Finding Patterns in Time Series

James E. Gentle

July 10, 2017

1 Introduction

Many really large datasets are time series, and such datasets present unique problems that arise from the passage of time reflected in the datasets. A problem of current interest is clustering and classification of multiple time series; see, for example, **?**, **?**, **?**, and **?**. When various time series are fitted to models, the different time series can be grouped into clusters based on the fitted models. If there are different identifiable classes of time series, the fitted models can be used to classify new time series.

For massive time series datasets, any assumption of stationarity is not likely to be met. It is generally futile to attempt to model large time series using traditional parametric models.

In all statistical models, we seek to identify some random variable with zero autocorrelations whose realizations are components of the observable variables. The model is then composed of two parts, a systematic component plus a random component.

The problem in modeling time series is identification of any such random variable in a model over a long time period, or even in a short time period when the data are massive.

Any useful time series model that extends over a lengthy time period must either be very weak, that is, a model in which the signal-to-noise ratio is relatively small, or else must be very complex with many parameters.

A common approach to model building in time series is to break the series into separate regimes and to identify an adequate local model within each regime. In this case, the problem of clustering or classification can be addressed by use of sequential patterns of the models for the separate regimes.

Regime Descriptors; Local Models

Within a particular time regime the time series data exhibit some degree of commonality that is captured in a simple model. The model may specify certain static characteristics such as average value (mean, median, and so on) or scale (variance, range, and so on). The model may also specify certain time-dependent characteristics such as trends or autocorrelations. The model within any regime may be very specific and may fit most of the observations within that regime,

or it may be very general with many observations lying at some distance from their fitted values. For the data analyst, the choice of an appropriate model presents the standard tradeoff between a smooth fit and an "over" fit.

The beginning and ending points of each regime are important components of the model. Independent of the actual beginning and ending points, the length of the regime is also an important characteristic.

The model may be formulated in various ways. For purposes of clustering and classification of time series, it may be desirable for the model to be one of a small pre-chosen set of models. It may also be desirable that the individual characteristics be specified as one a particular small set. For example, if the model specifies location, that is some average value within the regime, we may use categorical labels to specify ranked levels of location; "a" may denote small, "b" may denote somewhat larger average values, and so on. These relative values are constant within a given regime, but the set of possible categories depends on the values within other regimes in the time series.

Specifying a model in place of the full dataset allows for significant data reduction. Substitution of the individual values within a regime by the sufficient statistical descriptors is an important form of data reduction in time series.

Changepoints

Once we accept that different models (or models with different fitted parameters) are needed in different regimes, the main problem now becomes identification of the individual regimes; that is, identification of the *changepoints* separating regimes.

The complexity of this problem depends to a large extent on the "smoothness" of our individual models; if the models are linear in time, then changepoints are easier to identify than if the models are nonlinear in time or if they involve features other than time, such as autoregressive models.

The two change points that determine the extent of a regime together with the sufficient statistical descriptors describing the regime may be an adequate reduction of the full set of time series data within the regime.

Patterns

Once regimes within a time series are identified, the patterns of interest now become the sequences — or subsequences — of local models for the regimes.

Between any two changepoints, we have a local model, say $\mu_i(t)$. A particular sequence of local models, $\mu_i(t), \mu_{i+1}(t), \ldots, \mu_{i+r}(t)$, defines a *pattern*. We will often denote a pattern in the form P_{ri}, where $P_{ri} = (\mu_i(t), \mu_{i+1}(t), \ldots, \mu_{i+r}(t))$. While the model is a *function* of time together with descriptions of other model components of the temporal relationships, such as the probability distribution of a random "error" component, we may represent each $\mu_i(t)$ as a vector whose elements quantify all relevant aspects of the model.

Clustering, Classification, and Prediction

There is considerable interest currently in learning in time series data. "Learning" generally means clustering and/or classification of time series. This is one of the main motivations of our work in pattern recognition within time series.

Forecasting or prediction is also an important motivation for time series analysis, whether we use simple trend analysis, ARMA-type models, or other techniques of analysis.

Prediction in time series, of course, is often based on unfounded hopes. We view the prediction problem as a simple classification problem, in which statistical learning is used to develop a classifier based on patterns. The response could be another local model within the class of local models used between changepoints, or the response could be some other type of object, such as a simple binary "up" or "down". The length of time over which the prediction is made must, of course, be considered in the classification problem.

Measures of Similarity/Dissimilarity

Clustering or classification is often based on some metric for measuring dissimilarity of elements in a set. For clustering and classification of time series or subsequences of time series based on patterns, we need a metric $\rho(P_{ri}, P_{sj})$, where P_{ri} is a pattern consisting of a sequence $\mu_i, \mu_{i+1}, \ldots, \mu_{i+r}$ and P_{sj} is a pattern consisting of a sequence over s regimes beginning at the j^{th} one.

Outline

In the following we discuss methods for identifying changepoints in a univariate time series. In massive datasets a major challenge is always that of overfitting. With so much data, very complex models can be developed, but model complexity does not necessarily result in better understanding or in more accurate predictions.

We will generally consider linear models, either simple constant models or simple linear trends. By restricting our attention to models that are linear in time, we avoid some kinds of overfitting. In smoothing time series using a sequence of linear models, "overfitting" is the identification of spurious changepoints.

Our main concern will be on the identification of changepoints, and we will emphasize a technique called *alternate trends smoothing*.

After identification of changepoints, we briefly discuss the problem of defining patterns. The objectives of defining and identifying patterns are twofold: to cluster and/or to classify sets of time series, and to predict future values or trends in a time series.

Although we do not emphasize any specific area of application, some of our work has been motivated by analysis of financial time series, so we occasionally refer to financial time series data, in particular, to series of stock prices or of rates of return.

2 Data Reduction and Changepoints

Analysis of massive data sets, whether they are time series or not, often begins with some form of data reduction. This usually involves computation of summary statistics that measure central tendencies and other summary statistics that measure spread. These two characteristics of a dataset are probably the most important ones for a stationary population in which a single simple model is adequate.

Even assuming a single model for all data, just concentrating on summary measures can miss important information contained in some significant individual observations. These significant observations are often the extreme or outlying points in the dataset. One simple method of analyzing a time series is just to assume a single constant model and to identify the extreme points, say the 10% outliers, with respect to that model. These outliers may carry a significant amount of information contained in the full dataset. The set of outliers may be further reduced. **?**, for example, described a method for successively identifying extreme points in a time series for the purpose of data reduction. The extreme points alone provide a useful summary of the entire time series.

Another type of significant point in a time series is one that corresponds to a change in some basic characteristic of the time series. A changepoint may or may not be an extreme point. Changepoints can also be used for data reduction because they carry the most significant information, at least from one perspective.

In a time series, a changepoint is a point in time at which some property of interest changes. A changepoint, therefore, has meaning only in the context of a model. The model for the observable data may be some strong parametric model, such as an ARMA model, or it may be some weak parametric model, such a constant median and nothing more. In the former case, a changepoint would be a point in time at which one of the parameters changes its value. (Here, we are assuming ARMA models with constant parameters.) In the latter case, a changepoint would be any point at which the median changes. A changepoint may also be a point in time at which the class of appropriate model changes. Perhaps an ARMA model is adequate up to a certain point and then beyond that the constant variance assumption becomes entirely untenable.

From one perspective, the problem of identification of changepoints can be viewed as just a part of a process of model building. This, of course, is not a well-posed problem without further restrictions, such as use of some pre-selected class of models and specification of criteria for ranking models.

In the following, we will focus on identification of changepoints in simple piecewise linear models of an observable random variable. We do not assume finite moments, so we will refer to the parameter of central tendency as the "median", and the parameter of variability as the "scale". We will also focus most of our study on univariate time series, although we will consider some extensions to multivariate series.

There is a vast literature on identification of changepoints, but we do not attempt any kind of general review; rather we discuss some of the methods that

have proven useful for the identification of patterns.

2.1 Piecewise Constant Models

The simplest model for time series with changepoints is one in which each regime is modeled by a constant. The constant is some average value of the data over that regime. For our purposes, the nature of that "average" is not relevant; however, because of possibly heavy tails in the frequency distributions and asymmetry of the data, we often think of that average as a median.

There are various approaches to modeling time series with median values that change over time. The first step in any event is to determine the breakpoints. Sometimes, when the data-generating process is indeed a piecewise constant model, the breakpoints may be quite apparent, as in Figure 1.

