R Package Development by Means of Literate Programming (noweb)

Bernhard Pfaff

October 17, 2016

1 Introduction

2 Detecting Peaks/Troughs

2.1 Notation

A uniformly sampled time series $\mathbf{y} = \{y_1, \dots, y_i, \dots, y_T\}$ with T data points is considered. The detection of peak/trough points is achieved by a function $S(i, y_i, T)$ that returns for data point y_i a score value.[1]. If this score value surpasses a user-provided threshhold value θ, /i.e/, $S(i, y_i, T) \geq \theta$ then the point is considered as a local peak/trough.

Furthermore, in case local peak/trough points appear closely together with respect to time (clustered), then these points can be classified as a burst or bust, respectively.

2.2 Algorithms

In Palshikar (2009) five different score functions S have been suggested. All have in commom, that a centred window of size $2*k+1$ around y_i is considered. That is, for a positive integer k the k right neigbours $N^+(i, k, T) = \{y_{i+1}, \dots, y_{i+k}\}$ and the k left neighbours $N^-(i, k, T) = \{\{y_{i-k}, \dots, y_{i-1}\}$ are employed for assessing y_i as a local peak/trough. The union of $N^-(i, k, T)$ and $N^+(i, k, T)$ is defined as $N(i, k, T) = N^-(i, k, T) \cdot N^+(i, k, T)$ and if the centre point is included as $N'(i, k, T) = N^-(i, k, T) \cdot y_i \cdot N^+(i, k, T)$.

The first function, S_1, computes the score value as the average of the maximum differences between y_i with its left and right neighbours. The function is defined as:

$$S_1 = \frac{\max(y_i - y_{i-1}, \dots, y_i - y_{i-k}) + \max(y_i - y_{i+1}, \dots, y_i - y_{i+k})}{2} \tag{1}$$

The equation (1) can be casted in R as:

1 ⟨*score-maxdiff* 1⟩≡

```
scmaxdiff <- function(x, k){
```

[1] It suffices to provide a score function for peaks only. Trough points can be detected by using the negative values of the series $\mathbf{y}$.

```
    cp <- k + 1L
    lmax <- max(x[cp] - head(x, k))
    rmax <- max(x[cp] - tail(x, k))
    (lmax + rmax) / 2.0
}
```

Defines:
`scmaxdiff`, used in chunks 4–6.

Instead of using the maximum differences of y_i with its k left and right neighbours as in (1), an alternative is to compute the mean differences and evaluate the average thereof:

$$S_2 = \frac{\frac{(y_i - y_{i-1}, \ldots, y_i - y_{i-k})}{k} + \frac{(y_i - y_{i+1}, \ldots, y_i - y_{i+k})}{k}}{2} \tag{2}$$

This equation can be casted in R as:

2a ⟨*score-diffmean* 2a⟩≡

```
scdiffmean <- function(x, k){
    cp <- k + 1L
    ldmean <- x[cp] - mean(head(x, k))
    rdmean <- x[cp] - mean(tail(x, k))
    (ldmean + rdmean) / 2.0
}
```

Defines:
`scdiffmean`, used in chunks 4–6.

Another variation of score computation that has been proposed by Palshikar (2009) is to consider the differences to the mean of the k left and right neighbours, that is:

$$S_3 = \frac{(y_i - \frac{(y_{i-1}, \ldots, y_{i-k})}{k}) + (y_i - \frac{(y_{i+1}, \ldots, y_{i+k})}{k})}{2} \tag{3}$$

The equation (3) can be casted as R function `scavgdiff` for instance as follows:

2b ⟨*score-avgdiff* 2b⟩≡

```
scavgdiff <- function(x, k){
    cp <- k + 1L
    lmean <- mean(x[cp] - head(x, k))
    rmean <- mean(x[cp] - tail(x, k))
    (lmean + rmean) / 2.0
}
```

Defines:
`scavgdiff`, used in chunks 4–6.

