abs(cusum)-crit)/width] # peak width
return(cuts)
}
```

As the number of groups (or segments) increases by $3^i$with each iteration, the residuals are quickly driven towards the baseline as shown in Figure 2 (Iterations). As no step in the algorithm is greater than $\mathcal{O}(N)$ the computational cost of the whole algorithm is $\mathcal{O}(N)$ or

linear in N. The speed may be improved further by pruning or recording the maximum and minimum values in conjunction with the calculation of the partial sums. The search blocks when the maximum and the minimum are the endpoints of the segment.

The list of potential breaks in the output model (an example of which is shown on Table 1) can be filtered by criteria such as the AUC, a user-defined significance value, or peak width/height ratio. The algorithm embodies the cost function in equation 1 via the AUC; the point shown on the ROC curve (Figure 2 ROC). The AUC is defined so that any other point increases either omission or commission errors – a two term objective function.

4 Experiments

While actual results vary between runs, Figure1 shows a typical segmentation (red lines) on 4000 simulated random data with SD=1 and level changes of +1.0, -1.0 and +0.2 (dashed green line). All algorithms identified the three breaks correctly (true positives). SegNeigh, BinSeg and particularly PELT introduced tiny breaks shown as red dots (false positives). SumSeg introduced a small false positive at location 2249, but with prob=0.84 (see Table 1 for results), this break would be eliminated at the default AUC value of 0.9. A typical worst case computation time was SumSeg=0.054s, SegNeigh=37.272s, PELT=0.231s, and BinSeg=0.338s. The SegNeigh method is considerably slower than the other three methods and so was not evaluated further.

Table 1 is the SumSeg model on the 4000 simulated data points in Figure 1. The variables in the table are as follows: *index* - the location of the putative break, *value* - the sequence value at the break point, $F2$ the range of the segment (group) in which the break was found, *cusum* - the cumulative sum of the value by segment, *crit* - the critical value at one standard deviation for that segment, *prob* - the confidence level of that break, *peak* - the peak height/width ratio.

Table 1: **A SumSeg model**: Listing of the variables in the table of breaks on the simulated data in Figure 1.

	index	value	F2	cusum	crit	width	prob	peak
1	2001	0.79	(1,4e+03]	405.46	63.24	2989	1.00	0.11
2	999	-0.68	(1,4e+03]	-246.79	63.24	2989	1.00	0.06
3	3003	-0.28	(2e+03,4e+03]	-136.49	44.71	1384	1.00	0.07
4	2249	-1.62	(2e+03,4e+03]	-44.78	44.71	1384	0.84	0.00

Figure 2 (Length) is a linear-linear plot of length vs. computation time for SumSeg, PELT and BinSeg. SegNeigh is not tested due to the excessively long computation time. The computation time of PELT increases quickly and slightly accelerates with increasing N. The SumSeg algorithm has an intermediate slope while BinSeg is fast and linear. Figure 2 (Breaks) is a log-log plot of breaks vs. computation time for SumSeg and BinSeg with constant N. SumSeg is linear, indicative of $log(B)$ growth, while BinSeg grew much faster than $log(B)$. We note that BinSeg is limited to 2001 potential breaks while SumSeg is not limited. PELT is not plotted as it does not appear to have a parameter controlling the maximum number of breakpoints. A similar effect can be achieved to a degree by adjusting the penalty parameter *pen.value* which appears to represent the significance of the global model.

Wind speed data from Claremorris in the Republic of Ireland, available in the R package **gstat** [19], records 6574 daily wind speeds from 1961 to 1978. The dataset has a strong fine-scale structure due to daily periodicity. Figure 3 (Wind Speed) shows the coarsest segmentation achievable with the PELT (green) using $pen.value = 10^{-15}$. SumSeg produces a close approximation to PELT with parameter $max.breaks = 3$ (i.e. a maximum of 27 breaks).