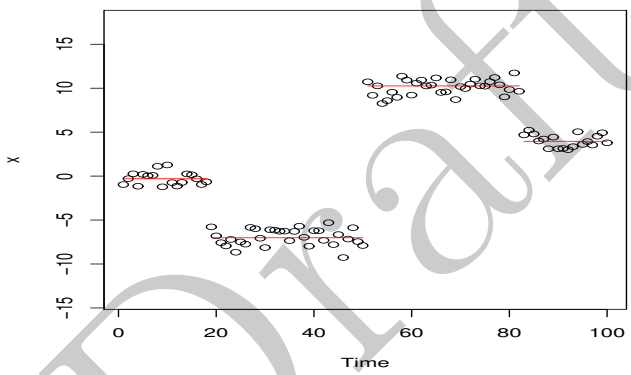

Figure 1: Time Series Following a Piecewise Constant Model

In other cases, we may choose to approximate the data with a piecewise constant model, as in Figure 2, even though it is fairly obvious that the underlying data-generating process is not piecewise constant or even piecewise linear.

There are various straightforward ways of determining the values of the approximating constants. A simple batch process is to use sample quantiles of the data that correspond to some parametric model, such as a normal distribution.

2.2 Models with Changing Scales

Piecewise constant models, such as the data in Figure 1 seem to follow, or other simple models for changing location may not to be of much interest, but there is a type of derived financial data that exhibit similar behavior. It is rates of

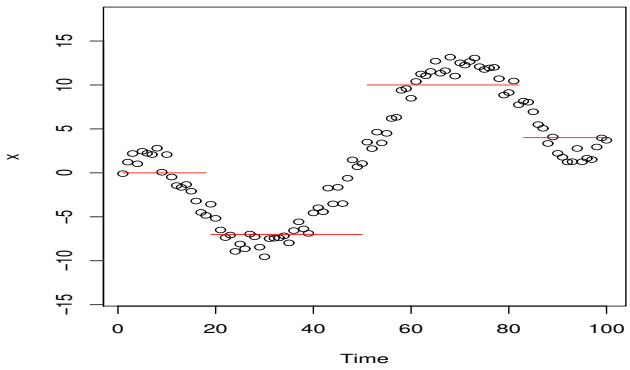

Figure 2: Time Series Approximated by a Piecewise Constant Model

return. The standard way of defining the rate of return for stock prices or stock indexes from time t to time $t+1$ is $\log X_{t+1} - \log X_t$, where X_{t+1} and X_t are the prices at the respective times.

A stylized property of rates of return is volatility clustering. Figure 3 is an illustration of this property for a small monthly sequence of the S&P 500 Index over a period from January, 2010, through November, 2015. (This short time series was just chosen arbitrarily. More data and data over other time spans may illustrate this better; but here, our emphasis is on a simple exposition. See **?** for more complete discussions of volatility clustering and other properties of financial time series.)

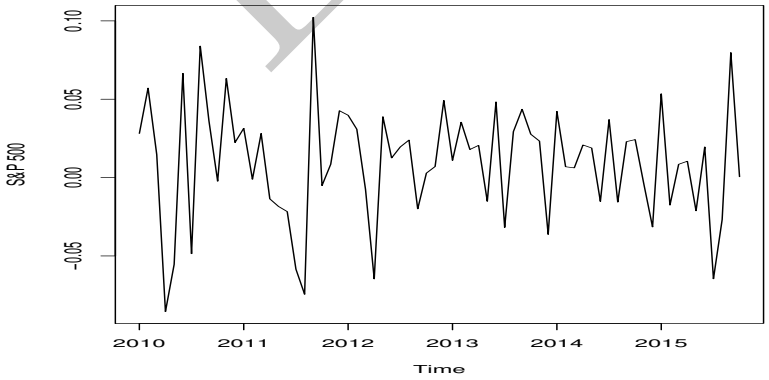

Figure 3: Monthly Log Rates of Return of S&P 500

Volatility clustering is an example of changes in a time series in which the location (mean or median) may be relatively unchanged, but the scales change from one regime to another. The changepoints in this case are points in the time series where the scales change. The derive time series, that is, the volatility time series can be approximated with a piecewise constant model.

In order to identify changes in scale or volatility, we must have some measure of volatility. It may not be obvious how to measure volatility in a time series, and this is especially true if the volatility it changing. A simple measure, called "statistical volatility" by economists, is just the sample standard deviation, which of course ignores any autocorrelations. To illustrate, however, we compute the statistical volatilities over the apparent separate regimes of the log returns shown in Figure 3. This type of analysis results in a piecewise constant time series shown in Figure 4 of the type we discussed in Section 2.1. There are various methods for detecting changepoints for scales, but we will not discuss them here.

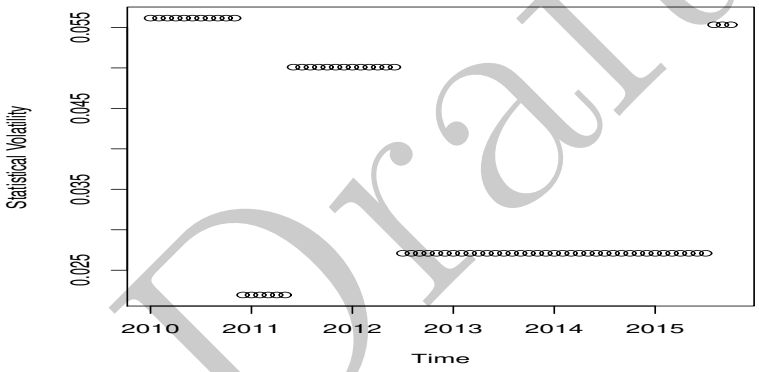

Figure 4: Monthly Statistical Volatility of Log Returns of S&P 500

2.3 Trends

The main interest in patterns in time series most often focuses on changes in trends. This is particularly true in financial time series, see, for example, **?** and **?** for methods that focus solely on trends.

More interesting simple linear models in time series are those that exhibit a "trend" either increasing or decreasing. Changepoints are the points at which the trend changes.

Identification of changepoints is one of the central aspects of technical analysis of financial data, and is the main feature of the so-called point and figure charts that have been used for many years (**?**). Point and figure charts are good

for identification of changepoints and the amount of change within an up or a down trend.

Some interesting patterns are easily seen in a point and figure chart. For example, a pattern that many technical analysts believe carries strong predictive powers is the "head-and-shoulders" pattern. Figure 5 shows the stock price for Intel Corporation (NASDAQ:INTC), and on the right side, a modified point and figure chart. (The modifications, suggested in **?**, among other things involve the definition of threshold change.) The head-and-shoulders pattern is clearly visible in both the graph of the raw prices on the left and the trend chart on the right. (This is a very strong head-and-shoulders pattern; most head-and-shoulders patterns that technical analysts would identify are not this clear.)

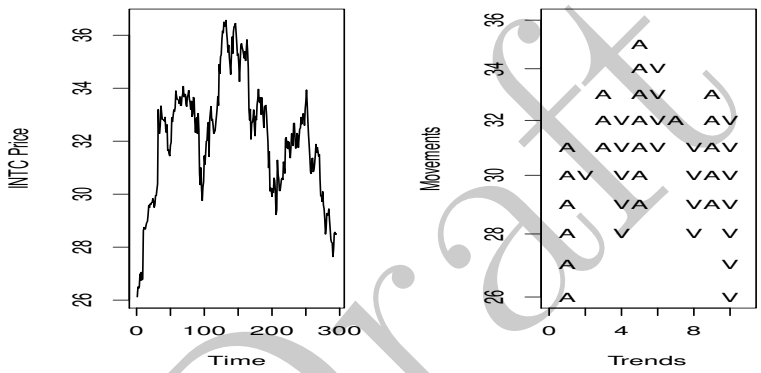

Figure 5: Intel Price, 2014-6-1, through 2015-7-31, and the Associated Trend Chart

Notice in the trend chart in Figure 5 that the time axis is transformed into an axis whose values represent only the ordered changepoints. One of the major deficiencies of point and figure charts and trend charts is that information about the length of time between changepoints is not preserved.

A very effective smoothing method is use of piecewise linear models. Piecewise linear fits are generalizations of the piecewise constant models, with the addition of a slope term. There are many variations on this type of fit, including the criterion for fitting (ordinary least squares is most common) and restrictions such as continuity (in which case the piecewise linear fit is a first degree spline). New variations on the basic criteria and restrictions are suggested often; see, for example, **?**.

Some breaks between trends are more interesting than others, depending on the extent to which the trend changes. Within a regime in which a single trend is dominant, shorter trends of different direction or of different magnitude may occur. This raises the issue of additional regimes, possibly leading to overfitting

or of leaving a regime in which many points deviate significantly from the fitted model.

