ANOMALY DETECTION AND OUTLIER ANALYSIS

Youssouph Cissokho[1], Soufiane Fadel[1], Richard Millson[1], Razieh Pourhasan[1], Patrick Boily[1,2,3]

Abstract

With the advent of automatic data collection, it is now possible to store and process large troves of data. There are technical issues associated to massive data sets, such as the speed and efficiency of analytical methods, but there are also problems related to the detection of anomalous observations and the analysis of outliers.

Extreme and irregular values behave very differently from the majority of observations. For instance, they can represent criminal attacks, fraud attempts, targeted attacks, or data collection errors. As a result, anomaly detection and outlier analysis play a crucial role in cybersecurity, quality control, etc. [1, 3, 4]. The (potentially) heavy human price and technical consequences related to the presence of such observations go a long way towards explaining why the topic has attracted attention in recent years.

This report contains a review of various detection methods, with particular attention paid to both supervised and unsupervised methods, as well as an application to time series data and a project suggestion (comparative analysis of various algorithms applied to 5 real-world datasets).

Keywords

Anomaly detection, outlier analysis.

Funding Acknowledgement

This report was funded by a University of Ottawa grant to develop teaching material in French (2019-2020). It was subsequently translated into English to produce the current document.

[1]Department of Mathematics and Statistics, University of Ottawa, Ottawa
[2]Data Action Lab, Ottawa
[3]Idlewyld Analytics and Consulting Services, Wakefield, Canada
Email: pboily@uottawa.ca

Contents

1. Introduction

Isaac Asimov, the prolific American author, once wrote that

> The most exciting phrase to hear [...], the one that heralds the most discoveries, is not "Eureka!" but "That's funny...".

However, anomalous observations are not only harbingers of great scientific discoveries – unexpected observations can spoil analyses or be indicative of the presence of issues related to data collection or data processing.

Either way, it becomes imperative for decision-makers and analysts to establish anomaly detection protocols, and to identify strategies to deal with such observations.

1.1 Basic Notions and Overview

Outlying observations are data points which are **atypical** in comparison to the unit's remaining features (*within-unit*), or in comparison to the measurements for other units (*between-units*), or as part of a collective subset of observations. Outliers are thus observations which are **dissimilar to other cases** or which contradict **known dependencies** or rules.[1]

Observations could be anomalous in one context, but not in another. Consider, for instance, an adult male who is 6-foot tall. Such a man would fall in the 86th percentile among Canadian males [23], which, while on the tall side, is not unusual; in Bolivia, however, the same man would land in the 99.9th percentile [23], which would mark him as extremely tall and quite dissimilar to the rest of the population.[2]

A common mistake that analysts make when dealing with outlying observations is to remove them from the dataset without carefully studying whether they are **influential data points**, that is, observations whose absence leads to **markedly different** analysis results.

When influential observations are identified, remedial measures (such as data transformation strategies) may need to be applied to minimize any undue effect. Note that outliers may be influential, and influential data points may be outliers, but the conditions are neither necessary nor sufficient.

Anomaly Detection

By definition, anomalies are **infrequent** and typically shrouded in **uncertainty** due to their relatively low numbers, which makes it difficult to differentiate them from banal **noise** or **data collection errors**.

Furthermore, the boundary between normal and deviant observations is usually **fuzzy**; with the advent of e-shops, for instance, a purchase which is recorded at 3AM local time does not necessarily raise a red flag anymore.

When anomalies are actually associated to **malicious activities**, they are more than often **disguised** in order to blend in with normal observations, which obviously complicates the detection process.

Numerous methods exist to identify anomalous observations; **none of them are foolproof** and judgement must be used. Methods that employ graphical aids (such as boxplots, scatterplots, scatterplot matrices, and 2D tours) to identify outliers are particularly easy to implement, but a low-dimensional setting is usually required for ease of interpretability.

Analytical methods also exist (using Cooke's or Mahalanobis' distances, say), but in general some additional level of analysis must be performed, especially when trying to identify influential points (*cf.* **leverage**).

With small datasets, anomaly detection can be conducted on a case-by-case basis, but with large datasets, the temptation to use **automated detection/removal** is strong – care must be exercised before the analyst decides to go down that route.[3]

In the early stages of anomaly detection, **simple data analyses** (such as descriptive statistics, 1- and 2-way tables, and traditional visualisations) may be performed to help identify anomalous observations, or to obtain insights about the data, which could eventually lead to modifications of the analysis plan.

Outlier Tests

How are outliers *actually* detected? Most methods come in one of two flavours: **supervised** and **unsupervised** (we will discuss those in detail in later sections).

Supervised methods use a historical record of **labeled** (that is to say, previously identified) anomalous observations to build a **predictive classification or regression model** which estimates the probability that a unit is anomalous; domain expertise is required to tag the data. Since anomalies are typically **infrequent**, these models often also have to accommodate the **rare occurrence problem**.[4]

Unsupervised methods, on the other hand, use no previously labeled information or data, and try to determine if an observation is an outlying one solely by comparing its behaviour to that of the other observations.

[1] Outlying observations may be anomalous along any of the individual variables, or in combination.

[2] Anomaly detection points towards interesting questions for analysts and subject matter experts: in this case, why is there such a large discrepancy in the two populations?

[3] This stems partly from the fact that once the "anomalous" observations have been removed from the dataset, previously "regular" observations can become anomalous in turn in the smaller dataset; it is not clear when that runaway train will stop.

[4] Supervised models are built to minimize a cost function; in default settings, it is often the case that the mis-classification cost is assumed to be symmetrical, which can lead to technically correct but useless solutions. For instance, the vast majority (99.999+%) of air passengers emphatically do not bring weapons with them on flights; a model that predicts that no passenger is attempting to smuggle a weapon on board a flight would be 99.999+% accurate, but it would miss the point completely.

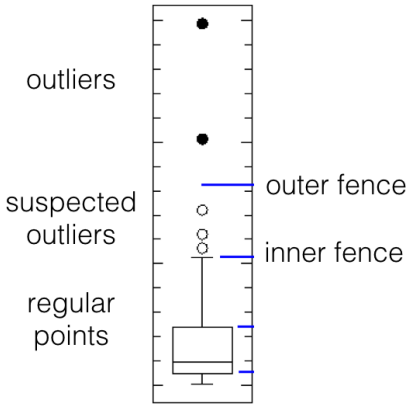


Figure 1. Tukey's boxplot test; suspected outliers are marked by white disks, outliers by black disks.

