# Chapter 2
# Preprocessing Procedures

## 2.1 Introduction

The purpose of this chapter is to present two important preprocessing procedures than need to be carried before someone moves to the phase of recognizing technical patterns in financial price series. First, the importance of detecting errors in a dataset (the editing process) and various manners of replacing missing values (imputation) are discussed. Second, after ensuring that datasets are cleaned, two methodologies of identifying regional peaks and bottoms are presented. The first method presented is the identification of regional locals by using a rolling window of fixed size, while the second method includes the identification of local extrema known in the context of data mining as perceptually important points. The identification of these regional locals is crucial in technical pattern recognition processes since the criteria used for identifying a pattern mainly refer to sequences of local extrema.

The rest of the chapter is organized as follows. In Sect. 2.2, means of dealing with problematic datasets are discussed. In Sect. 2.3 two methods for the identification of regional locals in price series are presented. More precisely, the use of a rolling window is presented in Sect. 2.3.1, while the identification of perceptually important points is presented in Sect. 2.3.2. Finally, in Sect. 2.4 we conclude. In Appendices 1 and 2, the reader will find the developed Matlab code for the methodology presented in Sects. 2.3.1 and 2.3.2 respectively.

## 2.2 Data Pre-processing

Even high quality data purchased from well reputed vendors might be problematic. The quality of data used is crucial in any empirical assessment. If not carefully pre-processed false conclusions might be reached, good trading systems



P.E. Tsinaslanidis, A.D. Zapranis, *Technical Analysis for Algorithmic Pattern Recognition*, DOI 10.1007/978-3-319-23636-0_2

might be rejected whilst in a worst-case scenario bad trading systems might be approved. The old acronym GIGO mainly used in the field of computer science can also be used in the scientific field of finance. It stands for "Garbage In, Garbage Out", and means that if invalid input data are used, invalid outputs will result.

The process of detecting errors in statistical data is called editing. Chambers (2001) mentions two different editing types: logical and statistical editing. The first one refers to the process of detecting data values which violate particular pre-defined rules. Values that fail a logical edit must be wrong and we call them illogical. The second type of editing is the process of identifying suspicious values, i.e. values that might be wrong. Values that fail the editing process are initially replaced as missing values and subsequently if necessary are replaced by known acceptable values, a process called imputation. The necessity of imputation varies, but generally it is being adopted if the trading system considered is designed to work with complete datasets.

Illogical values, as their name suggests, violate logical constraints. Common examples of illogical values are observing negative values for the trading volume, data falling on days where the market is closed (weekends or holidays) and bad quotes where for example a tick of 21.62 should be 31.62. Detecting illogical values can be in most cases a straightforward process. Consider the following example where the variable of interest is the market price of a stock. By calculating daily returns, we can easily spot daily returns which exceed the maximum decline/rise permitted, i.e. they violate the limit up or limit down boundaries (e.g. a daily change of 54 %). Such cases might result from missing prices or bad quotes as well. More generally, distances of high, low and close prices from the corresponding open prices can also be used for identifying illogical values.

Outliers are extreme values, whose presence are theoretically possible, but raise suspicions. One way to spot such cases is by calculating the variable's mean and the standard deviation, and subsequently identify observations distanced more than $\pm 3$ standard deviations from the mean. After a careful examination, if we conclude that an outlier is an error then we initially record it as a missing value. Kumiega and Van Vliet (2008) describe another data cleaning process for outliers. By this process, a compressing algorithm winsorizes outliers, by pulling them towards the mean and replacing them with a value at specified limit, say three standard deviations. However, in view of finance applications, where financial asset's returns exhibit time-varying volatility Kumiega and Van Vliet (2008) recommend winsorizing on a rolling basis.

Either because there have been missing values in the initial dataset, or because we replaced values failing the editing process with missing ones, the outcome is a dataset with "holes". Generally, there are two options to deal with this problem. We can either ignore these missing values by not including them in our final

dataset, or we can replace them with new acceptable values. The final choice depends on the application considered. The first procedure has the drawback of losing probable useful information from the initial sample. In addition, when the number of errors is significant, ignoring missing values will reduce the sample size. For the second alternative two methodologies can be used. In the first methodology, called donor imputation, the missing values for one or more variables, called recipients, are replaced by the corresponding similar to them values called donors (Beaumont and Bocci 2009). If donors are more than one then the replacing value is determined by various ways like choosing the first candidate donor, calculating the average value of all donors, choosing randomly among a set of potential donors (Random Hot-Deck imputation) or using a Nearest-Neighbour approach (Nearest-Neighbour) imputation. Alternatively, missing values can be replaced by acceptable values by using a moving average or a more complex model (like an ARIMA[1]) prior to the missing value. Finally, missing values of financial price series can also be replaced by linearly interpolating adjacent values.