The fourth proposed score function differs from the previous three in the sense that it does take explicitly the differences between y_i and its neighbours explicitly into account, but tries to capture its information content by means of relative entropy. The entropy of a vector A with elements $A = \{a_1, \ldots, a_m\}$ is given as:

$$H_w(A) = \sum_{i=1}^{M} (-p_w(a_i)\log(p_w(a_i))) \quad (4)$$

where $p_w(a_i)$ is an estimate of the density value at a_i. The score function is now based on computing the entropies of $H(N((k, i, T))$ and $H(N'(k, i, T))$. Hereby, the densities can be determined by means of a kernel density estimator. The score function is then defined as the difference of the entropies:

$$S_4 = H(N((k, i, T)) - H(N'((k, i, T)) \quad (5)$$

This concept is implemented in the function `scentropy()`. The empirical density is computed by calling `density()`. The ellipsis argument of `scentropy()` is passed down to this function and hereby allowing the user to employ other than the default arguments of `density()`.

3 ⟨*score-entropy* 3⟩≡

```
scentropy <- function(x, k, ...){
    cp <- k + 1L
    dfull <- density(x, ...)$y
    hfull <- sum(-dfull * log(dfull))
    dexct <- density(x[-cp], ...)$y
    hexct <- sum(-dexct * log(dexct))
    hfull - hexct
}
```

Defines:
`scentropy`, used in chunks 4–6.

Finally, a moment-based score function has been put forward in the article by Palshikar. Hereby, the first and second moment of $N((k, i, T))$ are computed and a t-type statistic can be computed as $(y_i - m)/s$. If this statistic surpasses a provided threshhold h, then the data point is considered as a local peak/trough.

$$S_5 = \begin{cases} 1 & (y_i - m)/s \geq h \\ 0 & \text{else} \end{cases} \tag{6}$$

This type of scoring algorithm is implemented as function `scttype()` below:

4a ⟨*score-ttype* 4a⟩≡

```
scttype <- function(x, k, tval){
    cp <- k + 1L
    m <- mean(x[-cp])
    s <- sd(x[-cp])
    tstat <- (x[cp] - m) / s
    if ( abs(tstat) < tval ){
        tstat <- 0
    }
    tstat
}
```

Defines:
`scttype`, used in chunks 4–6.

Incidentally, an ensemble forecast of these five algorithms can be utilized for local peak/trough classification can be employed. Hereby, one could either use a hybrid approach, whereby only those data points are considered as peak/trough points, if all five methods coincide. This concept is casted in the function `schybrid()`. Hereby, the signs of all five scoring algorithm are tested for equality.

4b ⟨*score-hybrid* 4b⟩≡

```
schybrid <- function(x, k, tval, ...){
    s <- c(sign(scmaxdiff(x, k)),
           sign(scavgdiff(x, k)),
           sign(scdiffmean(x, k)),
           sign(scentropy(x, k, ...)),
           sign(scttype(x, k, tval)))
    val <- unique(s)
    if ( length(val) < 2 ){
        return(s[1])
    } else {
        return(0)
    }
}
```

Defines:
`schybrid`, used in chunk 6a.
Uses `scavgdiff` 2b, `scdiffmean` 2a, `scentropy` 3, `scmaxdiff` 1, and `scttype` 4a.

It is also conceivable to base the classification on a majority vote. For instance, if three out of the five algorithm classify a data point as a local peak/trough, then this is taken as sufficient evidence. This approach is defined in the function `scvote()` below. The count of same 'votes' is set by the argument `confby`. Its default value is 3, *i.e.* a simple majority. For `confby = 5` the function would return the same classification as `schybrid()` does.

5 ⟨*score-vote* 5⟩≡

```
scvote <- function(x, k, tval, confby = 3, ...){
    s <- c(sign(scmaxdiff(x, k)),
           sign(scavgdiff(x, k)),
           sign(scdiffmean(x, k)),
           sign(scentropy(x, k, ...)),
           sign(scttype(x, k, tval)))
    pos <- rep(1, 5)
    zer <- rep(0, 5)
    neg <- rep(-1, 5)
    spos <- sum(s == pos)
    szer <- sum(s == zer)
    sneg <- sum(s == neg)
    v <- c(spos, szer, sneg)
    idx <- which(v >= confby)
    vals <- c(1, 0, -1)
    if ( length(idx) > 0 ){
        return(vals[idx])
    } else {
        return(0)
    }
}
```

Defines:
`scvote`, used in chunk 6a.
Uses `scavgdiff` 2b, `scdiffmean` 2a, `scentropy` 3, `scmaxdiff` 1, and `scttype` 4a.

