The **otsad** Package: Online Time-Series Anomaly Detectors

Alaiñe Iturria

Abstract

Anomalies in time-series data give essential and often actionable information to be extracted for critical situations. Although this field has been studied for generations, due to the continuous evolution of technology, new challenges arise that require further improvement and evolution of anomaly detection techniques. In this paper we present the **ostad** package that implements five of the most recent detection algorithms capable of dealing with various challenges, such as online and non-stationary time-series anomaly detection.

1 Introduction

Anomalies in time-series data give essential and often actionable information to be extracted for critical situations. This is an important task in many domains such as fault detection in manufacture, intrusion detection in cyber security or fraud detection in banks.

Anomaly detection is a wide field that has been studied for years. Chandola et al. [3], Hodge et al. [5] y Zhang et al. [8] provide a comprehensive overview of anomaly detection techniques. The detection of anomalies involves the identification of patterns in the data that differ from expected behavior. In the classical literature the anomaly detection methods can be broadly categorized in six families: classification-based, clustering-based, nearest neighbors-based, statistics, information-theoretic based and spectral techniques.

Due to the evolution of new technologies, the amount of data is increasing and they are collected faster. For this reason, detection techniques must face new challenges such as increased data and online processing capacity. In comparison to classic techniques, online time-series anomaly detection techniques do not have the complete data set to work with and must take time into account. In recent years, most of the work has focused on the evolution of predictive and moving window-based techniques [4].

In R statistical software, there are few packages available from *Comprehensive R Archive Network (CRAN)* to address the problem of time-series anomaly detection. The first and most popular is the **tsoutliers** package. There is also the **qicharts** package that implements basic control chart algorithms. Finally, another package that is becoming increasingly popular is **Twitter's AnomalyDetection** package (only available on GitHub). One of the main disadvantages of these three packages is that the implemented algorithms are not able to work online. However, there are two other packages, **SmartSifter** and **EnergyOnlineCPM** available for online time-series anomaly detection.

In conclusion, CRAN has very few algorithms implemented for the online detection of anomalies in time-series. For this reason, we propose to release the **otsad** package that implements 5 recent and powerful algorithms for the detection of anomalies in univariate time-series able to work online.

In the following section 2 we briefly introduce what online time-series anomaly detection is all about. In section 3 we show how to use the **otsad** package using two easy examples. Finally, in section 4, we present an overview of this work.

2 Online anomaly detection in time-series

The detection of anomalies in online time-series is a difficult task because they must be able to consider the time (or the position of each observation) and they do not have the complete dataset work with. As mentioned earlier, in last years it is focusing on the evolution of predictive-based and moving window-based detection techniques.

Predictive techniques are models in which the parameters or components of the model are modified as new data arrive to capture better normal data trends. They calculate if a value is anomalous as the dissimilarity or the error obtained between the observation and its prediction. On the other

hand, techniques based on moving windows are usually used to improve distance-based techniques. As is well known, distance calculation is an expensive process and proportional to the number of observations to be considered. The use of moving windows allows reducing the set of observations to be considered and maintaining the most recent subset of data.

Another major challenge with the introduction of online processing is that time-series can be stationary and non-stationary. A time-series is said to be stationary when the mean and variance remain constant over time. In addition, they must be neither seasonal nor trendy. Most statistical or predictive techniques assume that time-series are stationary, when in most real cases they never are. It is true that there are techniques to turn a non-stationary time-series into a stationary one, however, this is not a simple task and there are cases that cannot be made stationary.

With the **ostad** package we provide five algorithms that address these three challenges. All algorithms can be used online. In addition, two of the five algorithms can be used in stationary environments and the other three in non-stationary environments. Finally, three of the techniques are based on predictive methods while the other two use moving windows based on techniques.

3 The otsad package

The package provided implements, documents, explains and provides references for a set of detectors listed below. In our opinion, this is the first review and package for R that develops a set of current and powerful anomaly detectors for the online detection of anomalies in time-series.

The **otsad** package implements five anomaly detection algorithms: PEWMA, SD-EWMA, TSSD-EWMA, KNNN-CAD and KNN-LDCD. The first three algorithms belong to prediction-based techniques and the last two belong to window-based techniques. Figure 1 shows a table with the most important characteristics of each algorithm.