However, cleaning data might also raise some problems. It is not certain whether the cleaned data were the values observed, when the actual trading took place in real time (Kumiega and Van Vliet 2008). Thus, cleaned data, when used for back-testing a trading system, might affect the results obtained and their inferences.

## 2.3 Identification of Regional Locals

Financial price series have idiosyncratic characteristics over other price series. Significant points (like regional locals and turning points) are crucial in pattern recognition processes via visual assessment. Descriptions found in the literature, for the visual identification of technical patterns, involve the assignment of criteria (conditions) to sequences of regional locals. More precisely, Neftci (1991) states:

> . . .most patterns used by technical analysts need to be characterized by appropriate sequences of local minima and/or maxima. . .

This section presents two methodologies for identifying regional locals. More precisely, in Sect. 2.3.1 the identification of local minima and maxima with a rolling window is described while in Sect. 2.3.2 the identification of perceptually important points is presented.

[1] Autoregressive Integrated Moving Average.

### 2.3.1 Identify Regional Locals with a Rolling Window

The prerequisite process for the identification of the technical patterns examined in this book, involves the identification of regional peaks and bottoms with the following developed Matlab function (see Appendix 1)[2]:

$$[Peaks, Bottoms] = RW(ys, w, pflag)$$

The $RW(\cdot)$ function takes three input variables: *ys*, *w* and *pflag*. "*ys*" is a column vector of the y-coordinates (prices) of the examined price series, "*w*" is used to define the width of the *rolling window* (RW), and "*pflag*" returns a corresponding graph if it takes the value of one. The outputs are the regional locals identified by the function. Particularly, ***Peaks*** (***Bottoms***) is a $m \times 2$ ($k \times 2$) matrix containing the coordinates of the $m$ ($k$) identified peaks (bottoms). The first and second columns of the above outputs contain the y and x-coordinates respectively. For a given price series, an observation is identified as local peak (trough) if it is the largest (smallest) of the observations in a window of size $2w + 1$ centered on this observation. The window slides by one observation in each iteration and the process is repeated until the whole price series is scanned (Kugiumtzis et al. 2007).

Let $\{p_t\}_{t=1}^{\ell}$ and $t \in [1:\ell]$ be the prices (y-coordinates) and the time (x-coordinates) of the examined price series of length $\ell$ respectively.[3] The indicator $t$ refers to the oldest observation when it takes the value of one and to the most recent observation when it takes the value of $\ell$ respectively. The process for identifying the regional locals is described in (2.1) and (2.2) $\forall\, t \in [w+1:\ell-w]$.

$$Local\ Peak\ if\ \ p_t > \max\left\{p_{[t-w:t-1]}\right\} \&\ p_t > \max\left\{p_{[t+1:t+w]}\right\} \tag{2.1}$$

$$Local\ Trough\ if\ \ p_t < \min\left\{p_{[t-w:t-1]}\right\} \&\ p_t < \min\left\{p_{[t+1:t+w]}\right\} \tag{2.2}$$

An example is illustrated in Fig. 2.1 where $RW(\cdot)$ identified 6 regional peaks and seven regional bottoms (troughs) on NASDAQ Index by adopting a rolling window of 31 days $(w = 15)$. The identification process starts from the 16th observation $(t = 16)$ and terminates when $t = \ell - 15$, where $\ell$ is the length of the price series.

Alternative methods to identify regional locals are also provided in the bibliography. By implementing kernel mean regression algorithm we can smooth the price series and identify the corresponding extrema (Lo et al. 2000; Dawson and Steeley

[2] Functions presented in this book have the following general form: [output$_1$, output$_2$, …, output$_n$] = function's name (input$_1$, input$_2$, …, input$_n$). The variables in the squared brackets are the outputs generated by the corresponding function and the variables inside the brackets are the necessary inputs. We follow the same notation used in the Matlab software since all the identification mechanisms presented in this book were developed with the use of this software.

[3] Hereafter we will use the notation $[a:b]$ to refer to all positive natural values between the closed interval $[a, b]$, where $0 < a < b$ and $a, b \in \mathbb{N}$.