2.3 Combining score methods

6a ⟨*score-wrapper* 6a⟩≡

```
score <- function(x, k,
                  scoreby = c("vote", "avg", "diff", "max", "ent",
                              "ttype", "hybrid"),
                  tval = 1.0, confby = 3, ...){
    scoreby <- match.arg(scoreby)
    ans <- switch(scoreby,
                  vote = scvote(x, k, tval, confby, ...),
                  avg = scavgdiff(x, k),
                  diff = scdiffmean(x, k),
                  max = scmaxdiff(x, k),
                  ent = scentropy(x, k, ...),
                  ttype = scttype(x, k, tval),
                  hybrid = schybrid(x, k, tval, ...)
                  )
    ans
}
```

Defines:
`score`, used in chunk 7.
Uses `scavgdiff` 2b, `scdiffmean` 2a, `scentropy` 3, `schybrid` 4b, `scmaxdiff` 1, `scttype` 4a, and `scvote` 5.

The content/structure of the file `score.R` is given as:

6b ⟨*score.R* 6b⟩≡

⟨*man-func-score* 10⟩
⟨*score-wrapper* 6a⟩
```
#' @rdname score
```
⟨*score-maxdiff* 1⟩
```
#' @rdname score
```
⟨*score-diffmean* 2a⟩
```
#' @rdname score
```
⟨*score-avgdiff* 2b⟩
```
#' @rdname score
```
⟨*score-entropy* 3⟩
```
#' @rdname score
```
⟨*score-ttype* 4a⟩
```
#' @rdname score
```
⟨*score-hybrid* 4b⟩
```
#' @rdname score
```
⟨*score-vote* 5⟩

This code is written to file `score.R`.

@

Within this file, all score-related methods and the wrapper-function `score()` is included. The function definitions are interspersed with the roxygen tags, which will be parsed to the `Rd`-file `score.Rd`.

So far the function `score()` has been created, by which a single point is assessed for being a local maximum or minimum. For analyzing a whole time series for its local extrema, this routine can be applied to each data point and its left/right neighbours. This task is accomplished with the function `hiker()` as defined next.

7 ⟨*hiker-func* 7⟩≡

```
hiker <- function(y, k,
                  scoreby = c("vote", "avg", "diff", "max", "ent",
                              "ttype", "hybrid"),
                  tval = 1.0, confby = 3, ...){
⟨hiker-check 8a⟩
    ## rolling centered window for peak scores
    s <- rollapply(y, width = ms, FUN = score,
                   k = k, scoreby = scoreby, tval = tval, ...)
⟨hiker-output 8b⟩
}
```

Defines:
  `hiker`, used in chunk 9a.
Uses `score` 6a.

8a ⟨*hiker-check* 8a⟩≡

```
y <- as.zoo(y)
## checking arguments
k <- as.integer(abs(k))
ms <- 2 * k + 1L
if ( is.null(dim(y)) ){
    yname <- "series"
    n <- length(y)
    if ( n < ms ) {
        stop(paste("Sample size of 'y' is ", n,
                   " and k = ", k, ".\n", sep = ""))
        }
} else {
    n <- nrow(y)
    yname <- colnames(y)[1]
    if ( n < ms ) {
        stop(paste("Sample size of 'y' is ", n,
                   " and k = ", k, ".\n", sep = ""))
    }
    if ( ncol(y) > 1 ) {
        stop("Provide univariate time series of S3-class 'zoo'.\n")
    }
}
if ( (confby < 3) || (confby > 5) ){
    stop("\nArgument 'confby' must be integer and in set {3, 4, 5}.\n")
}
scoreby <- match.arg(scoreby)
```

@

8b ⟨*hiker-output* 8b⟩≡

```
## merging time series and scores
ans <- merge(y, s)
colnames(ans) <- c("Series", "Scores")
des <- switch(scoreby,
              vote = "majority vote",
              avg = "average of averaged differences",
              diff = "average of mean differences",
              max = "average of maximum differences",
              ent = "difference of entropies",
              ttype = "t-type statistic",
              hybrid = "hybrid")
new("HikeR", ys = ans, k = k, scoreby = des, yname = yname)
```

@

8c ⟨*hiker.R* 8c⟩≡

⟨*man-func-hiker* 11a⟩
⟨*hiker-func* 7⟩

This code is written to file `hiker.R`.