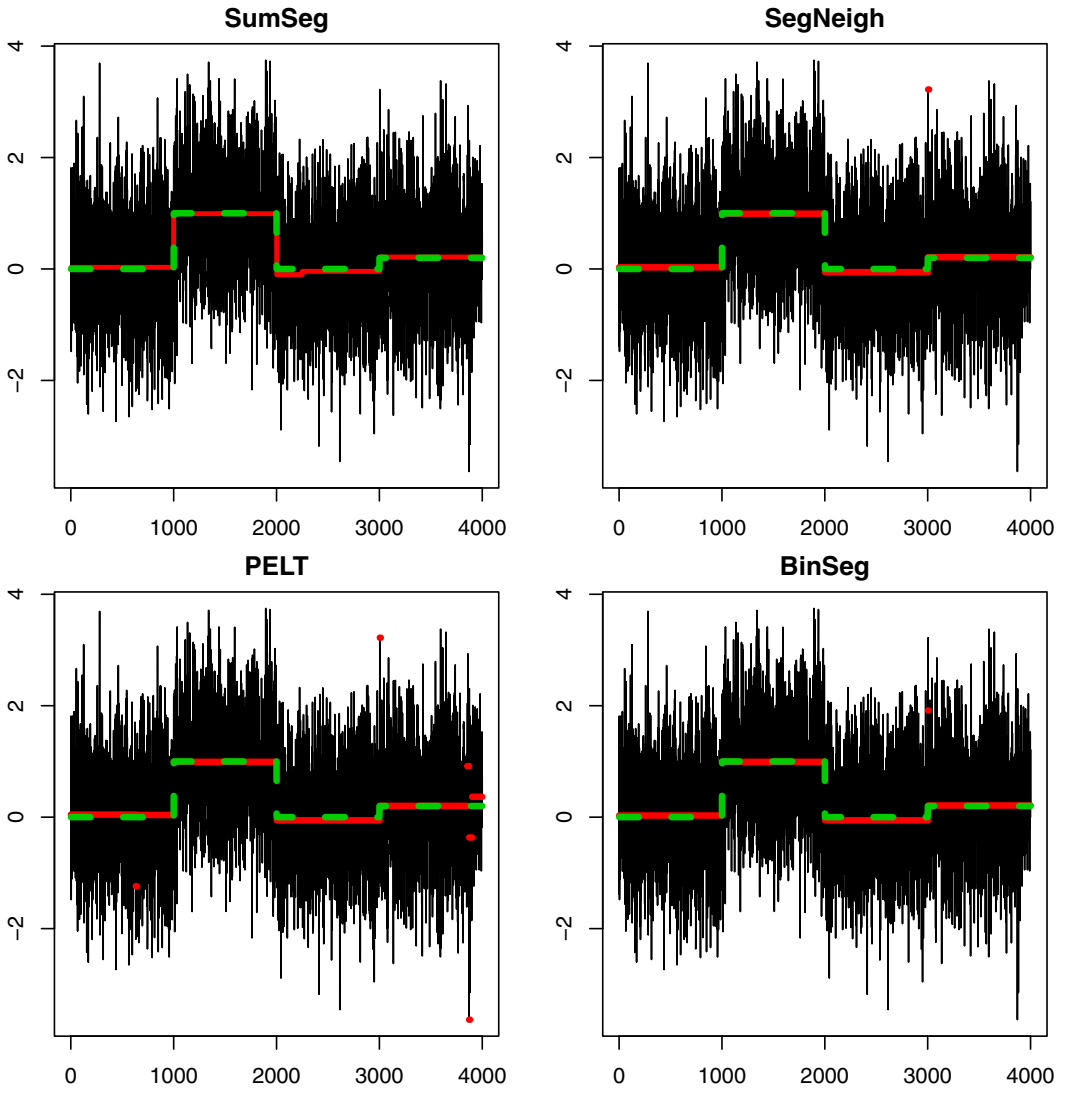

Figure 1: Recovery of known level changes (green) from the four algorithms on simulated data (in red): **SumSeg, SegNeigh, BinSeg** and **PELT**. The simulated data has length N=4000, level changes at 1000, 2000, and 3000, and the means for each segment are 0, 1, 0, and 0.2 with introduced random noise mean=0 and SD=1. The PELT algorithm introduced a number of small additional breaks (false positives). The computation time was SumSeg=0.054s SegNeigh=37.272s PELT=0.231s BinSeg=0.338s.