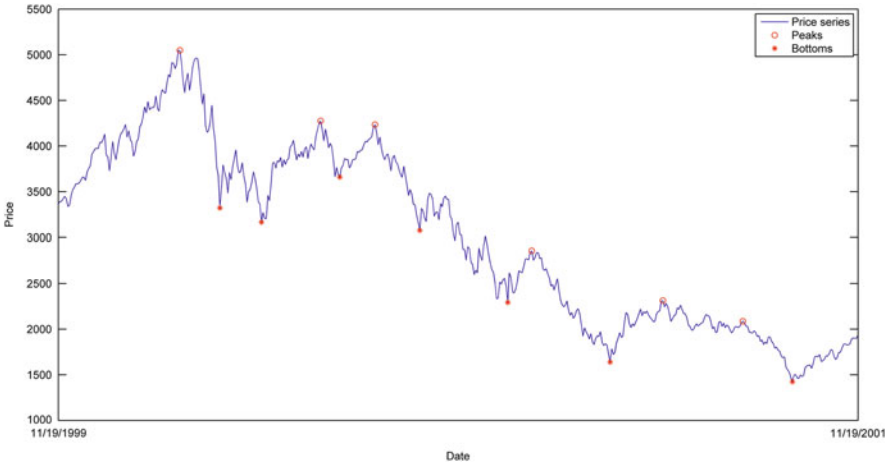


**Fig. 2.1** Identification of regional locals on NASDAQ Index. Six Peaks and seven Bottoms were identified by $RW(\cdot)$ with a rolling window of 31 days ($w = 15$)

2003; Savin et al. 2007). Lucke (2003) used a computer program which was originally designed to identify business cycle turning points. Identification of perceptually important points is an alternative algorithmic approach for detecting regional significant points and it is presented in Sect. 2.3.2.

### *2.3.2 Perceptually Important Points*

A promising method to exploit salient points from a price series is by using *Perceptually Important Points* (PIPs). The identification methodology was first introduced in Chung et al. (2001), and used in many applications on time series data mining. More precisely, they have been used mainly for purposes of dimension reduction (or else time series representation) (Fu et al. 2008; Phetchanchai et al. 2010), as a dynamic approach for time series segmentation (Fu et al. 2006; Jiang et al. 2007; Tsinaslanidis and Kugiumtzis 2014) and for clustering reasons (Fu et al. 2004).[4] PIPs have been also used in finance applications to identify technical patterns (Fu et al. 2007; Chen et al. 2013).

As a preprocessing step of our methodology, we may use PIPs in order to identify significant points. The algorithm starts by characterizing the first and the last observation as the first two PIPs. Subsequently, it calculates the distance between all remaining observations and the two initial PIPs and signifies as the third PIP the one with the maximum distance. The fourth PIP is the point that

[4] A comprehensive review on the existing time series data mining research is presented in Fu (2011), where variant methodologies that deal with the aforementioned aspects of data mining are highlighted.

maximizes its distance to its adjacent PIPs (which are either the first and the third, or the third and the second PIP). The algorithm stops when the required by the user number of PIPs is identified.

Three metrics are generally used for the distance in the PIPs algorithm, namely the Euclidean distance (ED) $d_E$, the perpendicular distance (PD) $d_P$ and the vertical distance (VD) $d_V$. Let $\{p_1, p_2, \ldots, p_\ell\}$ be the price time series of length $\ell$, and two adjacent PIPs $x_t = (t, p_t)$ and $x_{t+T} = \left(t+T, p_{t+T}\right)$. The Euclidean distance $d_E$ of each of the intermediate points $x_i = (i, p_i)$, for $i \in \{t+1, \ldots, t+T-1\}$ from the two PIPs is defined in (2.3).

$$d_E(x_i, x_t, x_{t+T}) = \sqrt{(t-i)^2 + (p_t - p_i)^2} + \sqrt{(t+T-i)^2 + \left(p_{t+T} - p_i\right)^2} \quad (2.3)$$

For the two other distances, we consider first the line connecting the two PIPs $x_t = (t, p_t)$ and $x_{t+T} = \left(t+T, p_{t+T}\right)$, $z_i = s\,i + c$, and $(i, z_i)$ the points on the line, where the slope is $s = \left(p_{t+T} - p_t\right)/T$ and the constant term is $c = p_t - t\left(p_{t+T} - p_t\right)/T$. Then the perpendicular distance $d_P$ of any intermediate point $x_i = (i, p_i)$, between the two PIPs from the line is given by (2.4).