The following traditional methods and tests of outlier detection fall into this category:[5]

- Perhaps the most commonly-used test is **Tukey's boxplot test**; for normally distributed data, regular observations typically lie between the **inner fences**

$$Q_1 - 1.5(Q_3 - Q_1) \quad \text{and} \quad Q_3 + 1.5(Q_3 - Q_1).$$

  Suspected outliers lie between the inner fences and their respective **outer fences**

$$Q_1 - 3(Q_3 - Q_1) \quad \text{and} \quad Q_3 + 3(Q_3 - Q_1).$$

  Points beyond the outer fences are identified as **outliers** (Q_1 and Q_3 represent the data's 1st and 3rd quartile, respectively; see Figure 1).
- The **Grubbs test** is another univariate test, which takes into consideration the number of observations in the dataset. Let x_i be the value of feature X for the i^{th} unit, $1 \leq i \leq N$, let $(\overline{x}, s_x)$ be the mean and standard deviation of feature X, let α be the desired significance level, and let $T(\alpha, N)$ be the critical value of the Student t-distribution at significance $\alpha/2N$. Then, the i^{th} unit is an **outlier along feature** X if

$$|x_i - \overline{x}| \geq \frac{s_x(N-1)}{\sqrt{N}} \sqrt{\frac{T^2(\alpha, N)}{N - 2 + T^2(\alpha, N)}}.$$

- Other common tests include:
  - the **Dixon** Q **test**, which is used in the experimental sciences to find outliers in (extremely) small datasets – it is of dubious validity;
  - the **Mahalanobis distance**, which is linked to the leverage of an observation (a measure of influence), can also be used to find multi-dimensional outliers, when all relationships are linear (or nearly linear);
  - the **Tietjen-Moore** test, which is used to find a specific number of outliers;
  - the **generalized extreme studentized deviate** test, if the number of outliers is unknown;
  - the **chi-square** test, when outliers affect the goodness-of-fit, as well as
  - DBSCAN and other clustering-based outlier detection methods.

[5] Note that **normality** of the underlying data is an assumption for most tests; how robust these tests are against departures from this assumption depends on the situation.

Visual Outlier Detection

The following three (simple) examples illustrate the principles underlying visual outlier and anomaly detection.

Example 1. On a specific day, the height of several plants in a nursery are measured. The records also show each plant's age (the number of weeks since the seed has been planted).

Histograms of the data are shown in Figure 2 (age on the left, height on the middle).

Very little can be said about the data at that stage: the age of the plants (controlled by the nursery staff) seems to be somewhat haphazard, as does the response variable (height). A scatter plot of the data (rightmost chart in Figure 2), however, reveals that growth is strongly correlated with age during the early period of a plant's life for the observations in the dataset; points clutter around a linear trend. One point (in yellow) is easily identified as an **outlier**. There are (at least) two possibilities: either that measurement was botched or mis-entered in the database (representing an invalid entry), or that one specimen has experienced unusual growth (outlier). Either way, the analyst has to investigate further.

Example 2. A government department has 11 service points in a jurisdiction. Service statistics are recorded: the monthly average arrival rates per teller and monthly average service rates per teller for each service point are available.

A scatter plot of the service rate per teller (y axis) against the arrival rate per teller (x axis), with linear regression trend, is shown in the leftmost chart in Figure 3. The trend is seen to inch upwards with increasing x values.

A similar chart, but with the left-most point removed from consideration, is shown in the middle chart of Figure 3. The trend still slopes upward, but the fit is significantly improved, suggesting that the removed observation is unduly **influential** (or anomalous) – a better understanding of the relationship between arrivals and services is afforded if it is set aside.

Any attempt to fit that data point into the model must take this information into consideration. Note, however, that influential observations depend on the analysis that is ultimately being conducted – a point may be influential for one analysis, but not for another.

Example 3. Measurements of the length of the appendage of a certain species of insect have been made on 71 individuals. Descriptive statistics have been computed; the results are shown in Figure 4.

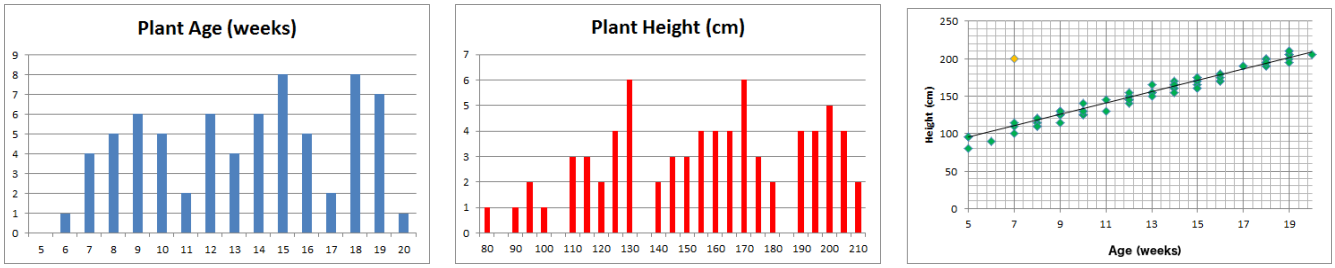


Figure 2. Summary visualisations for an (artificial) plant dataset: age distribution (left), height distribution (middle), height vs. age, with linear trend (right).

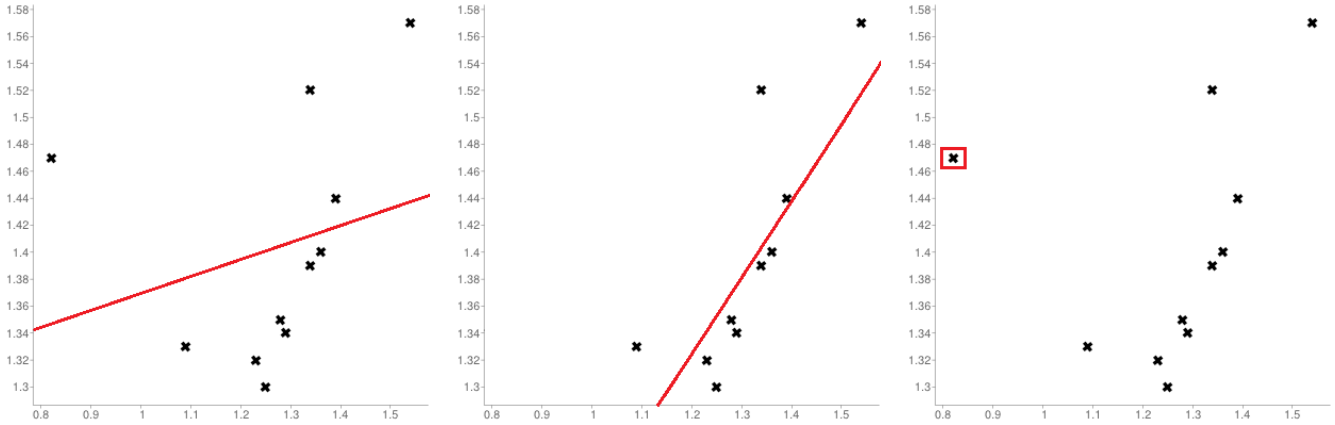


Figure 3. Visualisations for an (artificial) service point dataset: trend for 11 service points (left), trend for 10 service points (middle), influential observations (right).

Analysts who are well-versed in statistical methods might recognize the tell-tale signs that the distribution of appendage lengths is likely to be asymmetrical (since the skewness is non-negligible) and to have a "fat" tail (due to the kurtosis being commensurate with the mean and the standard deviation, the range being so much larger than the interquartile range, and the maximum value being so much larger than the third quartile).

The mode, minimum, and first quartile values belong to individuals without appendages, so there appears to be at least two sub-groups in the population (perhaps split along the lines of juveniles/adults, or males/females). The maximum value has already been seen to be quite large compared to the rest of the observations, which at first suggests that it might belong to an **outlier**.

The histogram of the measurements, however, shows that there are 3 individuals with very long appendages (see right-most chart in Figure 4): it now becomes plausible for these anomalous entries to belong to individuals from a different species altogether who were **erroneously added** to the dataset. This does not, of course, constitute a proof of such an error, but it raises the possibility, which is often the best that an analyst can do in the absence of subject matter expertise.