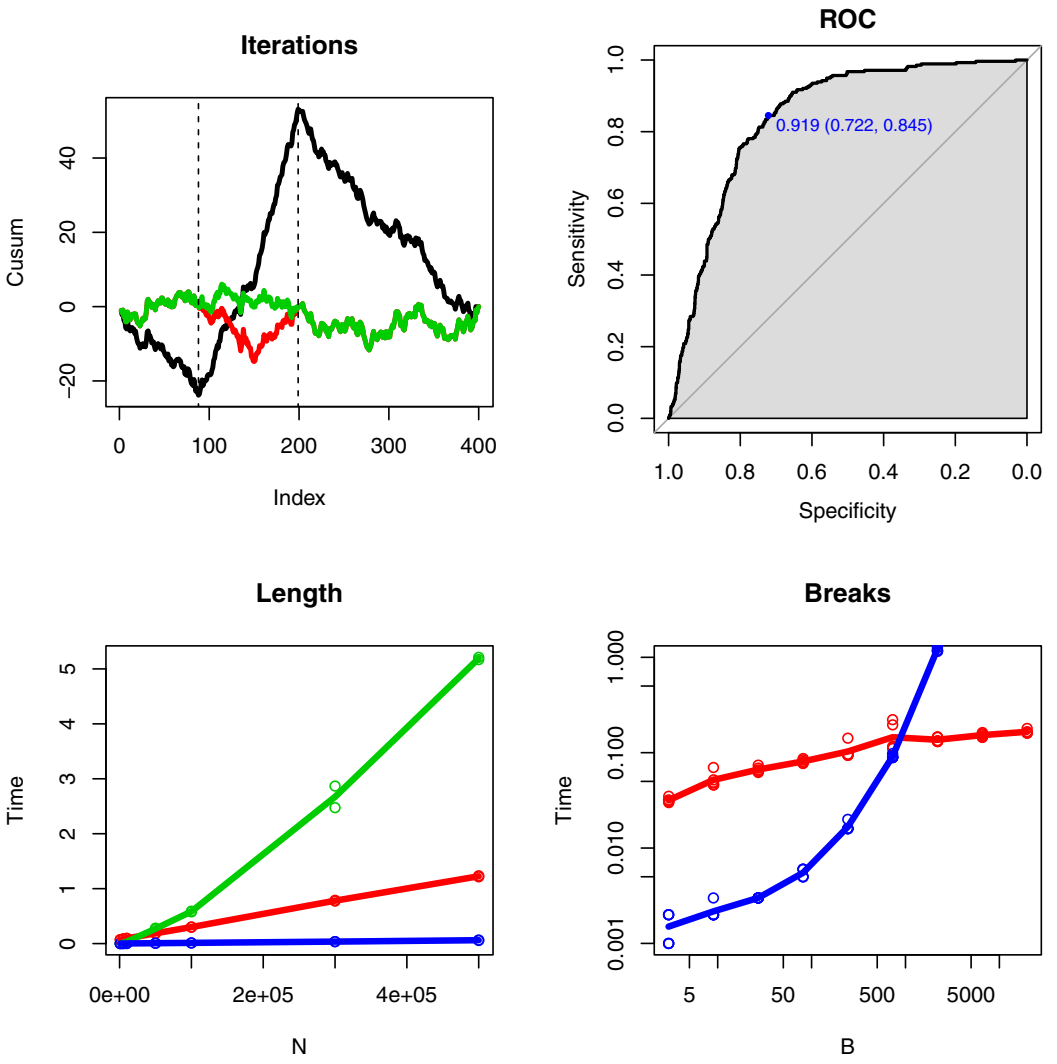

Figure 2: Simulated Data: **Iterations, ROC, Length** and **Breaks**. **Iterations** illustrates three successive iterations of the cumulative sums, with the break points of the first cumulative sum (black) shown by dashed vertical lines. The data is split into three segments at these two break points and rescaled, yielding the second (red, largely obscured by third green line) cumulative series. The final splits yield the green line. **ROC** illustrates the Receiver Operating Curve on simulated data. The value of the Area Under Curve for these data is around 0.9 for these data. Note that PELT, BinSeg and SegNeigh cannot be plotted in this manner as they do not assign Type I error probabilities to the individual breaks. **Length** shows the computation time with increasing data and constant breakpoints for SumSeg (red), PELT (green) and BinSeg (blue) on a linear-linear plot. **Breaks** shows computation time vs. number of breaks for SumSeg (red) and BinSeg (blue) on a log-log plot.