$$d_P(x_i, x_t, x_{t+T}) = \frac{|s\,i + c - p_i|}{\sqrt{s^2 + 1}} \quad (2.4)$$

Finally, (2.5) expresses the vertical distance $d_V$ of $x_i$ to the line.

$$d_V(x_i, x_t, x_{t+T}) = |s\,i + c - p_i| \quad (2.5)$$

For any of the three distances, denoted collectively $d$, the new PIP point, $x_i^* = \left(i^*, p_{i^*}\right)$, is the one that maximizes the distance $d$ at $i^*$ (2.6). In (2.6) "*argmax*" stands for the argument of maximum.

$$i^* = \underset{i}{\text{argmax}}\left(d(x_i, x_t, x_{t+T})\right) \quad (2.6)$$

Figure 2.2 presents a step-by-step identification process of five ED-PIPs on NASDAQ Index for the requested period 19/11/1999–19/11/2001. Figure 2.3 presents a simplified illustration of the identification of the third PIP according with the three aforementioned distance measures.

At this point the algorithmic methodology of PIPs' identification is presented. Let $\boldsymbol{P} = \{p_t\}_{t=1}^{\ell}$ and $\mathbf{T} = \{t\}_{t=1}^{\ell}$ are two $\ell \times 1$ column vectors containing the prices (y-coordinates) and the time (x-coordinates) of the examined price series of length $\ell$ respectively. We subsequently define $\boldsymbol{A_{xi}}$ and $\boldsymbol{A_{yi}}$ two $\ell \times 2$ matrices containing the x- and y-coordinates respectively, of the closer adjacent PIPs for the $i^{th}$ iteration.[5]

[5] Since the first two PIPs are defined as the first and the last observation the $i^{th}$ iteration identifies the $(i+2)^{th}$ PIP.

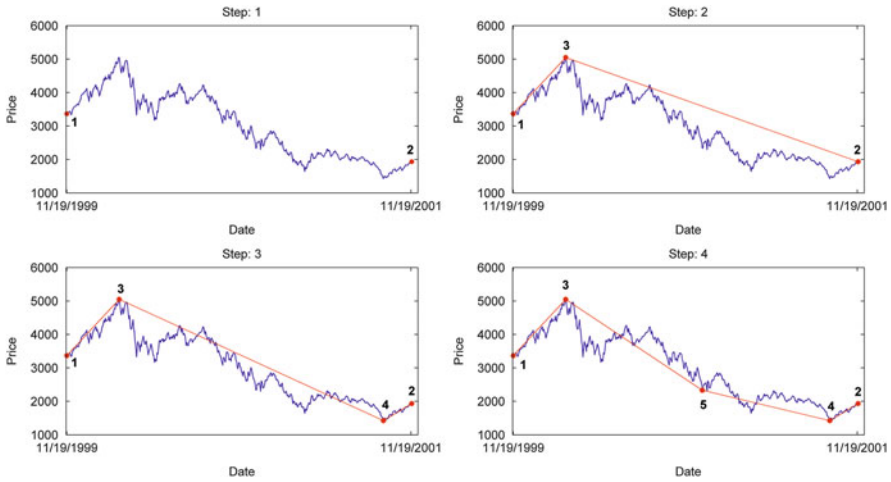


**Fig. 2.2** PIPs identification process. First 5 PIPs identified on NASDAQ Index, with the ED measure

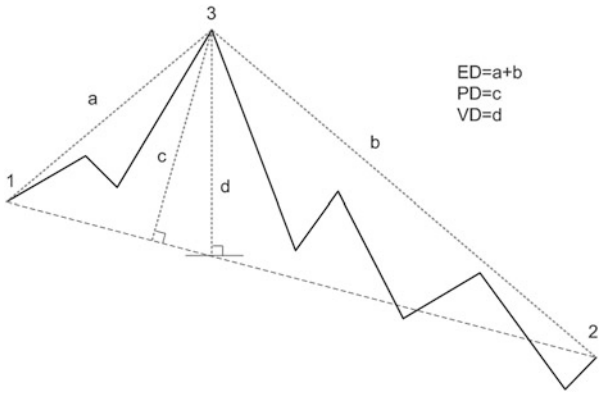


**Fig. 2.3** Identification of the third PIP with three different distance measures ED, PD and VD

Particularly, the first column contains the coordinates of the closer adjacent PIP whereas the second column contains the coordinates of the second closer adjacent PIP. Existed PIPs are indicated with "*nan*"[6] in order to avoid the identification of the same PIP. Consider the following simplified example. If $\ell = 6$, in the first iteration, $\iota = 1$, the third PIP is about to identified (since the first two PIPs are $(1, p_1)$ and $(6, p_6)$), and $\boldsymbol{A}_{x1}$ and $\boldsymbol{A}_{y1}$ are shown in (2.7) and (2.8).