- *PEWMA* or *Probabilistic reasoning for streaming anomaly detection* [2]. Probabilistic reasoning for streaming anomaly detection. This algorithm is a probabilistic method of EWMA which dynamically adjusts the parameterization based on the probability of the given observation. This method produces dynamic, data-driven anomaly thresholds which are robust to abrupt transient changes, yet quickly adjust to long-term distributional shifts.
- *SD-EWMA* or *Shift-Detection based on EWMA* [7].This algorithm is a novel method for covariate shift-detection tests for univariate time-series. It works in an online mode and it uses an exponentially weighted moving average (EWMA) model based control chart to detect the covariate shift-point in stationary time-series.
- *TSSD-EWMA* or *Two-Stage Shift-Detection based on EWMA* [7]. This algorithm is a novel method for covariate shift-detection tests based on a two-stage structure for univariate time-series. It works in an online mode and it uses an exponentially weighted moving average (EWMA) model based control chart to detect the covariate shift-point in non-stationary time-series. This algorithm works in two phases. In the first phase, it detects anomalies using the SD-EWMA algorithm. In the second phase, it checks the veracity of the anomalies using the Kolmogorov-Simirnov test to reduce false alarms.
- *KNN-CAD* or *Conformal k-NN Anomaly Detector* [1]. This algorithm is a model-free anomaly detection method for univariate time-series which adapts itself to non-stationarity in the data stream and provides probabilistic abnormality scores based on the conformal prediction paradigm.
- *KNN-LDCD* or *KNN - Lazy Drifting Conformal Detector* [6]. This algorithm is a variant of the KNN-CAD algorithm. The difference between the two, lies in the calculation of the measure of dissimilarity and in the calculation of the conformity function. The KNN-CAD and KNN-LDCD algorithms use both distance-based techniques, KNN, and statistical techniques to determine the degree of anomaly.

Each of these algorithms has been implemented to work in two different scenarios. On the one hand, classical processing, used when the complete data set (train and test) is available. On the other hand, the incremental processing used when no data set is available. It allows you to calculate the abnormality of the new observation(s) with the parameters updated in the last run performed.

The rest of the section is organized as follows. Section 3.1 is devoted to the installation process. In section 3.2 we present the documentation. Finally, in Section 3.3, we show how to use the algorithms using two simple examples.

3.1 Installation

The **otsad** package is available at GitHub repository, so it can be downloaded and installed directly from the R command line by typing:

Online anomaly detectors	Features	
	Stationarity	**Technique**
PEWMA	Stationary	Prediction
SD-EWMA	Stationary	Prediction
TSSD-EWMA	Non-stationary	Prediction
KNN-CAD	Non-stationary	Window-based
KNN-LDCD	Non-stationary	Window-based

Figure 1: Features of the anomaly detectors available in the **otsad** package.

```
install.packages("devtools")
devtools::install_github("alaineiturria/otsad")
```

To easily access all the package's functions, it must be attached in the usual way:

```
library(otsad)
```

3.2 Documentation

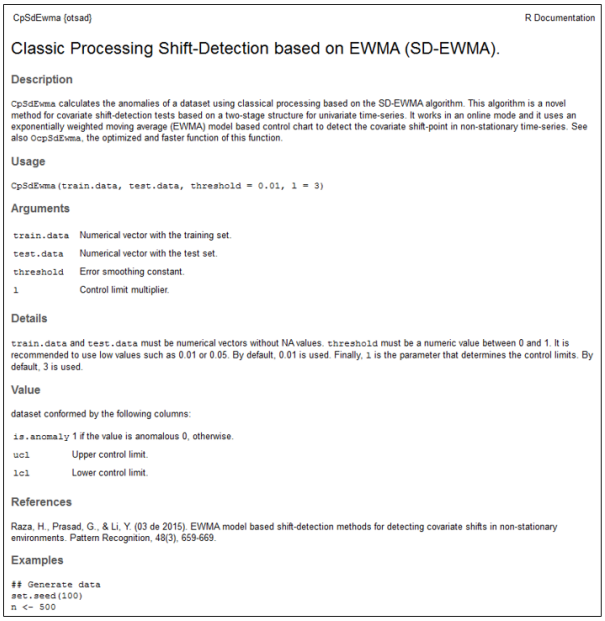
Considering that this vignette provides the user with an overview of the **otsad** package, it is also important to have access to the specific information of each of the available algorithms. This information can be checked on the documentation page corresponding to each algorithm. In all cases, the documentation has the same structure, consisting of the following sections (see Figure 2 for an example):

- A *description* section, which gives a brief description of what the algorithm consists of (like those given in Section 3).
- A *usage* section, where an overview of the function with the available parameters is given.
- A *arguments* section, where each of the input parameters is described.
- A *details* section, , which provides the user with more details on the algorithm, conditions and recommendations on the values that can be taken by each of the input parameters.
- A *value* section, where the output parameters of the function are described.
- A *references* section that points to the original contribution where the detector(s) was proposed, where further details, motivations or contextualization can be found.
- A *examples* section, where one or more examples of the use of the functions are shown.