This traditional approach to anomaly detection fails for high-dimensional datasets, however, and a fundamentally different approach is advocated.

1.2 Anomaly Detection as a Statistical Learning Problem

Fraudulent behaviour is not always easily identifiable, even after the fact. Credit card fraudsters, for instance, will try to disguise their transactions as regular and banal, rather than as outlandish; to fool human observers into confusing what is merely **plausible** with what is **probable** (or at least, **not improbable**).

At its most basic level, anomaly detection is a problem in **applied probability**: if I denotes what is known about the dataset (behaviour of individual observations, behaviour of observations as a group, anomalous/normal verdict for a number of similar observations, etc.), is

$$P(\text{obs. is anomalous} \mid I) > P(\text{obs. is normal} \mid I)?$$

Anomaly detection models usually assume **stationarity for normal observations**, which is to say, that the underlying mechanism that generates data does not change in a substantial manner over time, or, if it does, that its rate of change or cyclicity is known.

For time series data, this means that it may be necessary to first perform trend and seasonality extraction.

Example 4. Supply chains play a crucial role in the transportation of goods from one part of the world to another. As the saying goes, "a given chain is only as strong as its weakest link" – in a multi-modal context, comparing the various

Appendage length (mm)	
Mean	10.35
Standard Deviation	16.98
Kurtosis	16.78
Skewness	4.07
Minimum	0
First Quartile	0
Median	8.77
Third Quartile	10.58
Maximum	88
Range	88
Interquartile Range	10.58
Mode	0
Count	71

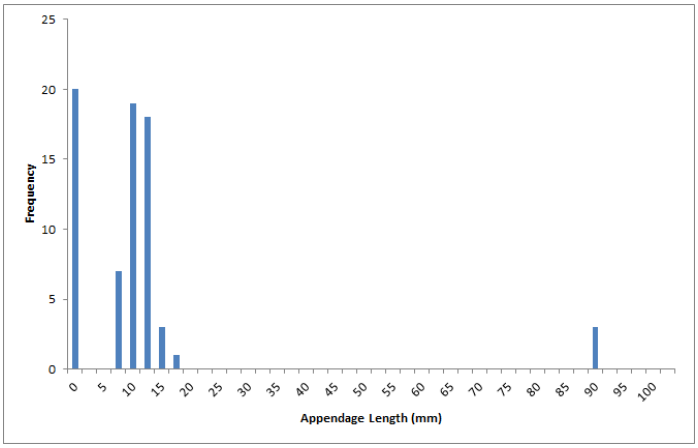


Figure 4. Summary and visualisation for an (artificial) appendage length dataset: descriptive statistics (left), appendage length distribution (right).

transportation segments is far from an obvious endeavour: if shipments departing Shanghai in February 2013 took two more days, on average, to arrive in Vancouver than those departing in July 2017, can it be said with any certainty that the shipping process has improved in the intervening years? Are February departures always slower to cross the Pacific Ocean?

The seasonal variability of performance is relevant to supply chain monitoring; the ability to quantify and account for the severity of its impact on the data is thus of great interest.

One way to tackle this problem is to produce an **index** to track container transit times. This index should depict the **reliability** and the **variability** of transit times but in such a way as to be able to allow for performance comparison between differing time periods.

To simplify the discussion, assume that the ultimate goal is to compare quarterly and/or monthly performance data, irrespective of the transit season, in order to determine how well the network is performing on the *Shanghai → Port Metro Vancouver/Prince Rupert → Toronto* corridor, say.

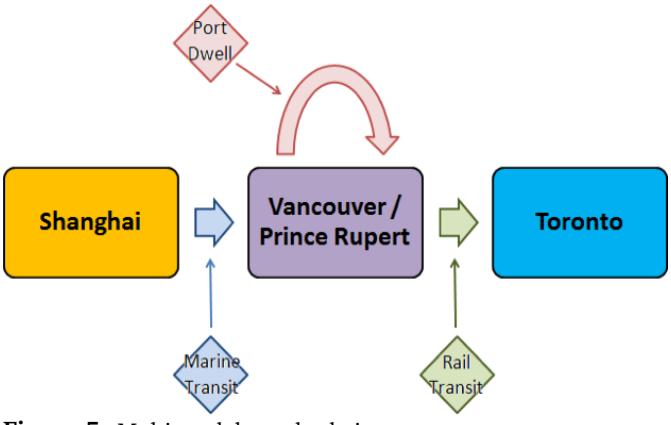


Figure 5. Multi-modal supply chain.

The supply chain under investigation has Shanghai as the point of origin of shipments, with Toronto as the final destination; the containers enter the country either through Vancouver or Prince Rupert. Containers leave their point of origin by boat, arrive and dwell in either of the two ports before reaching their final destination by rail.

For each of the three segments (Marine Transit, Port Dwell, Rail Transit), the data consists of the monthly empirical distribution of transit times, built from sub-samples (assumed to be randomly selected and fully representative) of all containers entering the appropriate segment.

Each segment's performance is measured using **fluidity indicators**, which are computed using various statistics of the transit/dwelling time distributions for each of the supply chain segments, such as:

Reliability Indicator (RI) – the ratio of the 95th percentile to the 5th percentile of transit/dwelling times (a high RI indicates high volatility, whereas a low RI (≈ 1) indicates a reliable corridor);

Buffer Index (BI) – the ratio of the positive difference between the 95th percentile and the mean, to the mean. A small BI (≈ 0) indicates only slight variability in the upper (longer) transit/dwelling times; a large BI indicates that the variability of the longer transit/dwelling times is high, and that outliers might be found in that domain;

Coefficient of Variation (CV) – the ratio of the standard deviation of transit/dwelling times to the mean transit/dwelling time.

The time series of monthly indicators (which are derived from the monthly transit/dwelling time distributions in each segment) are then **decomposed** into their

- trend;
- seasonal component (seasonality, trading-day, moving-holiday), and

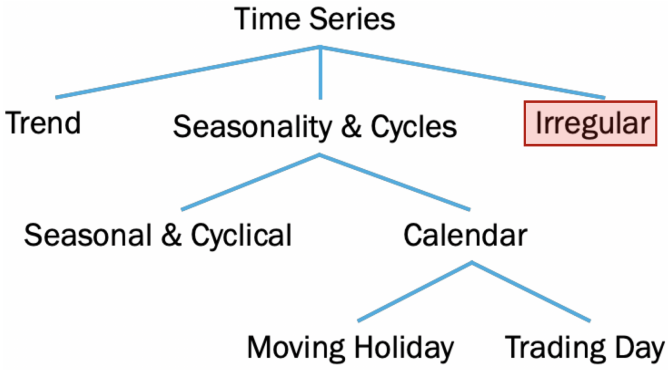


Figure 6. Conceptual time series decomposition; potential anomalous behaviour should be searched for in the irregular component.

- irregular component.

The trend and the seasonal components provide the **expected behaviour** of the indicator time series;[6] the irregular component arise as a consequence of supply chain **volatility**. A high irregular component at a given time point indicates a poor performance against expectations for that month, which is to say, an **anomalous observation**.

In general, the decomposition follows a model which is

- multiplicative;
- additive, or
- pseudo-additive.