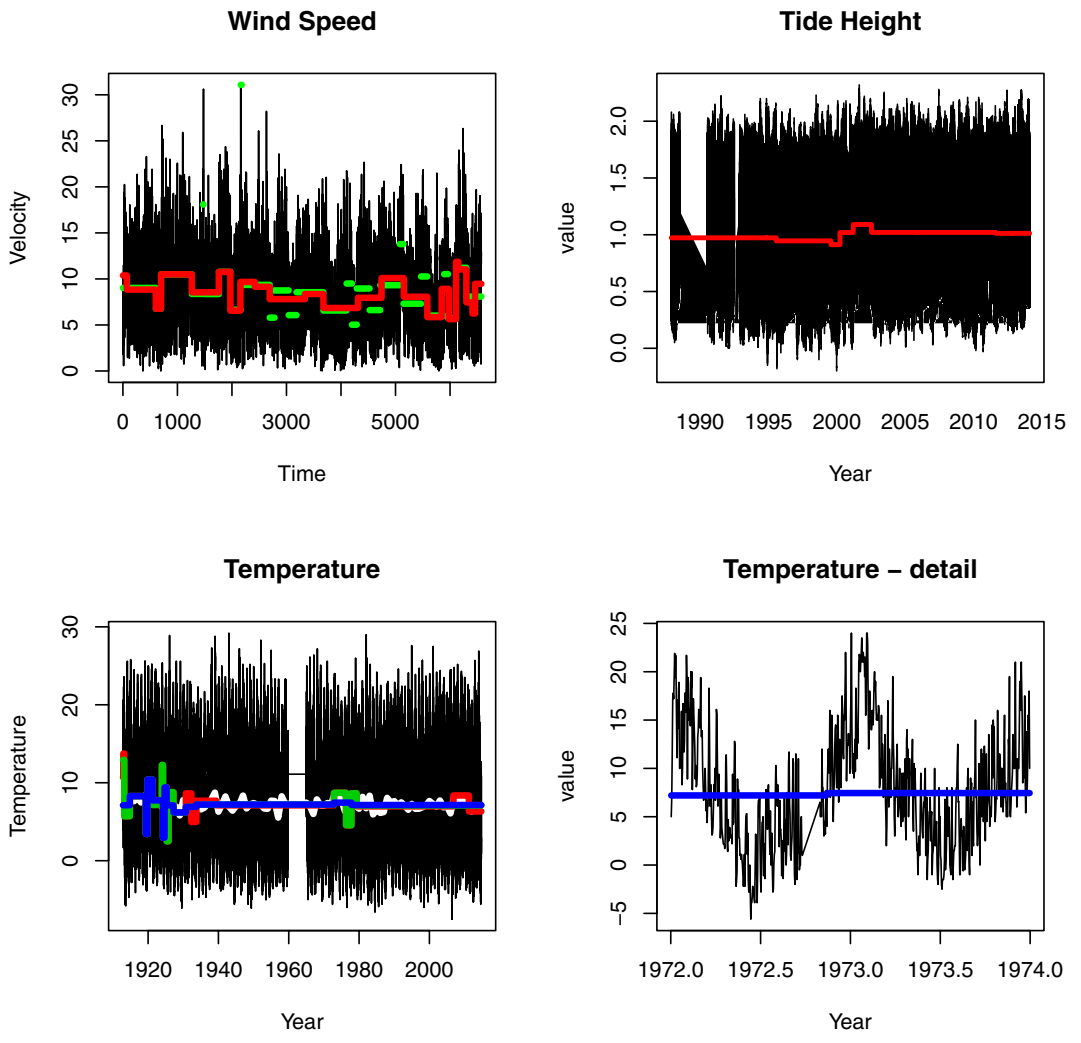

Figure 3: Real Data: **Wind Speed**, **Tide Height**, **Temperature** and **Temperature - detail**. **Wind Speed**: analysis of the PELT (green) and SumSeg (red) algorithms as they segment daily wind speeds for a single site, Claremorris in the Republic of Ireland. **Tide Height**: Port Hacking Tide Gauge Data consisting of 766,902 ocean heights collected every 15 minutes from 1987 to 2014. Note the notches at the lower margin of the data due to a malfunction are indicated in the segmentation (red line above). The gaps and missing values, and anomalies were caused by the tide gauge not 'bottoming-out' periodically can be seen as a notch in the minimum of the data (gray), and a corresponding step in the segmented model (red). **Temperature**: Rutherglen minimum temperature from Climate Data Online (CDO) raw data with three nested segmentation models at three levels of confidence: 0.9 (red), 0.95 (green) and 0.99 (blue). **Temperature - detail**: Zoom-in location of Rutherglen showing distinct break in the daily time series.