As usual in R, the documentation pages for each function can be loaded from the command line with the commands `?` or `help`:

```
?CpSdEwma
help(CpSdEwma)
```

In addition, a user manual is available in the github repository of the **otsad** package, which contains the complete documentation of the package and its functions.



CpSdEwma {otsad} R Documentation

Classic Processing Shift-Detection based on EWMA (SD-EWMA).

Description

`CpSdEwma` calculates the anomalies of a dataset using classical processing based on the SD-EWMA algorithm. This algorithm is a novel method for covariate shift-detection tests based on a two-stage structure for univariate time-series. It works in an online mode and it uses an exponentially weighted moving average (EWMA) model based control chart to detect the covariate shift-point in non-stationary time-series. See also `OcpSdEwma`, the optimized and faster function of this function.

Usage

```
CpSdEwma(train.data, test.data, threshold = 0.01, l = 3)
```

Arguments

`train.data`	Numerical vector with the training set.
`test.data`	Numerical vector with the test set.
`threshold`	Error smoothing constant.
`l`	Control limit multiplier.

Details

`train.data` and `test.data` must be numerical vectors without NA values. `threshold` must be a numeric value between 0 and 1. It is recommended to use low values such as 0.01 or 0.05. By default, 0.01 is used. Finally, `l` is the parameter that determines the control limits. By default, 3 is used.

Value

dataset conformed by the following columns:

`is.anomaly`	1 if the value is anomalous 0, otherwise.
`ucl`	Upper control limit.
`lcl`	Lower control limit.

References

Raza, H., Prasad, G., & Li, Y. (03 de 2015). EWMA model based shift-detection methods for detecting covariate shifts in non-stationary environments. Pattern Recognition, 48(3), 659-669.

Examples

```
## Generate data
set.seed(100)
n <- 500
```

Figure 2: Extract from CpSdEwma detectors documentation page, showing the highlighted above aspects.

3.3 Use example

Here is an example of how to solve a simple problem. The example will be made with the SD-EWMA algorithm. The data has been generated as follows:

```
## Generate data
set.seed(100)
n <- 500
x <- sample(1:100, n, replace = TRUE)
x[70:90] <- sample(110:115, 21, replace = TRUE) # distributional shift
x[25] <- 200 # abrupt transient anomaly
x[320] <- 170 # abrupt transient anomaly
df <- data.frame(timestamp = 1:n, value = x)
```

We can visualize the time-series as in Figure 3 by typing:

```
plot(x = df$timestamp, y = df$value, type = "l",
     main = "Time-Serie", col = "blue", xlab = "Time", ylab = "Value")
```

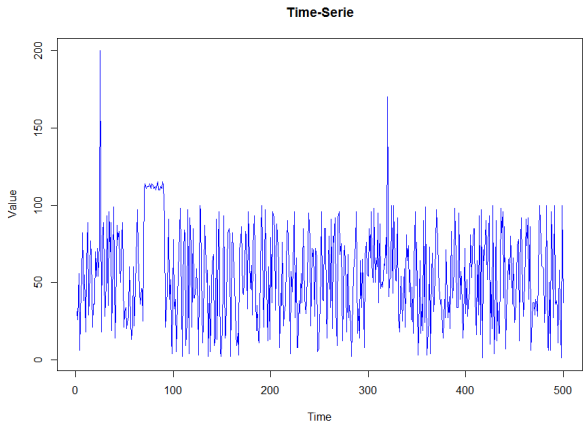


Figure 3: Our time-series visualization

The SD-EWMA algorithm is designed for stationary time-series. Therefore, we must first check that the time-series is stationary. To do this, we can display the acf and pacf graphics (Figure 4) by:

```
forecast::Acf(ts(df$value), main = "ACF", lag = 20)
forecast::Pacf(ts(df$value), main = "PACF", lag = 20)
```