The choice of a model is driven by data behaviour and choice of assumptions; the X12 model automates some of the aspects of the decomposition, but manual intervention and diagnostics are still required.[7] The additive model, for instance, assumes that:

1. the seasonal component S_t and the irregular component I_t are independent of the trend T_t;
2. the seasonal component S_t remains stable from year to year; and
3. there is no seasonal fluctuation: $\sum_{j=1}^{12} S_{t+j} = 0$.

Mathematically, the model is expressed as:

$$O_t = T_t + S_t + I_t$$

All components share the same dimensions and units. After seasonality adjustment,the seasonality adjusted series is:

$$SA_t = O_t - S_t = T_t + I_t$$

[6]Before carrying out seasonal adjustment, it is important to identify and pre-adjust for structural breaks (using the Chow test, for instance), as their presence can give rise to severe distortions in the estimation of the Trend and Seasonal effects. Seasonal breaks occur when the usual seasonal activity level of a particular time reporting unit changes in subsequent years. Trend breaks occurs when the trend in a data series is lowered or raised for a prolonged period, either temporarily or permanently. Sources of these breaks may come from changes in government policies, strike actions, exceptional events, inclement weather, etc.

[7]X12 is implemented in SAS and R, among other platforms.

The multiplicative and pseudo-additive models are defined in similar ways (consult [27–31] for details).[8]

The data decomposition/preparation process is illustrated with the 40-month time series of marine transit CVs from 2010-2013, whose values are shown in Figure 7. The size of the peaks and troughs seems fairly constant with respect to the changing trend; the SAS implementation of X12 agrees with that assessment and suggests the additive decomposition model, with no need for further data transformations.

The diagnostic plots are shown in Figure 8: the CV series is prior-adjusted from the beginning until OCT2010 after the detection of a level shift. The SI (Seasonal Irregular) chart shows that there are more than one irregular component which exhibits volatility. The adjusted series is shown below in Figure 9; the trend and irregular components are also shown separately for readability. It is on the irregular component that detection anomaly would be conducted.

The last example shows the importance of domain understanding and data preparation to the anomaly detection process. Given that the vast majority of observations in a general problem are typically "normal", another conceptually important approach is to view anomaly detection as a **rare occurrence learning** classification problem or as a **novelty detection** data stream problem (these problems will be tackled in other data science reports of this series).

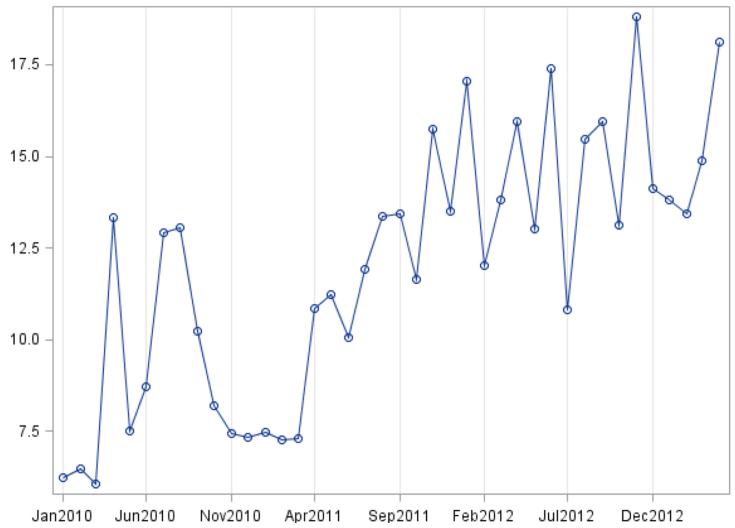
Either way, while there a number of strategies that use regular classification/clustering algorithms for anomaly detection, they are rarely successful unless they are adapted or modified for the anomaly detection context.

Basic Concepts

A generic system (such as the monthly transit times from the previous subsection, say) may be realized in **normal** states or in **abnormal** states. Normality, perhaps counterintuitively, is not confined to finding the most likely state, however, as infrequently occurring states could still be normal or plausible under some interpretation of the system.

As the authors of [12] see it, a system's states are the results of processes or behaviours that follow certain natural rules and broad principles; the observations are a manifestation of these states. Data, in general, allows for inferences to be made about the underlying processes, which can then be tested or invalidated by the collection of additional data. When the inputs are perturbed, the corresponding outputs

[8]The simplest way to determine whether to use multiplicative or additive decomposition is by graphing the time series. If the size of the seasonal variation increases/decreases over time, multiplicative decomposition should be used. On the other hand, if the seasonal variation seems to be constant over time, additive model should be used. A pseudo-additive model should be used when the data exhibits the characteristics of the multiplicative series, but parameter values are close to zero.



Estimation Summary	
For Variable hours_CV	
Number of Observations	40
Number of Residuals	27
Number of Parameters Estimated	3
Variance Estimate	5.6E-02
Standard Error Estimate	2.4E-01
Standard Error of Variance	1.5E-02
Log likelihood	0.4658
Transformation Adjustment	-69.5685
Adjusted Log likelihood	-69.1027
AIC	144.2053
AICC (F-corrected AIC)	145.2488
Hannan Quinn	145.3613
BIC	148.0928

Results of Automatic Transformation Selection	
For Variable hours_CV	
AICC (with aicdiff=-2.00) prefers	No transformation
Adjustment will be	Additive

Figure 7. Monthly marine transit CVs and estimation summary.

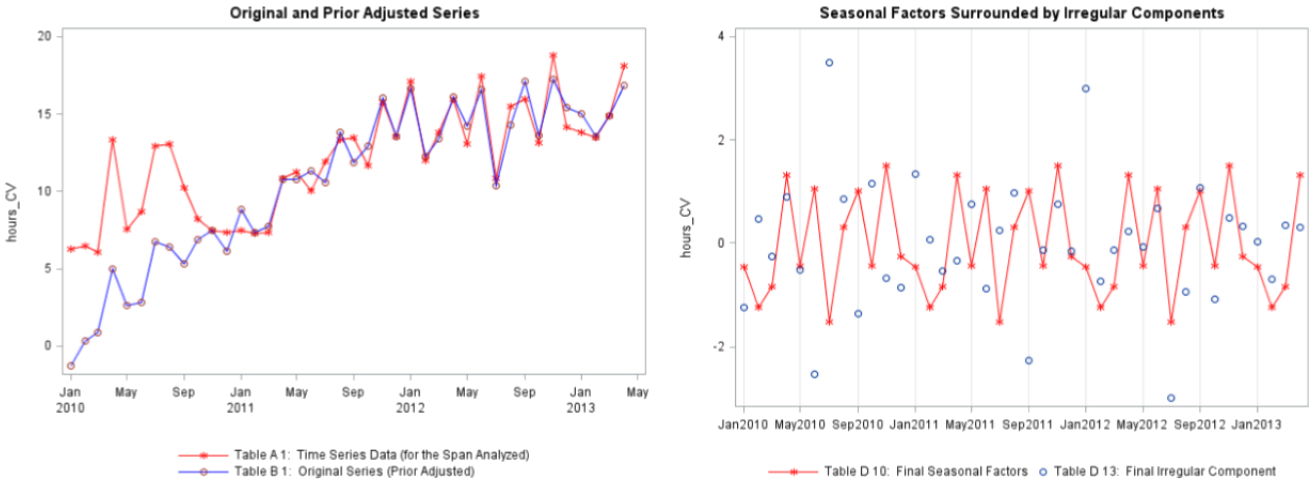


Figure 8. Diagnostic plots. Note that the analysis of a time series starts with estimation of the effects of festivals and trading days. These pre-calculated estimates are then used for prior adjustment of the series. The prior adjusted original series is subsequently analyzed using the seasonal adjustment.

are likely to be perturbed as well; if anomalies arise from perturbed processes, being able to identify when the process is abnormal, that is to say, being able to capture the various normal and abnormal processes, may lead to useful anomaly detection.