[6] *nan* stands for "not-a-number". A nan value is the result from operations which have undefined numerical results. When nan is involved in a calculation (for example $nan \times 10$) the result is also nan.

$$\boldsymbol{A}_{x1} = \begin{bmatrix} nan & 1 & 1 & 6 & 6 & nan \\ nan & 6 & 6 & 1 & 1 & nan \end{bmatrix}' \tag{2.7}$$

$$\boldsymbol{A}_{y1} = \begin{bmatrix} nan & p_1 & p_1 & p_6 & p_6 & nan \\ nan & p_6 & p_6 & p_1 & p_1 & nan \end{bmatrix}' \tag{2.8}$$

In (2.7) and (2.8), $\boldsymbol{X}'$ is the transpose of $\boldsymbol{X}$. If the third identified PIP is $(4, p_4)$, in the subsequent iteration, $\iota = 2$, (2.7) becomes

$$\boldsymbol{A}_{x2} = \begin{bmatrix} nan & 1 & 4 & nan & 4 & nan \\ nan & 4 & 1 & nan & 6 & nan \end{bmatrix}' \tag{2.9}$$

and (2.8)

$$\boldsymbol{A}_{y2} = \begin{bmatrix} nan & p_1 & p_4 & nan & p_4 & nan \\ nan & p_4 & p_1 & nan & p_6 & nan \end{bmatrix}'. \tag{2.10}$$

In every iteration, (2.11) measures the ED between the adjacent PIPs and intermediate points.

$$\begin{aligned} \boldsymbol{ED}_{\boldsymbol{\iota}} = &\left\{ \left[\boldsymbol{A}_{x\iota}\,(:,1) - \boldsymbol{T}\right]^{\diamond 2} + \left[\boldsymbol{A}_{y\iota}\,(:,1) - \mathbf{P}\right]^{\diamond 2} \right\}^{\diamond 1/2} \\ &+ \left\{ \left[\boldsymbol{A}_{x\iota}\,(:,2) - \boldsymbol{T}\right]^{\diamond 2} + \left[\boldsymbol{A}_{y\iota}(:,2) - \mathbf{P}\right]^{\diamond 2} \right\}^{\diamond 1/2} \end{aligned} \tag{2.11}$$

Here, $\boldsymbol{ED}_{\boldsymbol{\iota}} = \{ed_{t,\iota}\}_{t=1}^{\ell}$ is an $\ell \times 1$ column vector containing the Euclidean distances of all intermediate points from their two adjacent PIPs for the $\iota^{th}$ iteration, $\diamond$ symbolizes the Hadamard (or else element-wise) product, $\boldsymbol{X}^{\diamond n}$ is the $n^{th}$ element-wise power of the matrix $\boldsymbol{X}$, $n \in \mathbb{R}$, and $\boldsymbol{X}(:, j)$ represents the $j^{th}$ column of matrix $\boldsymbol{X}$. The new PIP identified in $\iota^{th}$ iteration (2.12) has coordinates $PIP_{\iota} = \left(PIP_{x,\iota}, PIP_{y,\iota}\right)$, where

$$PIP_{\iota} = \left( \underset{t}{\text{argmax}}(ed_{t,\iota}),\, p_{PIP_{x,\iota}} \right). \tag{2.12}$$

Alternatively, we can identify PIPs by measuring Perpendicular Distances (PD) of the intermediate points from the lines passing through their two adjacent PIPs. As already mentioned, on a two-dimensional, Cartesian coordinate system, the points $(i, z_i)$ of a straight line are defined by $z_i = s\, i + c$, where $s$ is the slope of the line and $c$ is the constant term. Equations (2.13) and (2.14) show the calculation of $\boldsymbol{S}_{\boldsymbol{\iota}}$ and $\boldsymbol{C}_{\boldsymbol{\iota}}$ respectively, which are two $\ell \times 1$ column vectors containing accord-

ingly the slopes and constant terms of lines defined by all successive pairs of existed PIPs, identified before the completion of the $\iota^{th}$ iteration.