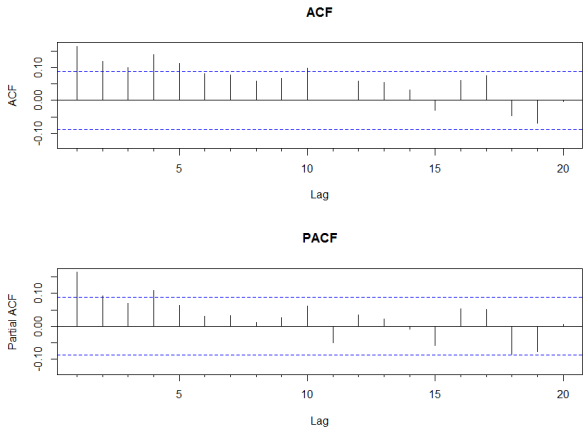


Figure 4: ACF and PACF graphs

We can observe that as almost all the lags are within the limits, the time-series is stationary. In addition, we can use two statistical tests, adf test and kpss test. The tests also indicate that the time-series is stationary.

```
library(tseries)
adf.test(df$value, alternative = 'stationary', k = 0)
kpss.test(df$value)
```

On the recommendation of the algorithm authors, also indicated in the details section of this function, we set `threshold` to 0.01.One of the usual values for the σ multiplier `l` is usually 3. Next, we apply the SD-EWMA anomaly detector, as the example is simple we use the classical processing algorithm. Note that the optimized algorithm could be used in the same way. Finally, we separate the training and test sets, using the first five values for training and the rest values for testing.

```
train <- df[1:5,"value"]
test <- df[6:n,"value"]
result <- CpSdEwma(train, test, threshold = 0.01, l = 3)
```

If we print the results we can observe that the results are of data.frame type and have three columns: *is.anomaly* indicating whether or not the test observation is anomalous and the columns *ucl* and *lcl* with the upper and lower control limits, used to determine whether or not the observation is anomalous.

```
head(result, n = 10)
```

```
##    is.anomaly        lcl      ucl
## 1       FALSE -16.718459 83.33762
## 2       FALSE -15.120715 84.87796
## 3       FALSE -12.127443 91.30896
## 4       FALSE -12.029491 90.89286
## 5       FALSE -10.427275 92.40430
## 6       FALSE -12.931167 90.31049
## 7       FALSE -10.756584 92.99798
## 8       FALSE  -7.669897 99.48715
## 9       FALSE  -9.332986 97.76851
## 10      FALSE  -9.501358 97.09333
```

Finally, we can see the results (Figure 5) by writing the following code:

```
library(ggplot2)
res <- cbind(df[6:n,], result)
rownames(res) <- 1:length(test)
```

```
ggplot(res, aes(timestamp, value)) +
      geom_line(na.rm=TRUE, color = "blue") +
      ggtitle("SD-EWMA Anomaly Detector") +
      xlab("Time") + ylab("Value") +
      theme(plot.title = element_text(lineheight=.8, face="bold", size = 20)) +
      theme(text = element_text(size=18)) +
      geom_point(data = res[res$is.anomaly == TRUE,],aes(colour = factor(is.anomaly))) +
      theme(legend.position="none")
```

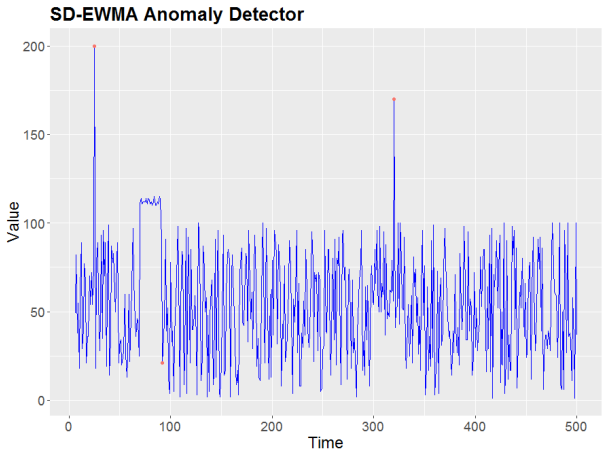


Figure 5: Sd-Ewma Anomaly detector results

In Figure 5 we can see that the detector has detected both abrupt transient anomalies well and the third one, distributional shift anomaly, a little late.