3 Model Building

One of the main objectives of building a model for a time series is to reduce the amount of data by use of an approximate representation of the dataset. While the main objectives of the standard models for analysis of time series, such as ARIMA and GARCH extensions in the time domain and Fourier series or wavelets in the frequency domain, may be to understand the data-generating process better, such models also provide an approximation or smoothing of the data and thereby achieve significant data reduction. Several approximations based on simple piecewise models, each with its three-letter-acronym, have been proposed. These obviously depend on identification of changepoints prior to or in conjunction with the modeling within the individual regimes. Representation of the sequence of models then becomes an important issue. While a model is usually represented as a parametized equation, a common method of simplifying the representation further is to define a set of models, often of a common form, but each instantiated with fixed values of all parameters, and then to associate a symbol with each instantiation of each model. Two methods following this approach are symbolic aggregate approximation (SAX), see **?**, and nonparametric symbolic approximate representation (NSAR), see **?**. **?** provides a general review of various methods of smoothing time series.

Because the identification of changepoints, that is, the identification of of regimes, is intimately tied to the identification and fitting of models within the individual regimes, it is not possible to separate those two steps. Usually, a form of the model is chosen and then regimes are chosen based on the goodness of fits of potential models of that form. Often, especially in the analysis of stock prices, there is no model within the regimes other than simple increasing or decreasing trends. **?** and **?**, for example, described methods for determining changepoints between increasing and decreasing price trends.

Batch Methods

For fitting piecewise constant models, there are various straightforward ways of determining the values of the approximating constants. If all of the data are available as we mentioned above, a simple batch process is to use sample quantiles of the data that correspond to some parametric model, such as a normal distribution, and then just identify regimes as those subsequences clustering around the sample quantiles.

Another simple batch approach is to fit a single model of the specified form, and then to identify subsequences based on points that are outliers with respect to a fitted model. This process is repeated recursively on subsequences, beginning with the full sequence.

Given a single linear trend over some regime, **?** defined measures for "perceptually important points", which would be candidate changepoints. The perceptually important points are ones that deviate most (by some definition) from a trendline.

Online Methods

Batch methods, such as ones that base local models on sample quantiles of the whole time series, or those that recursively identify subsequences with local models, have limited applicability. In most applications of time series analysis, new data are continually being acquired, and so an online method is preferable to a batch method.

An online method accesses the data one observation at a time and can retain only a predetermined amount of data to use in subsequent computations.

4 Model Building: Alternating Trends Smoothing

A method of identifying changepoints in a time series based on alternating up and down linear trends, called alternating trends smoothing, or ATS, is given in Algorithm 1. It depends on a smoothing parameter, h, which specifies the step size within which to look for changepoints.

Code for simple tests of ATS

```
# Experiments
# first changepoint within step size; two changepoints that are consecutive
x<-c(6,8,7,9,8,7,6,6,4,5,3,6,9,4,3,6,5,4,3,2,7,8,9,1,3,4,6,4,3,2,5,4,3,4,5,6,7)

# slope at first step point different from direction of first extreme point.
# compare ATS1 with ATS on these data
x<-c(6,8,7,9,8,5,6,6,4,5,3,6,5,4,3,6,5,4,3,2,7,8,9,1,3,4,6,4,3,2,5,4,3,4,5,6,7)

# slope at first step point = 0.
x<-c(6,8,7,9,8,6,6,6,4,5,3,6,9,4,3,6,5,4,3,2,7,8,9,1,3,4,6,4,3,2,5,4,3,4,5,6,7)

# slope=0 and data constant after first step.
x<-c(6,8,7,9,8,6,6,6,6,6,6,6,6,6,6,6,6)

# not enough data.
x<-c(6,8,5,6)

plot(x)
ATS(x,step=5,segments=TRUE)
```

Algorithm 1 Alternating Trends Smoothing (h)

1. Set $d = 1$ (changepoint counter)
2. While (more data in first time step)
   (a) for $i = 1, 2, \ldots, m$, where $m = h$ if h additional data available or else m is last data item:
   input x_i;
   (b) set $b_d = 1$; $c_d = x_1$
   (c) determine $j_+, j_-, x_{j_+}, x_{j_-}$ such that
   $x_{j_+} = \max x_1, \ldots, x_h$ and $x_{j_-} = \min x_1, \ldots, x_h$
   (d) set $s = (x_k - x_i)/(k - i)$ and $r = \text{sign}(s)$
   (e) while $r = 0$, continue inputting more data; stop with error at end of data
3. Set $j = i$ (index of last datum in previous step); and set $d = d + 1$
4. While (more data)
   (a) for $i = j + 1, j + 2, \ldots, j + m$, where $m = h$ if h additional data available or else $j + m$ is last data item:
   input x_i;
      i. while $(\text{sign}(s) = r)$
         A. set $k = \min(i + h, n)$
         B. if $(k = i)$ break
         C. set $s = (x_k - x_j)/(k - j)$
         D. set $j = k$
      ii. determine j_+ such that rx_{j_+} is the maximum of $rx_{j+1}, \ldots, rx_{j+m}$
      iii. set $b_d = j_+$; and set $c_d = x_{j_+}$
      iv. set $d = d + 1$; set $j = j_+$; and set $r = -r$
   (b) set $b_d = j_+$; and set $c_d = x_{j_+}$

∎

The output of this algorithm applied to a time series $x_1, x_2, \ldots$ is

$$(b_1, c_1), (b_2, c_2), \ldots,$$

where $b_1 = 1$, $c_1 = x_1$, $b_2 = t^{(2)}$, and $c_1 = x_{t^{(2)}}$, where $t^{(2)}$ is the time at which the first trend changes sign.

Between two breakpoints the trend is represented by the slope of the time series values at the two points divided by the time between the two points, and the smoothed time series is the piecewise linear trendlines that connect the values at the changepoints. The method is effective for finding interesting patterns. For example, the head-and-shoulders pattern in the Intel stock price, shown in Figure 5, is very apparent in the ATS representation of the time series shown in Figure 6. A step size of 30 was used in that fit.

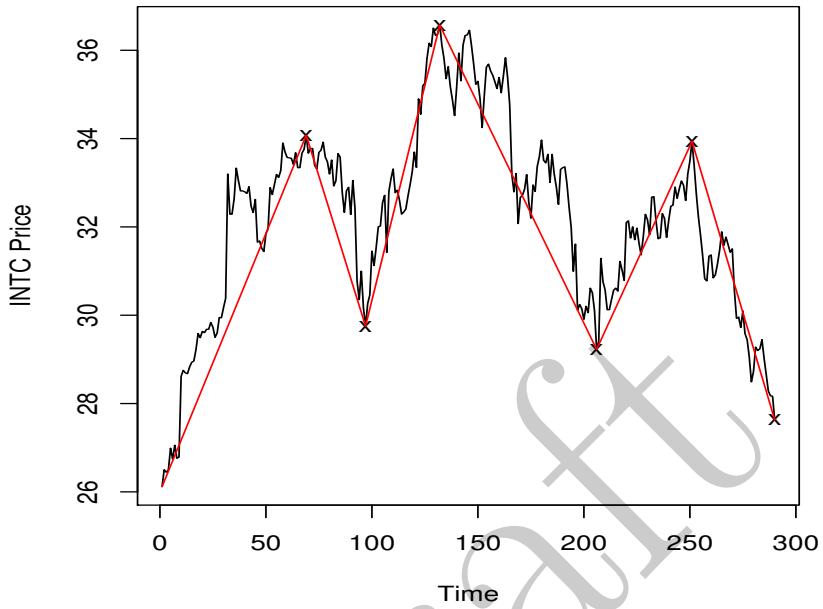

Figure 6: A Head-and-Shoulders Pattern in ATS

The output of the ATS algorithm applied to the INTC data in Figure 6 is

$$(1, 26.11), (69, 34.07), (97, 29.75), (132, 36.57), (206, 29.23), (251, 33.94), (290, 27.63).$$

Thus, the 290 raw data points are summarized in the 7 pairs of numbers representing the changepoints and their values.

While the ATS fit emphasizes only the signs of the trends, the actual slopes are very easily computed from the values at the changepoints.

The Tuning Parameter

The tuning parameter h in Algorithm 1 is a "step size". The process of identifying the next changepoint begins with the datum one step size beyond the current changepoint. Larger values of h tend to increase the distances between changepoints, but the actual distance between changepoints can be smaller than h; in fact, the distance between changepoints can be as small as 1 time unit.

Although in the standard implementation, the identification of trends in ATS is based on individual points, the aggregate behavior tends to dominate, especially after the first trend. In identifying the changepoint at the end of the

first regime, however, the dependence of the trend on the point one step size beyond can lead to misidentification of the trend. This is because ATS works by identifying changepoints based on changing trends, and in the first trend there is no previous trend for comparison. This can result in a trend determined by points x_1 and x_h differing completely from the apparent trend in the points $x_1, \ldots, x_{h-1}$. For this reason, a simple modification of the ATS algorithm in the first step is to use some other criterion for determining the trend. One simple approach is a least-squares fit of a line through x_1 and that comes close to the points $x_2, \ldots, x_h$. Because of the constraint that the line goes through x_1, the least-squares criterion might be weighted by the leverages of the other points. If the time-spacing is assumed to be equal over the points $x_1, \ldots, x_h$, the least-squares slope is

$$\arg\min_s \sum_{i=2}^{h} (x_i - si)^2/i = (h^2 - h)/\sum_{i=2}^{h} x_i. \quad (1)$$

If all of the first h observations follow the same trend, the modification has no effect.