@

3 Package structure

3.1 Preliminaries

First, a skeleton of the package

9a ⟨*DESCRIPTION.R* 9a⟩≡

```
Package: hiker
Title: Local Peak and Trough of a Time Series
Version: 0.0.0.9000
Authors@R: person("Bernhard", "Pfaff", email = "bernhard@pfaffikus.de",
                  role = c("aut", "cre"))
Description: Methods for detecting local peaks and troughs of a time series.
Depends: R (>= 3.3.1), zoo, methods
License: GPL-3
Encoding: UTF-8
LazyData: true
```

Uses `hiker` 7.
This code is written to file `DESCRIPTION.R`.

@

3.2 Import directives and S4-classes

9b ⟨*Allclasses.R* 9b⟩≡

```
#' @import methods
NULL
#' @import zoo
NULL
#' @importFrom stats density sd na.omit start end smooth
NULL
#' @importFrom utils head tail
NULL

# Setting old (aka S3) classes
setOldClass("zoo")


⟨man-class-HikeR 11b⟩
setClass("HikeR", slots = list(ys = "zoo",
                               k = "integer",
                               scoreby = "character",
                               yname = "character"))

⟨man-class-PTBB 12⟩
setClass("PTBB", slots = list(pt = "zoo",
                              type = "character",
                              h = "numeric"))
```

This code is written to file `Allclasses.R`.

@

4 Appendix

4.1 Roxygen Documentation

4.2 Documentation of functions

10 ⟨*man-func-score* 10⟩≡

```
#’ Basic scoring methods for local minima and maxima
#’
#’ These are basic functions for evaluating the centre
#’ point of a time series as local minimum or maximum.
#’ Hereby, a score value is computed according to various methods.
#’ If the score is positive, the centre point is tentatively
#’ classified as a local peak.
#’ Incidentally, negative scores indicate a local minima.
#’
#’ @param x \code{numeric}, vector of length \code{2 * k + 1}.
#’ @param k \code{integer}, the count of left/right neighbours.
#’ @param scoreby \code{character}, the scoring method to be used.
#’ @param tval \code{numeric}, factor for standard deviation band
#’ if \code{scoreby = ’ttype’}.
#’ @param confby \code{integer}, count of minimum vote,
#’ values in the set \code{3:5}.
#’ @param ... ellipsis argument.
#’
#’ @name score
#’ @family scores
#’ @return \code{numeric}, the score value.
NULL

#’ @rdname score
#’ @export
```

@

11a ⟨*man-func-hiker* 11a⟩≡

```
#' Peak/trough scores of time series points
#'
#' This function computes the score value for each
#' data point of a time series. The first and last
#' \code{k} observations are set to \code{NA}.
#'
#' @inheritParams score
#' @param y \code{zoo}, univariate time series.
#' @return An object of S4-class \code{HikeR}.
#' @family scores
#'
#' @references Girish K. Palshikar. Simple Algorithms for
#' Peak Detection in Time-Series. In \emph{Proc. 1st Int. Conf. Advanced Data Analysis,
#' Business Analytics and Intelligence}, 2009.
#'
#' @examples
#' TEX <- SP500[, "TEX"]
#' ans <- hiker(TEX, k = 8, scoreby = "hybrid", tval = 0.1)
#' ans
#' plot(ans)
#'
#' @export
```

@

4.3 Documentation of S4-classes

11b ⟨*man-class-HikeR* 11b⟩≡

```
#' S4 class \code{HikeR}
#'
#' Formal class for classifying local minima and maxima
#' of a time series.
#'
#' @slot ys \code{zoo}, time series with associated scores.
#' @slot k \code{integer}, count of left/right neigbours around centre point.
#' @slot scoreby \code{character}, scoring method.
#' @slot yname \code{character}, name of the series.
#' @exportClass HikeR
```

@

12 ⟨*man-class-PTBB* 12⟩≡

```
#' S4 class \code{PTBB}
#'
#' Formal class for peaks, troughs, burst, busts and
#' intermittent phase of a time series.
#'
#' @slot pt \code{zoo}, logical: indicating peak/trough points.
#' @slot type \code{character}, type of point/phase.
#' @slot h \code{numeric}, the threshhold for score evaluation.
#' @exportClass PTBB
```

@

4.4 Makefile

5 Chunk Index

6 Identifier Index

References

Palshikar, G. (2009). Simple algorithms for peak detection in time-series. In *First Int. Conf. Advanced Data Analysis, Business Analytics and Intelligence*, Ahmedabad, India.