Any **supervised** anomaly detection algorithm requires a training set of historical labeled data (which may be costly to obtain) on which to build the prediction model, and a testing set on which to evaluate the model's performance in terms of **True Positives** (TP, detected anomalies that actually arise from process abnormalities); **True Negatives** (TN, predicted normal observations that indeed arise from normal processes); **False Positives** (FP, detected anomalies corresponding to regular processes), and **False Negatives** (FN, predicted normal observations that are in fact the product of an abnormal process).

		Predicted Class	
		Normal	Anomaly
Actual Class	Normal	*TN*	*FP*
	Anomaly	*FN*	*TP*

As discussed previously, the rare occurrence problem makes optimizing for maximum **accuracy**

$$a = \frac{\mathrm{TN} + \mathrm{TP}}{\mathrm{TN} + \mathrm{TP} + \mathrm{FN} + \mathrm{FP}}$$

a losing strategy; instead, algorithms attempt to minimize the FP rate and the FN rate under the assumption that the cost of making a false negative error could be substantially higher than the cost of making a false positive error.

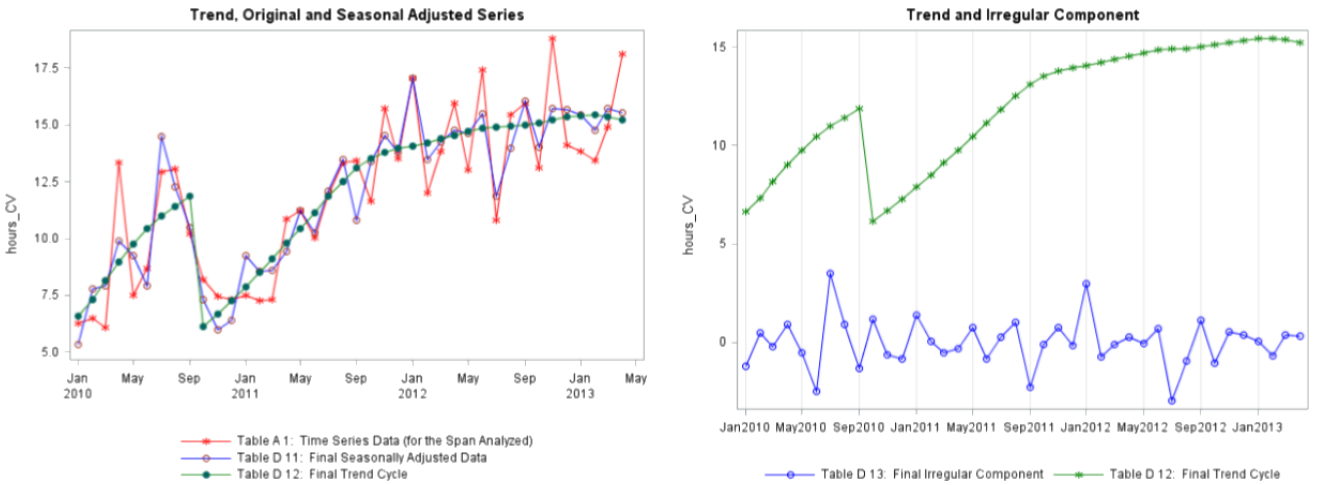


Figure 9. Adjusted time series components plots.

Assume that for a testing set with $d = \text{FN} + \text{TP}$ true outliers, an anomaly detection algorithm identifies $m = \text{FP} + \text{TP}$ suspicious observations, of which $n = \text{TP}$ are known to be true outliers. Performance evaluation in this context is often measured using:

Precision – the proportion of true outliers among the suspicious observations

$$p = \frac{n}{m} = \frac{\text{TP}}{\text{FP} + \text{TP}};$$

when most of the points identified by the algorithm are true outliers, $p \approx 1$;

Recall – the proportion of true outliers detected by the algorithm

$$r = \frac{n}{d} = \frac{\text{TP}}{\text{FN} + \text{TP}};$$

when most of the true outliers are identified by the algorithm, $r \approx 1$;

F_1**—Score** – the harmonic mean of the algorithm's precision and its recall

$$F_1 = \frac{2pr}{p + r} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}};$$

one drawback of precision, recall, and the F_1—score is that they do not incorporate TN in the evaluation process, but this is unlikely to be problematic as regular observations that are correctly seen as unsuspicious are not usually the observations of interest.[9];

Example 5. Consider a test dataset with 5000 observations, 100 of which are anomalous. An algorithm which predicts all observations to be anomalous would score $a = p = 0.02$, $r = 1$, and $F_1 \approx 0.04$, whereas an algorithm that detects 10 of the true outliers would score $r = 0.1$ (the other values would change according to the TN and FN counts).

[9]Nevertheless, the analyst for whom the full picture is important might want to further evaluate the algorithm with the help of the **Matthews Correlation Coefficient** [32] or the **specificity** $s = \frac{\text{TN}}{\text{FP}+\text{TN}}$.

Another supervised approach is to estimate the relative abnormality of various observations: it is usually quite difficult to estimate the probability that an observation $\mathbf{x}_1$ is anomalous with any certainty, but it might be possible to determine that it is more likely to be anomalous than another observation $\mathbf{x}_2$, say (denoted by $\mathbf{x}_1 \succeq \mathbf{x}_2$).

This paradigm allows the suspicious observations to be ranked; let $k_i \in \{1, \ldots, m\}$ be the rank of the i^{th} true outlier, $i \in \{1, \ldots, n\}$, in the sorted list of suspicious observations

$$\mathbf{x}_1 \succeq \mathbf{x}_{k_1} \succeq \cdots \succeq \mathbf{x}_{k_i} \succeq \cdots \mathbf{x}_{k_n} \succeq \mathbf{x}_m;$$

the **rank power** of the algorithm is

$$RP = \frac{n(n+1)}{2\sum_{i=1}^{n} k_i}.$$

When the d actual anomalies are ranked in (or near) the top d suspicious observations, $\text{RP} \approx 1$.

Rank power is well-defined only when $m \geq d$; as with most performance evaluation metrics, a single raw number is meaningless – it is in comparison with the performance of other algorithms that it is most useful.

On the **unsupervised** front, where anomalous/normal labels are not known or used, if anomalies are those observations that are dissimilar to other observations, and if clusters represent groupings of similar observations, then observations that do not naturally fit into a cluster could be potential anomalies (see Figure 10).

There are a number of challenges, not the least of which being that most clustering algorithms do not recognize potential outliers (DBSCAN is a happy exception) and that some appropriate measure of similarity/dissimilarity of observations has to be agreed upon (different measures could lead to different cluster assignments).

Finally, it is worth mentioning that the definitions of terms like **normal** and **anomalous** are kept purposely vague, to allow for flexibility.