This modification can also be at each step, and it often results in what appears visually to be a better fit. Nevertheless, it is not always easy to pick a good rule for determining the direction of a trend. Because of the way the algorithm looks backwards after detecting a change in the sign of the trend, the modification does not have as much effect in subsequent regimes after the first one.

If the length of the time series is known in advance, a step size equal to about one tenth of the total length seems to work reasonably well. Even so, it is often worthwhile to experiment with different step sizes.

Figure 7 shows ATS applied to the daily closing price of the stock of International Business Machines Corporation (NYSE:IBM) from January 1, 1970, through December 31, 2014. There are $n = 11,355$ points. Over this full period, a step size of $h = 1136$ (a step size of $n/10$) was used. This resulted in the alternating trend lines shown as solid red line segments.

Code to produce Figures 7 and 8

```
setwd('c:/Work/Papers_and_Talks/Working/SpringerHandbookBigData/')
source("c:/files/Computers_and_Software/Notes_ExamplePrograms/R_finance/financetools_funs.R
source("c:/files/Computers_and_Software/Notes_ExamplePrograms/R_TimeSeriesPatterns/ATS.R")

IBMd<-get.stock.price("IBM", start.date=c(1,1,1970), stop.date=c(12,31,2014),
                              freq="d",full.table=FALSE, print.info=FALSE)
plot(as.ts(IBMd),ylab="IBM Price")
ATS(IBMd,step=0,segments=TRUE)
ATS(IBMd[5816:length(IBMd)],step=0,segments=TRUE,offset=5815,color="blue",ltype=3,char="*")
savePlot("FPTS_35",type="eps")
```

```
plot(as.ts(IBMd),ylab="IBM Price")
chpt1<-ATS(IBMd,step=100,segments=TRUE)
chpt2<-ATS(chpt1[,2],step=5,segments=FALSE)
lines(chpt1[chpt2[,1],1],chpt1[chpt2[,1],2],col="blue",lty=3)
savePlot("FPTS_40",type="eps")
```

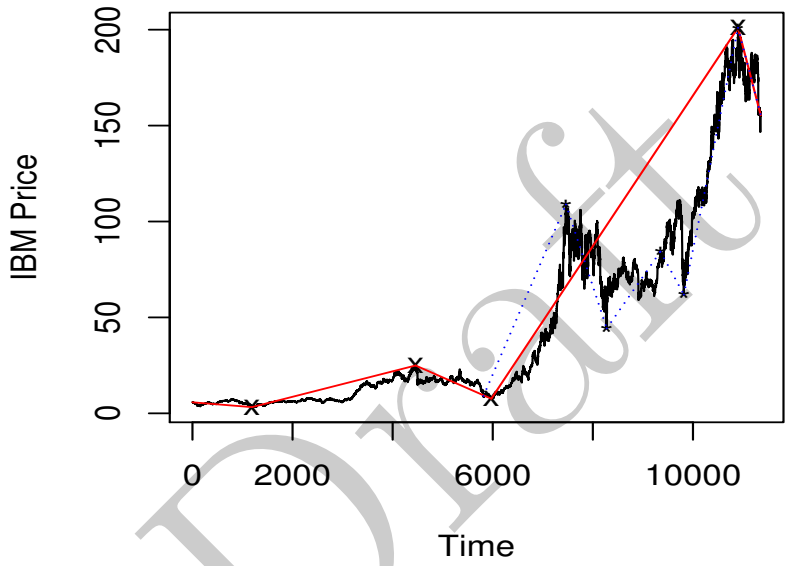

Figure 7: ATS with Different Stepsizes over Different Regions

There are only 5 changepoints identified in the IBM daily closing prices, and the changes are as likely to occur in the area of "low action" (the earlier times) as in the areas of higher volatility. This is because ATS operates in an online fashion; when processing the data in the earlier time regimes, it is not known that the trends will become more interesting. Only one changepoint from the observation at the time index of 5971 through the end of the series is identified. A smaller step size may be appropriate. Alternating trend segments were then determined for these data, beginning with January 1, 1993, (a time index of 5815 in the original series), and using a stepsize of 554. These are shown as dashed blue line segments in Figure 7. The ATS fit resulting from those two different stepsizes better captures the pattern of the time series.

Alternating trends smoothing can be applied recursively. Once the original data are smoothed, ATS can be applied to the changepoints determined in the

original smoothing. This is illustrated in Figure 8 using the same IBM daily closing price data as before.

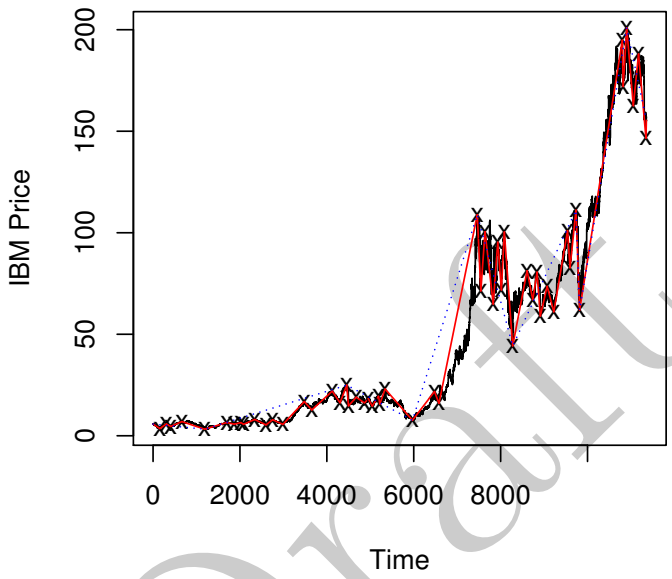

Figure 8: ATS Applied Twice

First, a step size of 100 was used. This resulted in 53 changepoints. These are shown in Figure 8, and the trend lines connecting them are shown as solid red line segments. Next, ATS was applied to the changepoints (55 points in all, including the first and last observations). A stepsize of 5 was used for this smoothing. This resulted in 9 changepoints. The alternating trend lines connecting them are shown as dashed blue line segments in Figure 8. The original set of 54 trendline segments may be considered too noisy to be a good overall model. That model may be considered to be overfit. The subsequent fit of the changepoints is of course much smoother than the fit of the original data.

This repeated ATS fitting is iterative data reduction. The first fit, reduced the data to 55 points (including the two endpoints). These points may contain sufficient information to summarize the original 11,355 points. Carrying the reduction further by applying ATS to the changepoints, we reduce the data to 11 points (again, including the two endpoints), and this may be sufficient for our purposes.

Making transformations to a time series before applying ATS results in dif-

ferent changepoints being applied. For example, using a log transformation of the IBM price data in Figure 7 would result in different changepoints, and even using only one stepsize for the whole time series, those changepoints on the log data may be more meaningful.

Modifications and Extensions

An alternating trends fit can be modified in several ways. One very simple way is to subdivide the regimes by identifying changepoints within any regime based on deviations from the trendline within that regime. This type of procedure for identifying changepoints has been suggested previously in a more general setting.

Consider again the IBM data in Figure 7, with the ATS fit using a stepsize of 1136. (This is the solid red line in the figure.) Over the region from a time of 5971 to a time of 10901, a single model was fit. This model is the line segment from the point $(5971, 7.63)$ to the point $(10901, 200.98)$. The most deviant point within this regime, as measured by the vertical residuals is at time point 9818 where the actual price is 61.90, while the point on the trendline is 158.51.

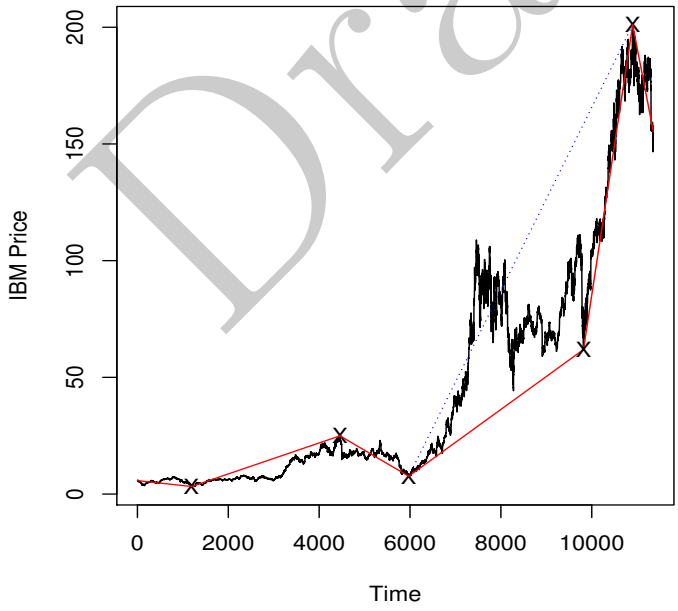

Figure 9: A Linear Trend Broken into Two Trends at the Most Extreme Point

This point can be considered to be a changepoint, as shown in Figure 9. In this case the trends on either side of the changepoint are both positive; that is, they are not alternating trends.