$$\boldsymbol{S_\iota} = \frac{\boldsymbol{A_{y\iota}}(:,1) - \boldsymbol{A_{y\iota}}(:,2)}{\boldsymbol{A_{x\iota}}(:,1) - \boldsymbol{A_{x\iota}}(:,2)} \tag{2.13}$$

$$\boldsymbol{C_\iota} = \boldsymbol{A_{y\iota}}(:,1) - \boldsymbol{S_\iota} \diamond \boldsymbol{A_{x\iota}}(:,1) = \boldsymbol{A_{y\iota}}(:,2) - \boldsymbol{S_\iota} \diamond \boldsymbol{A_{x\iota}}(:,2) \tag{2.14}$$

Subsequently, $\boldsymbol{PD_\iota} = \{pd_{t,\iota}\}_{t=1}^{\ell}$ is a $\ell \times 1$ column vector containing all points' corresponding perpendicular distances as measured within the $\iota^{th}$ iteration (2.15).[7]

$$\boldsymbol{PD_\iota} = \frac{|\boldsymbol{S_\iota} \diamond \boldsymbol{T} - \boldsymbol{P} + \boldsymbol{C_\iota}|}{\left(\boldsymbol{S_\iota}^{\diamond 2} + 1\right)^{\diamond 1/2}} \tag{2.15}$$

Similarly with the case of ED-PIPs the new PD-PIP has coordinates

$$PIP_\iota = \left(\underset{t}{\text{argmax}}(pd_{t,\iota}), p_{PIP_{x,\iota}}\right). \tag{2.16}$$

Finally by using the Vertical Distance (VD) as a distance measure, the $\boldsymbol{VD_\iota} = \{vd_{t,\iota}\}_{t=1}^{\ell}$ is an $\ell \times 1$ column vector defined by (2.17)

$$\boldsymbol{VD_\iota} = |\boldsymbol{Z}_\iota - \boldsymbol{P}| \tag{2.17}$$

where

$$\boldsymbol{Z}_\iota = \boldsymbol{S_\iota} \diamond \mathbf{T} + \boldsymbol{C_\iota}. \tag{2.18}$$

Similarly, the new VD-PIP has coordinates

$$PIP_\iota = \left(\underset{t}{\text{argmax}}(vd_{t,\iota}), p_{PIP_{x,\iota}}\right) \tag{2.19}$$

We prefer to use matrix notations for the above calculations in order to avoid "*for-loops*" in programming and enhance the computational speed. The Matlab code for the PIPs identification process we described above is provided in Appendix 2. Alternatively the reader may use the relevant equations referred in Fu et al. (2008) or (2.3)–(2.6).

[7] Since the division between matrices is not defined in Eqs. (2.13) and (2.14) an element wise division is implied.

## 2.4 Conclusions

For reliable results when assessing a trading system the use of high-quality data is necessary. Data retrieved even from well-reputed vendors may be problematic. Illogical, suspicious and missing values are the most common problems that need to be dealt before performing an empirical analysis. In this chapter these issues were discussed and various methods for dealing with these problems were highlighted.

The second part of this chapter focuses on the identification of regional locals on financial price series. Someone who is already familiar with the descriptions for identifying various technical patterns, should have noticed that these descriptions mainly refer to specific criteria that a sequence of regional locals must fulfil for a successful pattern confirmation. In this chapter two methodologies for identifying regional locals were presented. The first includes the use of a RW centered on each observation. This observation is characterized as a local peak (bottom) if it is the maximum (minimum) of all observations included in this window. This is a sequential algorithmic approach which starts from the first and terminates to the last available observation. This means that locals that correspond to earlier observations are identified first. When using RW the user must set the window size and the number of regional locals that will be identified is not known, a priori. On the contrary, PIPs identification is an algorithmic process where the user must specify the distance measure to be used and the number of regional locals to be identified. In this process a series is scanned dynamically until the desired number of regional locals is identified. It is up to the discretion of the user to decide which one of the presented methods (if any) should be used.

## Appendix 1: RW Function

### *1.1. The Function*

$$[Peaks, Bottoms] = RW(ys, w, pflag)$$

### *1.2. Description*

This function identifies regional peaks and bottoms of a price series with a RW of size $2w + 1$, and a slide step of one observation.

Inputs

1. *ys*: Price series (Column Vector).
2. *w*: Is used to define the size of the RW, which is $2w + 1$.
3. *pflag*: If 1, the function generates a plot.