Figure 3 (Tide Height) shows 766,902 measurements of tide data recorded every 15 minutes from the Port Hacking Tide Gauge in Australia between 1987 to 2014. SumSeg identifies artifacts in the data seen as the notches at the lower margin of the data around 2002, apparently caused by a sticking gauge mechanism.

Figure 3 (Temperature) shows a nested segmentation model of 34,642 daily minimum temperature measurements taken at Rutherglen from 1910 to the present downloaded from the Australian Bureau of Meteorology website. A number of significant step changes (blue - 0.99CL) occurred in the early century, while level changes also occurred in the 1970's (green - 0.90CL) and 2010's (red - 0.95CL). One advantage of analyzing daily data is the capacity to pinpoint the exact date of a change. As an example, Figure 3 (Temperature-detail) shows a small level change in the early 1970's coinciding with a gap in the data that may indicate an undocumented station move or reconfiguration.

From the above experimental results, it can be concluded that the proposed algorithm is able to accurately detect the changes in data sequences, and effectively segment the data into homogeneous subsets with a linear computational cost.

5 Conclusions

A novel segmentation algorithm for large data has been proposed and evaluated on simulated and real world data. The computation time of SumSeg is linear in length of the series and logarithmic in number of breaks. It outputs a set of possible break points and their statistical significance, which could then be evaluated and optimized using the receiver operating curve and AUC value. We compared the suitability of SumSeg for big data applications with three alternative algorithms: SegNeigh an exact algorithm, BinSeg a binary recursive, and PELT a modern dynamic programming algorithm. SegNeigh is disadvantaged by a quadratic increase in computation time with N and PELT increased rapidly also. While fast and linear in N BinSeg is disadvantaged by quadratic increase in time with number of breaks B. We were unable to produce coarse segmentations in PELT as demonstrated on the wind speed data. The utility of the capacity generate a range of coarse to fine nested segmentations was demonstrated on the temperature data from Rutherglen. The data and code, available as an R package at http://github.com/davids99us/anomaly, will be further refined through application to real geophysical data problems.

6 Acknowledgments

We are grateful to Bill Johnston for reviewing an early draft and supplying the Port Hacking tide gauge data, which is curated by the New South Wales Public Works, Office of Finance and Services.

References

[1] P. Domonkos, V. Venema, I. Auer, O. Mestre, M. Brunetti, The historical pathway towards more accurate homogenisation, Advances in Science and Research 8 (2012) 45–52. doi:10.5194/asr-8-45-2012.
URL http://www.adv-sci-res.net/8/45/2012/

[2] W. R. Lai, M. D. Johnson, R. Kucherlapati, P. J. Park, Comparative analysis of algorithms for identifying amplifications and deletions in array cgh data, Bioinformatics (Oxford, England) 21 (19) (2005) 3763–70.
URL http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2819184/

[3] D. R. B. Stockwell, A. Cox, Structural break models of climatic regime-shifts: claims and forecasts, ArXiv e-printsarXiv:0907.1650.