Continuing to consider changepoints within regimes identified with a single trend, in this case we would likely identify a changepoint at about time 7500, which would correspond to the maximum deviation from the single trendline between the original changepoint at time 5971 and the newly identified changepoint at time 9818. In this case, while the single trend is positive, the first new trend would be positive and the second new trend would be negative.

While in many applications in finance, only the sign of a trend is of primary interest, occasionally, the magnitude may also be of interest, and a changepoint might be identified as a point at which the slope of the trend changes significantly. The basic ideas of ATS can be adapted to this more general definition of changepoints; however, some of the simplicity of the computations of ATS would be lost. This approach would also require an additional tuning parameter to quantify "significance" of change in trend, when the sign of the trend may not change.

Another possible improvement to the basic ATS algorithm is to allow the stepsize to be adjusted within the computations. The alterations to the stepsizes could be based on the number of changepoints or on goodness-of-fit within a regime, and in either of these general approaches, there are several possible ways of doing it. The modification shown in Figure 9 could be performed routinely as a postprocessing step in any ATS fit. That modification of course would require an additional tuning parameter to be used in deciding whether or not to break up an existing regime.

In very noisy data goodness-of-fit measures can often be misleading. This is because single or a few outliers can cause the measure to indicate an overall lack of fit. (Note that the basic ATS fitting, although individual points are used in determining changepoints, the method is generally resistant to outliers.) As in most data analysis, outliers often must be treated in ad hoc ways. This is because they often contain completely new information.

5 Bounding Lines

In statistical modeling it is common practice to associate "confidence bounds" with a fitted model. These are usually based on some underlying probability distribution, and they can take various forms depending on the model, which includes the relevant probability distributions.

In our objective of finding patterns in the data, we have not assumed any specific probability model. In other applications of trend analysis, it is common to identify bounding lines within a given regime that generally are in the same directions as the trend over that region, but which are either above all the points in the regime (a "resistance line") or below all the points (a "support line"). Such bounding lines do not depend on any probability model.

As we have emphasized, finding the changepoints is the paramount prob-

lem. Once the regimes are identified, however, it may be of interest to identify bounding lines for those regimes.

For a given time series $\{x_t : t = 1, \ldots, n\}$, we seek a lower bounding line $x = a + bt$ that satisfies the optimization problem

$$\begin{array}{l} \min_{a,b} \sum_{t=1}^{n} \rho(x_t - a + bt) \\ \text{s.t. } x_t \geq a + bt, \quad \text{for } t = 1, \ldots, n, \end{array} \tag{2}$$

where $\rho(\cdot)$ is a nonnegative function such that for a vector $v = (v_1, \ldots, v_n)$, $\|v\| = \sum_{t=1}^{n} \rho(v_t)$ is a norm. An upper bounding line is defined the same way, except that the inequality in the constraint is reversed.

In general, this is a hard optimization problem, but for the L_1 norm, that is, when $\rho(\cdot) = |\cdot|$, it is straightforward; and a method is given in Algorithm 2. The method depends on the fact that the L_1 norm satisfies the triangular inequality with equality: that is, $\|y + z\|_1 = \|y\|_1 + \|z\|_1$. The method also depends on the fact that the data are equally spaced along the time axis.

Algorithm 2 L_1 **Lower Bounding Line:** $x_t = \tilde{a} + \tilde{b}t$

1. Fit the data $x_t = a + bt$ by a minimum L_1 criterion to obtain parameters a_* and b_*.
2. Determine the position of the minimum residual, k, and adjust a_*: $a_* \leftarrow x_k - b_* k$.
3. If $k - 1 = n - k$, then set $\tilde{b} = b_*$ and $\tilde{a} = a_*$, and stop.
4. Else if $k \leq n/2$,
   (a) rotate the line $x = a_* + b_* t$ clockwise about the point (k, x_k) until for some point (i, x_i), $x_i = a_* + b_* i$.
   (b) set $\tilde{b} = (x_i - x_k)/(i - k)$ and $\tilde{a} = x_k - \tilde{b}k$, and stop.
5. Else if $k > n/2$,
   (a) rotate the line $x = a_* + b_* t$ counterclockwise about the point (k, x_k) until for some point (j, x_j), $x_j = a_* + b_* j$.
   (b) set $\tilde{b} = (x_k - x_j)/(k - j)$ and $\tilde{a} = x_k - \tilde{b}k$, and stop. ∎

Theorem 1 *Algorithm 2 yields a solution to optimization problem (2), when* $\rho(\cdot) = |\cdot|$.

Proof. Let a_* and b_* be such that $\sum_{t=1}^{n} |x_t - a_* - b_* t| = \min_{a,b} \sum_{t=1}^{n} |x_t - a - bt|$. Let k be such that

$$x_k = \arg\min_{x_i} (x_i - a_* - b_* i).$$

For the optimal values of $\tilde{a}$ and $\tilde{b}$, we must have $x_k \geq \tilde{a} + \tilde{b}k$.

There are three cases to consider: $k = (n+1)/2$ (corresponding to step 3 in Algorithm 2), $k \leq n/2$ (corresponding to step 4 in the algorithm), and $k > n/2$ (corresponding to step 5 in the algorithm). We will first consider the case

$k \leq n/2$. The case $k > n/2$ follows the same argument. Also following that argument, it will be seen that the solution given for case $k = (n+1)/2$ is optimal.

If $k \leq n/2$, consider the line $x = x_k - b_*k + b_*t$. (The intercept here is what is given in step 2 of the algorithm.) This line goes through (k, x_k), and also satisfies the constraint $x_t \geq x_k - b_*k + b_*t$, for $t = 1, \ldots, n$. The residuals with respect to this line are $x_t - x_k + b_*k - b_*t$ and all residuals are positive. Any point t for $t < k$ is balanced by a point $\tilde{t} > k$, and there are an additional $n - 2k$ points $x_{\tilde{t}}$ with $\tilde{t} > k$. If any point $x_{\tilde{t}}$ with $\tilde{t} > k$ lies on the line, that is, $x_{\tilde{t}} = x_k - b_*k + b_*\tilde{t}$, then this line satisfies the optimization problem (2) because any change in either the intercept $x_k - b_*k$ or the slope b_* would either violate the constraints or would change the residuals in a way that would increase the norm of the residuals. In this case the solution is as given at the end of step 4, because $\tilde{b} = b_*$.

The step now is to rotate the line in a clockwise direction, which results in an increase in any residuals indexed by t for $t < k$ and a decrease of the same amount in the same number of residuals $x_{\tilde{t}}$ with $\tilde{t} > k$. (This number may be 0.) It is the decrease in the residuals of the additional points indexed by $\tilde{t} > k$ that allows for a possible reduction in the residual norm (and there is a positive number of such points).

Consider the line $x = x_k - b_*k + (b_* + \delta_b)t$ such that δ_b is the minimum value such that there is a point $x_{\tilde{t}}$ with $\tilde{t} > k$ such that $x_{\tilde{t}} = x_k - b_*k + (b_* + \delta_b)\tilde{t}$. This line satisfies the constraints and by the argument above, is optimal. Hence, the solution is as given at the end of step 4, because $\tilde{b} = b_* + \delta_b$.

Now consider the case $k = (n+1)/2$. (In this case, n is an odd integer, and $k - 1 = n - k$ as in Algorithm 2.) Following the same argument as above, we cannot change the intercept or the slope because doing so would either violate the constraints or would change the residuals in a way that would increase the norm of the residuals. Hence, we have $\tilde{b} = b_*$ and $\tilde{a} = x_k - b_*k$ as given in the algorithm is a solution to optimization problem (2) when $\rho(\cdot) = |\cdot|$. ∎

One possible concern in this method is that the L_1 fit may be nonunique. This does not change any of the above arguments about an optimal solution to optimization problem (2) when $\rho(\cdot) = |\cdot|$. It is possible that the solution to this optimization problem is nonunique, and that is the case independently of whether or not the initial fit in Algorithm 2 is nonunique.

The discussion above was for lower bounding lines under an L_1 criterion. Upper bounding lines are determined in the same way following a reversal of the signs on the residuals.