Outputs

1. *Peaks*: An $(n \times 2)$ matrix containing the coordinates of the n identified peaks.
2. *Bottoms*: A $(k \times 2)$ matrix containing the coordinates of the k identified bottoms.

   The first and second column of these outputs contains the *y*- and *x*-coordinates respectively.

## *1.3. Code*

```
function [Peaks,Bottoms]=RW(ys,w,pflag)
l=length(ys);
Peaks_Bottoms=zeros(l,2);%Preallocation
for i=w+1:l-w%Index peaks and bottoms with ones
  if ys(i,1)>max(ys(i-w:i-1))...
      && ys(i,1)>max(ys(i+1:i+w))
    Peaks_Bottoms(i,1)=1;
  end
  if ys(i,1)<min(ys(i-w:i-1))...
      && ys(i,1)<min(ys(i+1:i+w))
    Peaks_Bottoms(i,2)=1;
  end
end
P_Indx=find(Peaks_Bottoms(:,1));
B_Indx=find(Peaks_Bottoms(:,2));
Peaks=[ys(P_Indx),P_Indx];
Bottoms=[ys(B_Indx),B_Indx];
if pflag==1
  plot(ys),hold on
  plot(Peaks(:,2),Peaks(:,1),'ro')
  plot(Bottoms(:,2),Bottoms(:,1),'r*')
  legend('Price series','Peaks','Bottoms'), hold off
end
end
```

# Appendix 2: PIPs Function

## 2.1. The Function

$$[PIPxy] = PIPs(ys, n\ of\ PIPs, type\ of\ dist, pflag)$$

## 2.2. Description

This function identifies the Perceptually Important Points (PIPs) on a price series.

Inputs

1. *ys*: Price series (Column Vector).
2. *n of PIPS*: Number of requested PIPs.
3. *type of dist*: 1 = (Euclidean Distance) ED, 2 = (Perpendicular Distance) PD and 3 = (Vertical Distance) VD.
4. *pflag*: If 1, the function generates a plot.

Outputs

1. *PIPxy*: A $(n\ of\ PIPs \times 2)$ matrix containing the coordinates of PIPs. The first (second) column presents the x-coordinates (y-coordinates).

## 2.3. Code

```
function [PIPxy]=PIPs(ys,nofPIPs,typeofdist,pflag)
l=length(ys);
xs=(1:l)';% Column vector with xs

PIP_points=zeros(l,1);% Binary indexation
PIP_points([1,l],1)=1;% One indicate the PIP points.The first two
PIPs are the first and the last observation.

Adjacents=zeros(l,2);
currentstate=2;% Initial PIPs

while currentstate<=nofPIPs
  Existed_Pips=find(PIP_points);
  currentstate=length(Existed_Pips);
  locator=nan(l,currentstate);
  for j=1:currentstate
     locator(:,j)=abs(xs-Existed_Pips(j,1));
  end
  b1=zeros(1,l);b2=b1;
```

```
  for i=1:l
    [~,b1(i)]=min(locator(i,:));% Closer point
    locator(i,b1(i))=nan; % Do not consider Closer point
    [~,b2(i)]=min(locator(i,:));% 2nd Closer Point
        Adjacents(i,1)=Existed_Pips(b1(i));%x-coordinates of the
closer point
        Adjacents(i,2)=Existed_Pips(b2(i));%x-coordinates of the
2nd closer points
  end
  %% Calculate Distance
  Adjx=Adjacents;
  Adjy=[ys(Adjacents(:,1)),ys(Adjacents(:,2))];
   Adjx(Existed_Pips,:)=nan;% Existed PIPs are not candidates for
new PIP.
  Adjy(Existed_Pips,:)=nan;
  if typeofdist==1
    [D]=EDist(ys,xs,Adjx,Adjy);
  elseif typeofdist==2
    [D]=PDist(ys,xs,Adjx,Adjy);
  else
    [D]=VDist(ys,xs,Adjx,Adjy);
  end
  [~,Dmax]=max(D);
  PIP_points(Dmax,1)=1;
  currentstate=currentstate+1;
end
PIPxy=[Existed_Pips, ys(Existed_Pips)];
%% Plot
if pflag==1
  plot(ys), hold on
  plot(Existed_Pips,ys(Existed_Pips),'r*'),hold off
end
end
%% Distance measures
% Euclidean Distance
function [ED]=EDist(ys,xs,Adjx,Adjy)
ED=((Adjx(:,2)-xs).^2+(Adjy(:,2)-ys).^2).^(1/2)+...
  ((Adjx(:,1)-xs).^2+(Adjy(:,1)-ys).^2).^(1/2);
end
% Perpendicular Distance
function [PD]=PDist(ys,xs,Adjx,Adjy)
slopes=(Adjy(:,2)-Adjy(:,1))./(Adjx(:,2)-Adjx(:,1));
constants=Adjy(:,2)-slopes.*Adjx(:,2);
PD=abs(slopes.*xs-ys+constants)./(slopes.^2+1).^(1/2);
% line function: y=kx+m (1)
```

```
% the perpendicular distance (PD) from a point p(x1,y1) to a line
% is given by the following formula:
% PD=abs(k*x1-y1+m)/sqrt(k^2+1)
end
% Vertical Distance
function [VD]=VDist(ys,xs,Adjx,Adjy)
slopes=(Adjy(:,2)-Adjy(:,1))./(Adjx(:,2)-Adjx(:,1));
constants=Adjy(:,2)-slopes.*Adjx(:,2);
Yshat=slopes.*xs+constants;
VD=abs(Yshat-ys);
end
```