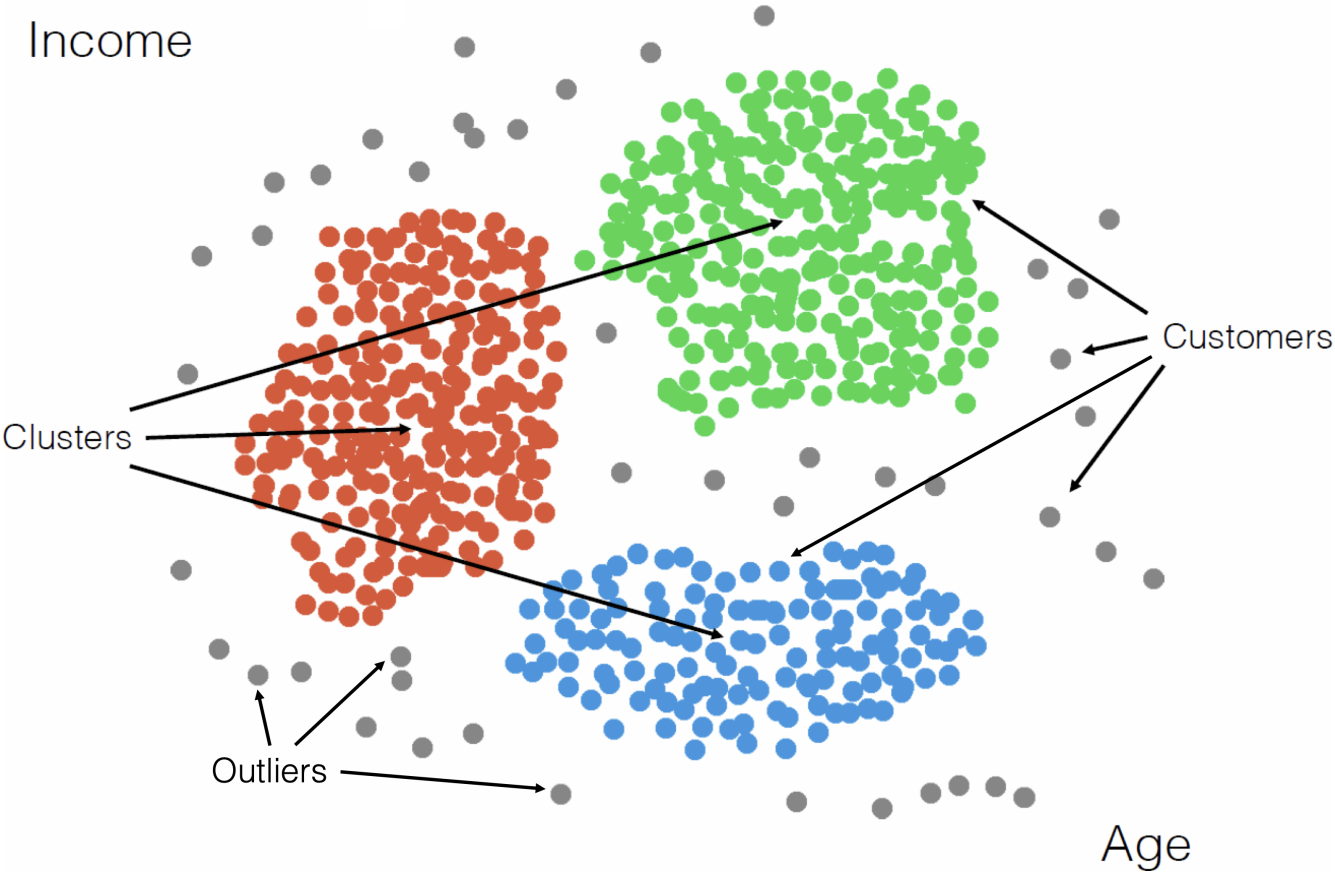


Figure 10. Clusters of customers (red, green, blue) and potential anomalies/outliers (grey) in an artificial dataset.

1.3 Suggested References

The main references that were consulted in the preparation fo this report are:

- Aggarwal, C.C. [2017], *Outlier Analysis* (2nd ed.), Springer [1]
- Mehrotra, K.G., Mohan, C.K., Huang, H. [2017], *Anomaly Detection Principles and Algorithms*, Springer [12]

Other good survey documents include Chandola, Banerjee, and Kumar's *Outlier detection: a survey* [21], and Hodge and Austin's *A survey of outlier detection methodologies* [22].

Specific methods and approaches are the focus of other papers: [2,8,41] (high-dimensional data), [7] (DOBIN), [9] (outlier ensembles), [17,24] (isolation forest), [18,25] (DBSCAN), [39] (LOF), [37,38,40,42,43] (subspace method), [35] (time series data).

On the practical side of things, we would be remiss if we didn't mention Arora's *An Awesome Tutorial to Learn Outlier Detection in Python using PyOD Library* [13]; note that there is a plethora of quality tutorials for anomaly detection in the programming language of your choice online.

1.4 Structure and Organization

In this report, we aim to provide a better understanding of anomaly detection, outlier techniques and some of the field's challenges. We also provide some step-by step applications of the techniques over real-life examples.

The purpose of Section 2 is to provide a comprehensive and structured overview of different methods of anomaly detection and outliers analysis in the **quantitative case**; the **qualitative case** is tackled in Section 3. In these sections, particular attention is paid to supervised and unsupervised methods, including **distance-based** and **density-based** methods.

Section 4 is dedicated to approaches for **large data sets**, known as HDLSS (high dimension low sample size) where the sample size n is smaller than the dimension p. Outlier detection in HDLSS dataset is even more challengin, mostly due to the curse of dimensionality. **Feature bagging**, **ensemble methods**, and various **dimension reduction** methods are also discussed.

Finally, in Sections 5 and 6, we provide detailed practical, real-life examples of anomaly detection in stock exchange data, and suggest project work for airline data and fatal driving collision data.

2. Quantitative Methods of Anomaly Detection

Quantitative methods are divided into distance- and density-based methods.

2.1 Distance-Based Methods

In order to determine whether an observation is anomalous or not, it must be compared to a set of other observations (anomalies are relative, not absolute). In the **distance-based context**, one natural way to compare observations is to consider their distance from one another, with increasing distance from the others being increasingly suggestive of anomalous status.

This approach works both in continuous and discrete cases, as long as a **distance function** or a **pre-computed table of pair-wise distances** between observations is given.

The choice of which sets of points to use in this comparison distinguishes the different distance-based algorithms.

This discussion begins with the introduction of some notation. Let $D \subset \mathbb{R}^n$ be an n-dimensional data set, $\mathbf{p}, \mathbf{q} \in D$, $P \subset D$ be a subset of D, and $d : D \times D \to \mathbb{R}$ gives the distance between $\mathbf{p}$ and $\mathbf{q}$, written $d(\mathbf{p}, \mathbf{q})$.

An anomaly detection algorithm provides a function $a : D \to \mathbb{R}$ that describes how anomalous a given point is. This induces an ordering on the points of D: if $a(\mathbf{p}) < a(\mathbf{q})$ for $\mathbf{p}, \mathbf{q} \in D$, then $\mathbf{p}$ is **less anomalous** than $\mathbf{q}$.

It could be necessary to define a threshold beyond which a point is considered anomalous; if $\alpha \in \mathbb{R}$ is such a threshold, then any $\mathbf{p} \in D$ is **absolutely anomalous** if $a(\mathbf{p}) > \alpha$.