Bounding lines can easily be drawn over any region of a univariate time series. They may be more meaningful if separate ones are drawn over separate regimes of a time series, as in Figure 10, where separate bounding lines are shown for the six regimes corresponding to alternating trends, that were shown in Figure 6.

Code to produce Figure 10

```
setwd('c:/Work/Papers_and_Talks/Working/SpringerHandbookBigData/')
```

```
source("c:/files/Computers_and_Software/Notes_ExamplePrograms/R_finance/financetools_funs.R
source("c:/files/Computers_and_Software/Notes_ExamplePrograms/R_TimeSeriesPatterns/ATS.R")
source("c:/files/Computers_and_Software/Notes_ExamplePrograms/R_TimeSeriesPatterns/Bounding

INTCd<-get.stock.price("INTC", start.date=c(6,1,2014), stop.date=c(7,24,2015),
                               freq="d",full.table=FALSE, print.info=FALSE)
plot(as.ts(INTCd),ylab="INTC Price")
brks <- ATS(INTCd,segments=TRUE)
k <- dim(brks)[1]-1
for (i in 1:k){
    env <- 1
    xs <- BoundingLines(INTCd[brks[i]:brks[i+1]],env,segments=TRUE,offset=brks[i])
    env <- -1
    xs <- BoundingLines(INTCd[brks[i]:brks[i+1]],env,segments=TRUE,offset=brks[i])
}

savePlot("FPTS_47",type="eps")
```

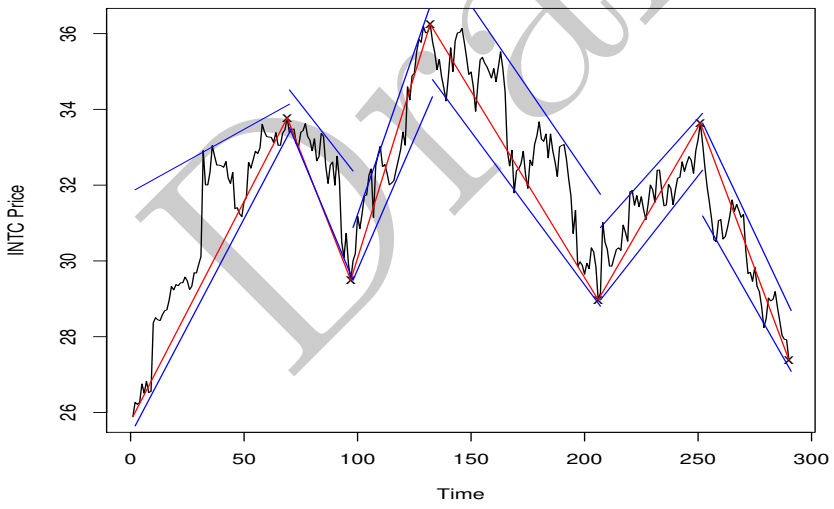

Figure 10: Bounding Lines in Six Regimes

6 Patterns

One of our motivations for fitting time series is for clustering and classification of time series based on similarities of Usually, in large time series datasets a

single model does not fit well, so our approach has been to identify a sequence of local models, $\mu_1(t), \mu_2(t), \ldots, \mu_k(t)$, in k regimes.

The models $\mu_i(t)$ may be of various forms, and they may contain various levels of information. For example, in piecewise constant modeling, the form of the model is

$$\mu_i(t) = c_i \mathrm{I}_{[t_{i1}, t_{i2}]}(t), \tag{3}$$

where $\mathrm{I}_S(t)$, is the indicator function: $\mathrm{I}_S(t) = 1$ if $t \in S$, and $\mathrm{I}_S(t) = 0$ otherwise. (This formulation allows the global model to be written as $\sum_i \mu_i(t)$.) The form of the model in ATS is

$$\mu_i(t) = (a_i + s_i t)\mathrm{I}_{[t_{i1}, t_{i2}]}(t), \tag{4}$$

where in the notation of Algorithm 1, $a_i = c_i$, $s_i = (c_{i+1} - c_i)/(b_{i+1} - b_i)$, $t_{i1} = c_i$ and $t_{i2} = c_{i+1}$, and of course a global is just the sum of these local models.

For comparing different time series for clustering or classification, we may focus on patterns of models, $\mu_i(t), \mu_{i+1}(t), \ldots, \mu_{i+r}(t)$, on r successive regimes, not necessarily beginning at the start of the time series and not necessarily extending over the full extent of the time series. We compare the pattern $P_{ri} = (\mu_i(t), \mu_{i+1}(t), \ldots, \mu_{i+r}(t))$ with patterns from other time series. The obvious basis for comparison would be a metric, or a measure with some of the properties of a metric, applied to the patterns; that is, we define a a metric function $\rho(P_{ri}, P_{sj})$, where P_{ri} is a pattern consisting of a sequence $\mu_i(t), \mu_{i+1}(t), \ldots, \mu_{i+r}(t)$ and P_{sj} is a pattern consisting of a sequence over s regimes beginning at the j^{th} one.

Because the patterns depend on fitted models, the fact that a pattern

$$(\mu_i(t), \quad \mu_{i+1}(t), \quad \mu_{i+2}(t))$$

in one time series is exactly the same as a pattern

$$(\mu_j(t), \quad \mu_{j+1}(t), \quad \mu_{j+2}(t))$$

in another time series does not mean that the actual values in the two time series are the same over those regimes or even that the values have some strong association, such as positive correlation, with each other. This is generally not inconsistent with our objectives in seeking patterns in time series or in using those patterns in clustering and classification.

In this section, we discuss some of the issues in clustering and classification of time series, once a sequence of regimes is identified. An important consideration in analyzing multiple time series is registration of the different time series; that is, shifting and scaling the time series so that the regimes in the separate time series can be compared. We also briefly indicate possible approaches for further data reduction. Once these issues have been addressed, the problems of clustering and classification are similar to those in other areas of application. Examples and the details of the use of the clustering and classification methods are discussed by **?**.

6.1 Time Scaling and Junk

Prior to comparing two time series or the patterns in the two series, we generally need to do some registration of the data. Usually, only subsequences of time series are to be compared, and so the beginning and ending points in the two series must be identified. The actual time associated with the subsequences may be different, and the subsequences may be of different lengths. There are various methods of registering two subsequences. The most common methods are variants of "dynamic time warping" (DTW), which is a technique that has been around for many years. There are several software libraries for performing DTW.

In the overall task of identifying and comparing patterns in time series, the registration step, whether by DTW or some other method, can be performed first or later in the process. Our preference generally is to identify breakpoints prior to registration.

Similarity of patterns is not an absolute or essential condition. Similarity, or dissimilarity, depends on our definition of similarity, which in turn depends on our purposes. We may wish to consider two patterns to be similar even in the time intervals of the piecewise models do not match. We also may wish to ignore some models within a pattern, especially models of brief duration.

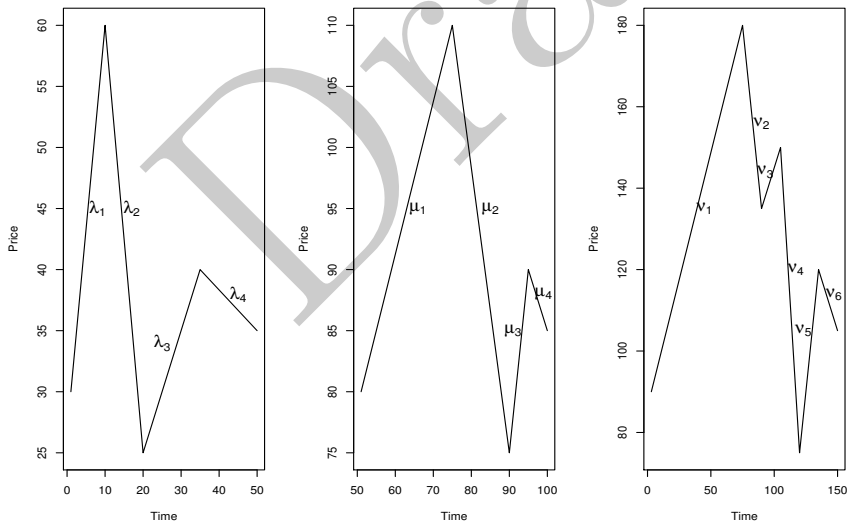

Figure 11: Three Patterns that Are Similar

In Figure 11 we see three patterns that for some purposes we would wish to consider to be similar to each other. The times as well as the actual values are rather different, however. Among the three time series, the times are both

shifted (the starting time of course is almost always arbitrary) and scaled. The unit of time is not always entirely arbitrary. It depends on our ability to sample at different frequencies, and the sampling rate is not always adjustable. The unit of time may also be important in an entirely different way. It is a well-recognized property of markets that frequency of trading (or frequency of recording data) results in different market structures (see, for example, **?**).