## References


Beaumont J-F, Bocci C (2009) Variance estimation when donor imputation is used to fill in missing values. Can J Stat 37(3):400–416

Chambers R (2001) Evaluation criteria for statistical editing and imputation. National Statistics Methodological Series No. 28. Office for National Statistics, UK

Chen C-H, Tseng VS, Yu H-H, Hong T-P (2013) Time series pattern discovery by a PIP-based evolutionary approach. Soft Comput 17:1699–1710

Chung FL, Fu TC, Luk R, Ng V (2001) Flexible time series pattern matching based on perceptually important points. Paper presented at the international joint conference on artificial intelligence workshop on learning from temporal and spatial data

Dawson ER, Steeley JM (2003) On the existence of visual technical patterns in the UK stock market. J Bus Financ Account 30(1 and 2):263–293

Fu TC (2011) A review on time series data mining. Eng Appl Artif Intell 24(1):164–181

Fu TC, Chung FL, Luk R, Ng CM (2004) Financial time series indexing based on low resolution clustering. In: Workshop at the 4th international conference on data mining, pp. 5–14

Fu TC, Chung FL, Luk R, Ng CM (2007) Stock time series pattern matching: template-based vs. rule-based approaches. Eng Appl Artif Intell 20(3):347–364

Fu TC, Chung FL, Luk R, Ng CM (2008) Representing financial time series based on data point importance. Eng Appl Artif Intell 21(2):277–300

Fu TC, Chung FL, Ng CM (2006) Financial time series segmentation based on specialized binary tree representation. In: International conference on data mining, pp. 3-9

Jiang J, Zhang Z, Wang HA (2007) New segmentation algorithm to stock time series based on PIP approach. In: International conference on wireless communications, networking and mobile computing, pp. 5609–5612

Kugiumtzis D, Vlachos I, Papana A, Larsson PG (2007) Assessment of measures of scalar time series analysis in discriminating Preictal states. Int J Bioelectromag 9(3):134–145

Kumiega A, Van Vliet B (2008) Quality money management. Elsevier, Amsterdam

Lo AW, Mamaysky H, Wang J (2000) Foundations of technical analysis: computational algorithms, statistical inference, and empirical implementation. J Financ 55(4):1705–1765

Lucke B (2003) Are technical trading rules profitable? Evidence for head-and-shoulder rules. Appl Econ 35:33–40

Neftci SN (1991) Naïve trading rules in financial markets and Wiener-Kolmogorov prediction theory: a study of "Technical Analysis". J Bus 64(4):549–571

Phetchanchai C, Selamat A, Rehman A, Saba T (2010) Index financial time series based on zigzag-perceptually important points. J Comput Sci 6(12):1389–1395

Savin G, Weller P, Zvingelis J (2007) The predictive power of "Head-and-Shoulders" price patterns in the U.S. stock market. J Financ Econometr 5(2):243–265

Tsinaslanidis PE, Kugiumtzis D (2014) A prediction scheme using perceptually important points and dynamic time warping. Expert Syst Appl 41(15):6848–6860

Springer

http://www.springer.com/978-3-319-23635-3

Technical Analysis for Algorithmic Pattern Recognition

Tsinaslanidis, P.E.; Zapranis, A.D.

2016, XIV, 204 p., Hardcover

ISBN: 978-3-319-23635-3