Similarity Measures

A **similarity measure** is a real-valued function that describes the similarity between two objects. A common construction is to define the similarity w between two points $\mathbf{p}, \mathbf{q}$ as

$$w(\mathbf{p}, \mathbf{q}) = \frac{1}{1 + d(\mathbf{p}, \mathbf{q})}, \text{ for some distance } d,$$

so that $w \to 1$ as $d \to 0$, and $w \to 0$ as $d \to \infty$.

A similarity measure can also be constructed between probability distributions. Let X and Y be two n-dimensional random vectors of (possibly) different distribution with probability mass/density functions (p.m.f./p.d.f.) f_X and f_Y, respectively. Let Ω be their shared domain. For discrete random variables, the **Hellinger distance** is defined by

$$H(X, Y) = \left(1 - \sum_{\mathbf{z} \in \Omega} \sqrt{f_X(\mathbf{z}) f_Y(\mathbf{z})}\right)^{1/2};$$

for continuous random variables, it is defined by

$$H(X, Y) = \left(1 - \int_\Omega \sqrt{f_X(\mathbf{z}) f_Y(\mathbf{z})}\, d\mathbf{z}\right)^{1/2}.$$

If $f_X = f_Y$ (or $f_X = f_Y$ almost everywhere in the continuous case, that is, except over a countable set), then

$$\sum_\Omega \sqrt{f_x f_Y} = 1 \quad \text{or} \quad \int_\Omega \sqrt{f_X f_Y}\, d\mathbf{z} = 1$$

and $H(X, Y) = 0$. The fact that $H(X, Y) \in [0, 1]$ is a consequence of Cauchy's inequality, with $f_X^* = \sqrt{f_X}$ and $f_Y^* = \sqrt{f_Y}$:

$$\begin{aligned} 0 \le \int_\Omega \sqrt{f_X f_Y}\, d\mathbf{z} &= \int_\Omega f_X^* f_Y^*\, d\mathbf{z} \\ &\le \left(\int_\Omega |f_X^*|^2\, d\mathbf{z}\right)^{1/2} \left(\int_\Omega |f_Y^*|^2\, d\mathbf{z}\right)^{1/2} \\ &= \left(\int_\Omega f_X\, d\mathbf{z}\right)^{1/2} \left(\int_\Omega f_Y\, d\mathbf{z}\right)^{1/2} = 1; \end{aligned}$$

(a similar argument holds for discrete random variables).

Recall that the covariance matrices Σ_X and Σ_Y are $n \times n$-matrices whose (i, j)-th entries are the covariance between the i-th and j-th positions of X and Y, respectively. Given a collection of identically distributed samples, these covariance matrices can be estimated.

We can also consider a single point $\mathbf{p}$ to represent probability distribution. In that case, the Hellinger distance between that point and any other distribution with mean μ and covariance matrix Σ can be studied using the framework above, using the **Mahalanobis distance**:

$$M(\mathbf{p}) = \sqrt{(\mathbf{p} - \mu)^\top \Sigma^{-1} (\mathbf{p} - \mu)}.$$

Alternatively, if $\mathbf{p}$ and $\mathbf{q}$ are drawn from the same distribution with covariance Σ, then the Mahalanobis distance is a dissimilarity measure between $\mathbf{p}$ and $\mathbf{q}$:

$$d_M(\mathbf{p}, \mathbf{q}) = \sqrt{(\mathbf{p} - \mathbf{q})^\top \Sigma^{-1} (\mathbf{p} - \mathbf{q})}.$$

Now, if Σ is diagonal, then

$$d_M(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^n \frac{(p_i - q_i)^2}{\sigma_i^2}},$$

where σ_i^2 is the variance along the i-th dimension. If Σ is the identity matrix, then we recover the **Euclidean distance**

$$d_2(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^n (p_i - q_i)^2}.$$

When using the Euclidean distance in an anomaly detection context, a **linear normalization** is usually applied to each dimension so that each entry lies in the hypercube $[-1, 1]^n$.

The **Minkowski distance** of order p is a generalization of the Euclidean distance:

$$d_p(\mathbf{p},\mathbf{q}) = \left(\sum_{i=1}^{n} |p_i - q_i|^p\right)^{1/p}$$

For $p = 2$ we recover the Euclidean distance d_2, for $p = 1$ the **Manhattan distance**

$$d_1(\mathbf{p},\mathbf{q}) = \sum_{i=1}^{n} |p_i - q_i|,$$

and for $p = \infty$ the **supremum distance**

$$d_\infty(\mathbf{p},\mathbf{q}) = \max_{i=1}^{n} |p_i - q_i|.$$

The Minkowski distance d_p is only actually a distance function (i.e., a **metric**) when $p \geq 1$, but an exception is made for

$$d_{-\infty}(\mathbf{p},\mathbf{q}) = \min_{i=1}^{n} |p_i - q_i|$$

to fall within the same framework.

The **Jaccard similarity** of two datasets P and Q, is defined as the size of their intersection divided by the size of their union

$$J(P,Q) = \frac{|P \cap Q|}{|P \cup Q|} = \frac{|P \cap Q|}{|P| + |Q| - |P \cap Q|}$$

Their **Jaccard distance** is then taken to be $1 - J(P,Q)$.

This definition can be extended to compare binary vectors (i.e. vectors with entries in $\{0,1\}$) of the same length. Given two binary vectors $\mathbf{p}$ and $\mathbf{q}$ of length n, consider an arbitrary set D of size n. Then $\mathbf{p}$ and $\mathbf{q}$ can be viewed as subsets of D: if $p_i = 1$ then $\mathbf{p}$ is said to contain the i-th element of D, while if $p_i = 0$ then it does not. Viewing $\mathbf{p}$ and $\mathbf{q}$ in this way allows us to compute their Jaccard similarity, and thus their Jaccard distance.

Finally, let $\mathbf{p}, \mathbf{q} \neq \mathbf{0}$. Recall that $\mathbf{p} \cdot \mathbf{q} = \|p\|\|q\| \cos\theta$, where θ is the angle between $\mathbf{p}$ and $\mathbf{q}$. The **cosine similarity** between $\mathbf{p}$ and $\mathbf{q}$ is the cosine of θ, which can be computed as

$$\cos\theta = \frac{p \cdot q}{\|p\|\|q\|} = \frac{\sum_{i=1}^{n} p_i q_i}{\sqrt{\sum_{i=1}^{n} p_i^2}\sqrt{\sum_{i=1}^{n} q_i^2}}.$$

This value ranges between 1 and -1, with 1 attained when $\mathbf{p} = \mathbf{q}$, -1 when $\mathbf{p} = -\mathbf{q}$, and 0 when $\mathbf{p}$ and $\mathbf{q}$ are perpendicular.

Armed with these concepts, we can now explore distance- (and eventually density-) based methods for anomaly detection.

Distance-Based Approaches

All these distance functions can be used to create basic anomaly detection algorithms (the ideas can also be extended to more complex algorithms).

Given some distance function d, dataset D, and integers $k, \nu \leq |D|$, the **distance to all points** anomaly detection algorithm considers each point $\mathbf{p}$ in D and adds the distance from $\mathbf{p}$ to every other point in D, i.e.

$$a(\mathbf{p}) = \sum_{\mathbf{q} \neq \mathbf{p} \in D} d(\mathbf{q},\mathbf{p}).$$

The ν points with largest values for a are then said to be **anomalous according to** a. This approach often selects the most extreme observations as anomalous, which may be of limited use in practice.