This has been an example using classical processing. But by using the incremental function we can simulate online processing as follows:

```
## Initialize parameters for the loop
last.res <- NULL
res <- NULL
nread <- 250
numIter <- n%/%nread
iter <- seq(1, nread * numIter, nread)

## Calculate anomalies
for(i in iter) {
  # read new data
  newRow <- df[i:(i + nread - 1),]
  # calculate if it's an anomaly
  last.res <- IpSdEwma(
    data = newRow$value,
    n.train = 5,
    threshold = 0.01,
    l = 3,
    last.res = last.res$last.res
  )
  # prepare the result
  if(!is.null(last.res$result)){
    res <- rbind(res, cbind(newRow[(nread - nrow(last.res$result) + 1):nread,], last.res$result))
  }
}
```

In the same way we can view the results (Figure reffig:result2) by writing:

```
library(ggplot2)

ggplot(res, aes(timestamp, value)) +
      geom_line(na.rm=TRUE, color = "blue") +
```

```
    ggtitle("Online SD-EWMA Anomaly Detector") +
    xlab("Time") + ylab("Value") +
    theme(plot.title = element_text(lineheight=.8, face="bold", size = 20)) +
    theme(text = element_text(size=18)) +
    geom_point(data = res[res$is.anomaly == TRUE,],aes(colour = factor(is.anomaly))) +
    theme(legend.position="none")
```

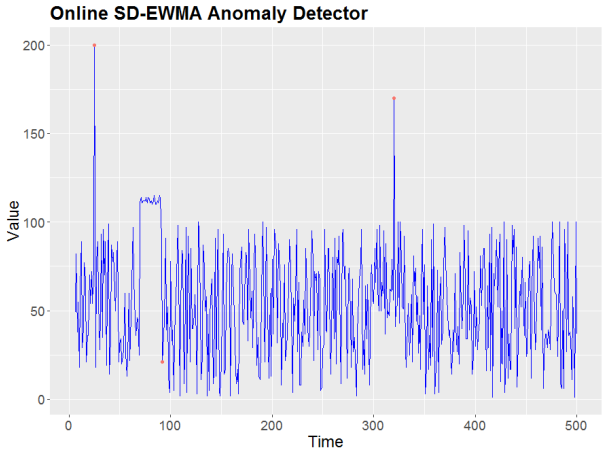


Figure 6: Online Sd-Ewma Anomaly detector results

4 Summary

In this vignette we introduce the **otsad** package, which implements five recent algorithms for online anomaly detection. To introduce the context and motivation for this work we present the new challenges faced by time-series anomaly detection techniques with the introduction of the need for online processing. As explained in section 2, the package proposes three algorithms based on the most recent prediction techniques (derived from EWMA) and two, based on moving window techniques (derived from KNN-CAD). Likewise, we cover the problem of the stationarity of time-series since two of the proposed algorithms have been designed to work in stationary environments and the other three can work in non-stationary environments. Finally, all algorithms can be used in two different scenarios: classical processing and incremental processing.

About the future extensions of this package, there are several aspects that may be addressed in the future. For example, functions for displaying the results. In addition, a new algorithm and a new metric are already being developed for the evaluation of detector quality. Several pre-labelled time-series data sets will also be included.

References

[1] E. Burnaev and V. Ishimtsev. Conformalized density- and distance-based anomaly detection in time-series data. *ArXiv e-prints*, August 2016.

[2] K. M. Carter and W. W. Streilein. Probabilistic reasoning for streaming anomaly detection. In *2012 IEEE Statistical Signal Processing Workshop (SSP)*, pages 377–380, Aug 2012.

[3] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009.

[4] Manish Gupta, Jing Gao, Charu C. Aggarwal, and Jiawei Han. Outlier detection for temporal data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 26(9):2250–2267, 9 2014.

[5] Victoria Hodge and Jim Austin. A survey of outlier detection methodologies. *Artif. Intell. Rev.*, 22(2):85–126, October 2004.

[6] V. Ishimtsev, I. Nazarov, A. Bernstein, and E. Burnaev. Conformal k-NN Anomaly Detector for Univariate Data Streams. *ArXiv e-prints*, June 2017.

[7] Haider Raza, Girijesh Prasad, and Yuhua Li. Ewma model based shift-detection methods for detecting covariate shifts in non-stationary environments. *Pattern Recogn.*, 48(3):659–669, March 2015.

[8] Yang Zhang, N. Meratnia, and P. Havinga. Outlier detection techniques for wireless sensor networks: A survey. *Commun. Surveys Tuts.*, 12(2):159–170, April 2010.