Figure 11 also illustrates another problem for comparing patterns. The blip in the time series on the right side results in two additional model terms in the fitted time series. We would actually like to compare the patterns

$$(\mu_1, \mu_2, \mu_3, \mu_4)$$

and

$$(\nu_1, \mathcal{S}(\nu_2, \nu_3, \nu_4), \nu_5, \nu_6),$$

where the specific correspondences are $\mu_1 \approx \nu_1$, $\mu_3 \approx \nu_5$, $\mu_4 \approx \nu_6$, and $\mu_2 \approx \mathcal{S}(\nu_2, \nu_3, \nu_4)$, where $\mathcal{S}(\nu_2, \nu_3, \nu_4)$ is some smooth of the three models ν_2, ν_3, and ν_4.

The plot on the right side of Figure 11 compared with the other two plots illustrates the intimate connection between smoothing or model fitting and pattern recognition. A smoother fit of the time series shown on the right side would have resulted in just four models (three changepoints), in which the second model would be some smooth of ν_2, ν_3, and ν_4, that is, $\mathcal{S}(\nu_2, \nu_3, \nu_4)$.

The extra blip in the time series on the right side of Figure 11 is "junk" at a lower level of resolution.

6.2 Further Data Reduction: Symbolic Representation

While the individual components of a pattern P_{ri} may contain various details of the models, in some cases some details can be suppressed while salient features of the pattern, that is, the *sequence*, are retained. For example, for a sequence of constant models such as in equation (3), the most important features of the sequence may be a sequence of indicators whether the c_i were small, mid-size, or large; that is, the pattern is a sequence of the form *abcde*..., where each of the *a*s, *b*s, and so on, are just the values `s`, `m`, and `l`, indicating "small", "medium", and "large". For example, the pattern `sllmls` would indicate a sequence of six piecewise constant models of the form of equation (3), in which c_1 is (relatively) small, c_2 and c_2 are large, and so on. This representation, of course, does not include information about the t_{i1}s or t_{i2}s or even the exact values of the c_is, but the *patterns* of small, medium, and large may be information for clustering or classifying time series.

This further step of data reduction of forming categories of models and associating each category with a single symbol can be very useful in data mining of time series, and has been used in various ways for some time. One widely-used symbolic approximation for time series is SAX, which is based on a modification of a sequence of piecewise constant models (called PAA), see **?**. Another

type of symbolic approximation of a time series, called NSAR for nonparametric symbolic approximate representation, was described by **?**. This method is based on preliminary wavelet transformations and so enjoys the multi-resolution properties of wavelets.

The transformation of models with quantitative parameters to models of categorical symbols requires some a priori definition of the symbols, possibly a complete listing or catalogue of the symbols, or at least some formula for defining new symbols. In batch processing of the data, the range of possible models can be determined before the transformation of quantitative models to categorical models.

In ATS, each model is characterized by four real numbers. A sequence of r models is characterized by $2r + 2$ real numbers. To reduce the data further, the real numbers are binned into ordered groups. These ordered bins can be associated with a unique set of symbols.

The replacement of models with continuous numeric parametrizations by symbolic representations results in loss of data. A linear model in a given regime may be transformed into a model that carries only the information that a particular coefficient is large, relative to the same coefficient in other regimes. The symbolic approximations may even lose information concerning the time of the changepoints.

6.3 Symbolic Trend Patterns (STP)

The symbolic approximation of SAX is based on a type of piecewise constant modeling called "piecewise aggregate approximation" ("PAA") as described by **?**. The same idea of SAX can be applied to the models in ATS, as described by **?** who called it "symbolic trend patterns", or "STP". These symbols consist of pairs of symbols or syllables. They are formed by selection of a consonant

J, K, L, M, N

that represents duration of an upward trend, or of a consonant

P, Q, R, S, T

that represents duration of an downward trend, and selection of a vowel

A, E, I, O, U

that represents magnitude of a trend. In many cases, however, the vowels could represent the magnitude of the change in value instead of the magnitude of a trend, that is the slope of the segment between the changepoints.

In each case, the individual letters as listed above represent increasing magnitudes. Thus, "P" represents a downward trend of short duration, and "A" represents a trend (up or down) of very small magnitude.

If these symbols are defined and assigned in a batch fashion, they can represent quantiles (or approximate quantiles) of the fully observed time series.

For example, the ATS fit of the INTC data in the head-and-shoulders pattern shown in Figure 6 could be represented by the STP symbolic approximation

LO, PE, KO, RO, JE, QI

where the vowel is used to represent magnitude of change, rather than rate of change. The ATS fit of the IBM price data shown by the solid red lines in

Figure 7 could be represented by the symbolic approximation

PA, LE, QE, NU

where again the vowel represents magnitude of change.

Of course, because the trends in ATS alternate, if a single direction is given, then there would be no need for different symbols to be used to designate up and down moves.

6.4 Patterns in Bounding Lines

Following any kind of data reduction, there are enhanced opportunities for identifying patterns. Trends are a simple form of data reduction any offer various methods of pattern identification. Likewise, the bounding lines discussed in Section 5, may be used to develop patterns. The bounding lines have the same kinds of characteristics as the trend lines of Section 4; they have slopes and duration. When bounding lines are determined in regions determined by the ATS their slopes will generally (but not necessarily) have the same sign as the slopes of the trends.

Another interesting characteristic of bounding lines is their relationship to each other; in particular, whether they seem to be converging or diverging. (By their definition, they can never cross within the region for which they are defined.) In Figure 10, for example, we see that the bounding lines in the leftmost regime seem to be converging, while those in the second regime from the left seem to be diverging. Technical security analysts sometimes attach meaning to such patterns.

6.5 Clustering and Classification of Time Series

In clustering and classification of data we need appropriate measures of similarity or dissimilarity. The most useful measures of dissimilarity are metrics, because of their uniqueness and the ordering they induce, and the most useful metrics on $\mathbb{R}^d$ are those induced by a norm. For a given class of patterns, whether defined on a finite set of symbols or on $\mathbb{R}^d$, there is a wide choice of possible metrics. For metrics induced by norms the equivalence relation among any set of norms yields an equivalence of metrics. This equivalence carries over to metrics on a set of ordered bins or symbols (see **?**).

The problem of clustering or classification on a set of time series is essentially the problem of clustering or classification on a set of patterns. Despite the equivalence of metrics, on a given class of patterns, different metrics can lead to different clusters or different classifiers.

The most challenging problem in clustering and classification of time series arises from the time scaling and "junk" models that constitute a pattern. The three similar time series shown in Figure 11, for example, may be associated with the STP approximations

JO, PU, ME, RA
MO, RU, KE, PA
MO, PI, JA, RI, KE, PA

The question here is how to recognize the similarity of the patterns. These patterns exhibit different time scalings and in one case includes a superfluous model. An approach to the problem at this point is the same approach that was used from the start: further discretization; that is, further data reduction, with its concomitant loss of information.

One way of dealing with the time scale is a further discretization; instead of ten different values, we may just use two, up or down. The first two patterns are now the same:

$$+O, -U, +E, -A$$

A model with both short duration and small change in magnitude is a candidate for a superfluous modes; that is, one that can be smoothed away by combinations with nearby models. Applying this approach to the smoothed time series on the right side of Figure 11 would result in the second through fourth models being combined into a single model, which would be represented as RU or -U.

This approach to the problem involves combinations and adjustments of any or all of the models in a set of patterns, and so is obviously not entirely satisfactory. For clustering and classification, of course, we do not need for the patterns to be exactly alike, so another approach would be based on use of appropriate metric.

A metric that weights differences in direction of a trend much more heavily than differences in length of two trends in the same direction would achieve some of the same effect as considering the duration to be a binary variable.

Classification of time series is closing related to the standard problem of prediction or forecasting in time series. For a given pattern, the predicted value is merely the predicted class of the pattern.