[4] S. Chowdhury, B. Verma, D. Stockwell, A novel texture feature based multiple classifier technique for roadside vegetation classification, Expert Systems with Applications.

[5] M. W. Hall, R. M. Kirby, F. Li, M. D. Meyer, V. Pascucci, J. M. Phillips, R. Ricci, J. E. van der Merwe, S. Venkatasubramanian, Rethinking abstractions for big data: Why, where, how, and what, CoRR abs/1306.3295.
URL http://arxiv.org/abs/1306.3295

[6] J. M. Craddock, Methods of comparing annual rainfall rrecord for climatic purposes, Weather 34 (9) (1979) 332–346. doi:10.1002/j.1477-8696.1979.tb03465.x.
URL http://dx.doi.org/10.1002/j.1477-8696.1979.tb03465.x

[7] T. Buishand, Some methods for testing the homogeneity of rainfall records, Journal of Hydrology 58 (1982) 11–27.
URL http://www.sciencedirect.com/science/article/pii/002216948290066X

[8] H. Alexandersson, A homogeneity test applied to precipitation data, Journal of Climatology 6 (6) (1986) 661–675. doi:10.1002/joc.3370060607.
URL http://dx.doi.org/10.1002/joc.3370060607

[9] A. Zeileis, C. Kleiber, Validating multiple structural change models – a case study, J. Appl. Econ. 20 (5) (2005) 685–690.
URL http://dx.doi.org/10.1002/jae.856

[10] I. E. Auger, C. E. Lawrence, Algorithms for the optimal identification of segment neighborhoods., Bull Math Biol 51 (1) (1989) 39–54.

[11] R. Killick, I. Eckley, changepoint: An r package for changepoint analysis, Journal of Statistical Software 58 (3) (2014) 1–19.

[12] K. Willett, C. Williams, I. T. Jolliffe, R. Lund, L. V. Alexander, S. Brönnimann, L. A. Vincent, S. Easterbrook, V. K. C. Venema, D. Berry, R. E. Warren, G. Lopardo, R. Auchmann, E. Aguilar, M. J. Menne, C. Gallagher, Z. Hausfather, T. Thorarinsdottir, P. W. Thorne, A framework for benchmarking of homogenisation algorithm performance on the global scale, Geoscientific Instrumentation, Methods and Data Systems 3 (2) (2014) 187–200. doi:10.5194/gi-3-187-2014.
URL http://www.geosci-instrum-method-data-syst.net/3/187/2014/

[13] D. A. Rhoades, M. J. Salinger, Adjustment of temperature and rainfall records for site changes, International Journal of Climatology 13 (8) (1993) 899–913. doi:10.1002/joc.3370130807.
URL http://adsabs.harvard.edu/abs/1993IJCli..13..899R

[14] M. J. Menne, C. N. Williams, Homogenization of Temperature Series via Pairwise Comparisons, Journal of Climate 22 (7) (2009) 1700–1717. doi:10.1175/2008JCLI2263.1.
URL http://adsabs.harvard.edu/abs/2009JCli...22.1700M

[15] D. R. Stockwell, Is Temperature or the Temperature Record Rising?, in: Australian Environment Foundation Conference, Sydney, Australia, 2012, p. 10.
URL http://vixra.org/abs/1209.0088

[16] O. Mestre, C. Gruber, C. Prieur, H. Caussinus, S. Jourdain, Splidhom: A method for homogenization of daily temperature observations, J. Appl. Meteor. Climatol. 50 (11) (2011) 2343–2358. doi:10.1175/2011JAMC2641.1.
URL http://dx.doi.org/10.1175/2011JAMC2641.1

[17] E. Terzi, P. Tsaparas, Efficient algorithms for sequence segmentation., in: SDM, SIAM, 2006, pp. 316–327.

[18] M. Dowle, R's data.table package extends data.frame. (February 2015).
URL https://github.com/Rdatatable/data.table/wiki

[19] E. Pebesma, spacetime: Spatio-temporal data in r, Journal of Statistical Software 51 (7) (2012) 1–30.
URL http://www.jstatsoft.org/v51/i07