A R Codes

A.1 ATS

```
#<pre>
## Function to determine changepoints in linear trends of sequenced univariate data.
## Except fot the last trend, each linear trend is followed by a trend with
## a slope of opposite sign.
## Optionally, the function will add trend-line segments to an existing plot.
##
## Arguments
##    xdata     Data; either a one-dimensional numeric array or a matrix of
##              class xts. Data must not contain missing values.
##    step      The window size within which to identify trends;
##              a nonnegative integer.
##              If step is input as 0 or if step is not specified,
##              a default value of length(x)/10 is assigned.
##              Small values result in more and shorter trends.
## Optional printing arguments; these arguments affect only the printing.
##    segments  Logical variable indicating whether to add trend-line segments
##              to an existing plot of a univariate data vector against its index.
##              ** If segments=TRUE, there must be an existing plot over the
##              appropriate range.
##              ** If segments=FALSE, the additional arguments are not used.
##    offset    If segments=TRUE, the index of the original data plotted at which
##              to begin plotting of trend lines for the current series x.
##              The index of x is treated as starting at offset+1 with respect
##              to the index of time used in the original plot.
##    ltype     If segments=TRUE, the line type, using the standard values in R.
##    color     If segments=TRUE, the line color, using the standard values in R.
##    char      If segments=TRUE, the character to print at the changepoints.
##
## Value        A matrix with two columns.
##              The first column is the index of the changepoint (with no offset).
##              The second column is the value of x at the changepoint.
##              The (1,1) element is always 1.
##              The (1,2) element is x[1].
##
ATS <- function(xdata,step=0,segments=FALSE,offset=0,ltype=1,color="red",char="x"){
##    JEG 2015-12-28
##    JEG 2016-2-12
##    JEG 2017-6-12
##    JEG 2017-6-20
    if (class(xdata)=="xts"||class(xdata)=="zoo")  {
      x <-as.numeric(xdata[,1])
    }  else x <- xdata
    n <- length(x)
    error <- FALSE
    if (sum(is.na(x)>0)) {
      print("Error: The data contain one or more missing values.")
      error <- TRUE
    }
    if (n<2) {
      print("Error: The length of data is less than 2.")
      error <- TRUE
    }
    if (step<0) {
```

```
54        print("Error: The step size is negative.")
55        error <- TRUE
56      }
57      if (step == 0) step <- max(1,round(n/10))
58      if (n<step) {
59        print("Error: The length of data is less than the step size.")
60        error <- TRUE
61      }
62      if(error)return()
63      chpts <- matrix(c(1,x[1]),ncol=2)
64  #Set initial starting point and slope (use least squares through first point)
65      sp <- 1
66      x0 <- sp
67      b <- 6*(sum((1:step)*(x[2:(step+1)]))-x[1]*step*(step+1)/2)/
68            (step*(step+1)*(2*step+1))
69      ep <- sp + step
70      diff <- x[ep]-x[sp]
71      while (sign(diff)!=sign(b)){
72        ep <- ep-1
73           if (ep <= 1) {
74            print("data constant in first step")
75            return()
76           }
77           diff <- x[ep]-x[sp]
78      }
79      while (diff==0){
80           ep <- ep+1
81           if (ep >= n) {
82            print("slope=0 and data constant after first step")
83            return()
84           }
85           diff <- x[ep]-x[sp]
86      }
87      slope <- (x[ep] - x[sp])/step
88      cs <- sign(slope)
89      while (ep < n){
90           spstart <- sp
91           while (sign(slope) == cs) {
92                   ep <- min(sp+step, n)
93                   if (sp==ep) break
94                   diff <- x[ep] - x[sp]
95                   while (diff==0 & ep>sp+1){
96                     ep <- ep-1
97                     diff <- x[ep] - x[sp]
98                   }
99                   if (diff==0){
100                    ep <- min(sp+step, n)
101                      while (diff==0 & ep<n){
102                        ep <- ep+1
103                        diff <- x[ep] - x[sp]
104                      }
105                  }
106                  slope <- diff/(ep-sp)
107                  sp <- ep
108          }
109          sp <- spstart-1+rev(which(cs*x[spstart:sp]==max(cs*x[spstart:sp])))
                  [1]
110          if(segments){
111              text(sp+offset,x[sp],char)
112              lines(c(x0+offset,sp+offset),c(x[x0],x[sp]),col = color,lty=
                      ltype)
113          }
114          chpts <- rbind(chpts,c(sp,x[sp]))
115          x0 <- sp
116          cs <- -cs
117       }
118       if(segments)lines(c(x0+offset,ep+offset),c(x[x0],x[ep]),col = color,lty=
               ltype)
```

```
119      if (chpts[dim(chpts)[1],1]!=n) chpts <- rbind(chpts,c(n,x[n]))
120      return(chpts)
121  }
122
123  #</pre>
```

c:/files/Computers_and_Software/Notes_ExamplePrograms/R_TimeSeriesPatterns/ATS.R

```
 1  #<pre>
 2  ## Function to compute an upper or lower bounding line
 3  ## for sequenced univariate data.
 4  ## The bounding line is either above or below all of the data.
 5  ## Optionally, the function will add a bounding line segment to
 6  ## an existing plot.
 7  ##
 8  ## Arguments
 9  ##    x          Data; a one-dimensional array.
10  ##    env        Indicator of whether bounding line is to be above or below.
11  ##               If env = 1, the bounding line is above ("resistance").
12  ##               If env = -1, the bounding line is below ("support").
13  ## Optional printing arguments; these arguments affect only the printing.
14  ##    segments   Logical variable indicating whether to add a bounding line
15  ##               segment to an existing plot of a univariate data vector
16  ##               against its index.
17  ##               ** If segments=TRUE, there must be an existing plot over the
18  ##               appropriate range.
19  ##               ** If segments=FALSE, the additional arguments are not used.
20  ##    offset     If segments=TRUE, the index of the original data plotted at
21  ##               which to begin plotting of trend lines for the current series
       x.
22  ##               The index of x is treated as starting at offset+1 with
     respect
23  ##               to the index of time used in the original plot.
24  ##    ltype      If segments=TRUE, the line type, using the standard values in
       R.
25  ##    color      If segments=TRUE, the line color, using the standard values
     in R.
26  ##
27  ## Value         A vector of length 2 containing the ordinates of the bounding
       line
28  ##               at the beginning and the end of x.
29  ##
30  #
     ###############################################################################

31  BoundingLines <- function(x,env=0,segments=FALSE,offset=0,ltype=1,color="blue
     "){
32  ##    JEG 2016-02-20
33  ##    requires  l1fit{L1pack}
34      require(L1pack)
35      if (env!=1 & env!=-1) {
36        print("env must be either 1 or -1")
37        return()
38      }
39      n <- length(x)
40      if (n<3) {
41        print("length of data must be greater than 2")
42        return()
43      }
44  ##      Determine L1 fit and compute residuals, and determine position of max
     res.
45      ab <- l1fit(1:n,x)
46      a <- ab$coef[1]
47      bc <- ab$coef[2]
48      r <- x-a-bc*(1:n)
49  ##      Determine position of max residual, above or below, and determine L1
     norm.
```

```
  maxr <- max(env*r)
    k <- which(env*r==maxr)
    dc <- sum(abs(r-env*maxr))
    if (n/2==(n-1)/2 & k==n/2){
##    Max residual is at middle point (n must be odd)
      x1 <- maxr+a+bc
      x2 <- maxr+a+bc*n
      if (segments) lines(c(1+offset,n+offset),c(x1,x2),col=color)
      return(c(x1,x2))
    }
    if (k<=n/2){
##    Find maximum rotation based on points on the right
##    If env=1, this is a clockwise rotation; if env=-1, it is counterclockwise.
        tans <- env*(x[(k+1):n]-x[k])/(1:(n-k))
        maxtanright <- max(tans)
        j <- which(tans==maxtanright)+k
        bright <- (x[k]-x[j])/(k-j)
        x1 <- x[k]-bright*(k-1)
        x2 <- x[k]+bright*(n-k)
      if (segments) lines(c(1+offset,n+offset),c(x1,x2),col=color)
      return(c(x1,x2))
    }
    if (k>n/2){
##    Find maximum rotation based on points on the left
##    If env=1, this is a counterclockwise rotation; if env=-1, it is clockwise.
        tans <- env*(x[1:(k-1)]-x[k])/((k-1):1)
        maxtanleft <- max(tans)
        i <- which(tans==maxtanleft)
        bleft <- (x[k]-x[i])/(k-i)
        x1 <- x[k]-bleft*(k-1)
        x2 <- x[k]+bleft*(n-k)
      if (segments) lines(c(1+offset,n+offset),c(x1,x2),col=color)
      return(c(x1,x2))
    }
}

#
    ##############################################################################

###  Functions
#
    ##############################################################################

Distances2Line <- function(xs,t0,b){
##  Computes the sum of the vertical distances of the points in a given vector
##  to a line with a given slope that goes through a given point in the vector.
##  The points are assumed to be equally spaced.
##
## Arguments
##    xs        A vector.
##    t0        The index in xs of the given point.
##    b         The slope of the line.
##
## Value        The sum of the vertical distances to the line.
  n <- length(xs)
  x0 <- xs[t0]
  sumabs <- 0
  if (t0>1) sumabs <- sumabs+sum(abs(xs[1:(t0-1)]-(x0+b*(1:(t0-1)-t0))))
  if (t0<n) sumabs <- sumabs+sum(abs(xs[(t0+1):n]-(x0+b*((t0+1):n-t0))))
  sumabs
}
#</pre>
```

c:/files/Computers_and_Software/Notes_ExamplePrograms/R_TimeSeriesPatterns/BoundingLines.R