The **distance to nearest neighbour** algorithm defines

$$a(\mathbf{p}) = \min_{\mathbf{q} \neq \mathbf{p} \in D} d(\mathbf{q},\mathbf{p}),$$

with a similar definition for the ν anomalous points.

The **average distance to k nearest neighbours** and **median distance to k nearest neighbours** are defined similarly.

2.2 Density-Based Methods

Density-based approaches, on the other hand, view points as anomalous if they occur in **low density regions**.

Local Outlier Factor

The **Local Outlier Factor** (LOF) algorithm was proposed in 2000 by [39] (a summary can be found in Section 6.4.2 of [12]). LOF works by measuring the local deviation of each point in a dataset from its k nearest neighbours, with a point said to be anomalous if this **deviation is large**.

A **local k—region** around a point $\mathbf{p}$ is defined as the k nearest neighbours of $\mathbf{p}$. The density of points in each of their respective local k—neighbourhoods is estimated, and compared to the density of the local k—neighbourhoods of the point within their own k—neighbourhood.

This can then be used to identify outliers that inhabit regions of lower density than their neighbours, as $\mathbf{p}$ would be in Figure 11. The formal procedure is shown in Algorithm 1.

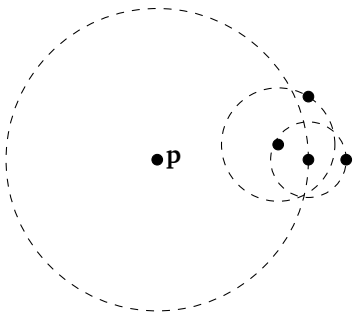


Figure 11. For $k = 2$, $\mathbf{p}$ is an outlier as it has lower density than its neighbours.

Algorithm 1: Local Outlier Factor (LOF)

Input: dataset D, point $\mathbf{p} \in D$, integer k for number of nearest neighbours to consider, distance function d
Compute the distance between all points in D
for $\mathbf{p} \in D$ **do**
  for $\mathbf{q} \in D \setminus \{\mathbf{p}\}$ **do**
    Compute $d(\mathbf{p}, \mathbf{q})$
  end
  Order D by increasing distance from $\mathbf{p}$
  Set $d_k(\mathbf{p}) = d(\mathbf{p}, \mathbf{q}_k)$
end
Find the k nearest neighbours of $\mathbf{p}$
Set $N_k(\mathbf{p}) = \{\mathbf{q} \in D \setminus \{\mathbf{p}\} : d(\mathbf{p}, \mathbf{q}) \leq d_k(\mathbf{p})\}$
Define the reachability distance $d_{\text{reach}}(\mathbf{p}, \mathbf{q}) = \max\{d_k(\mathbf{q}), d(\mathbf{p}, \mathbf{q})\}$
Define the average reachability distance $\overline{d_{\text{reach}}}(\mathbf{p}) = \frac{\sum_{\mathbf{q} \in N_k(\mathbf{p})} d_{\text{reach}}(\mathbf{p}, \mathbf{q})}{|N_k(\mathbf{p})|}$
Define the local reachability density $\ell_k(\mathbf{p}) = \left(\overline{d_{\text{reach}}}(\mathbf{p})\right)^{-1}$
Compute the local outlier factor $a_k(\mathbf{p}) = \frac{\sum_{\mathbf{q} \in N_k(\mathbf{p})} \frac{\ell_k(\mathbf{q})}{\ell_k(\mathbf{p})}}{|N_k(\mathbf{p})|}$
Output: LOF $a_k(\mathbf{p})$

LOF is able to identify **local outliers**, but selecting a threshold beyond which a point is considered an outlier is difficult.

LOF introduces the idea of a **reachability distance**, which improves the stability of results within clusters/regions: within a local k—region around $\mathbf{p}$, it is simply the maximal distance to its k—neighbours; outside of that region, it is the actual distance from $\mathbf{p}$.

In Figure 12 (with $k = 3$), for instance, the points $\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3$ all have the same reachability distance from $\mathbf{p}$ as they are all 3-neighbours of $\mathbf{p}$, that is,

$$d_{\text{reach}}(\mathbf{p}, \mathbf{q}_1) = d_{\text{reach}}(\mathbf{p}, \mathbf{q}_2) = d_{\text{reach}}(\mathbf{p}, \mathbf{q}_3) = d(\mathbf{p}, \mathbf{q}_3).$$

The point $\mathbf{q}_4$, on the other hand, has $d_{\text{reach}}(\mathbf{p}, \mathbf{q}_4) = d(\mathbf{p}, \mathbf{q}_4)$ as it is not a k-neighbour of $\mathbf{p}$.

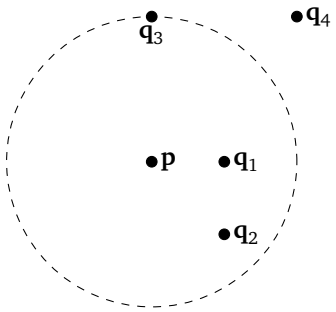


Figure 12. The region of uniform reachability distance around $\mathbf{p}$ for $k = 3$.

DBSCAN

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) was proposed in 1996 by [18] (a summary can be found in Section 4.1.5 of [12]). As its name suggests, it is a density-based clustering algorithm that groups nearby points together and labels points that do not fall in the clusters as **anomalies**.

Hierarchical DBSCAN (HDBSCAN) [25] was introduced in 2013. It notably removes the problem of choosing the parameter for the radius of a neighbourhood by considering all possible radii. Further documentation can be found at [26].

In DBSCAN,

- a point $\mathbf{p}$ is a **core point** if there are a minimum number m of points within distance r of $\mathbf{p}$;
- a point $\mathbf{q}$ is a **border point** if it is not itself a core point but is within distance r of one, and
- a point $\mathbf{o}$ is an **outlier** if it is neither a core nor a border point.

DBSCAN considers each point in the dataset individually. If that point is an outlier, then it is added to a list of outliers. Otherwise if it is a core point, then its r-neighbourhood forms the beginning of a new cluster. Each point in this r-neighbourhood is then considered in turn, with the r-neighbourhoods of other core points contained in the neighbourhood being added to the cluster.

This expansion repeats until all points have been examined. During this step points that were previously labelled as outliers may be updated as they become border points in this new cluster. This process continues until every point has either been assigned to a cluster or labelled as an outlier (see Algorithm 2 and Figure 13).

While DBSCAN's dual use as a clustering algorithm may seem irrelevant in the outlier detection setting, its ability to succesfully identify clusters is crucial to being able to label the remaining points as outliers.

On the one hand, in DBSCAN the number of clusters does not need to be known beforehand (unlike in k—means and other clustering algorithms) and clusters of arbitrary shape can be detected.

Furthermore, when using HDBSCAN, only the parameter for the minimum cluster size m is required, which can

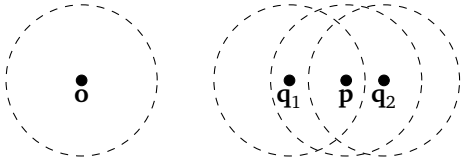


Figure 13. For minimum neighbourhood size $m = 2$ and this fixed radius r, $\mathbf{o}$ is an outlier, $\mathbf{p}$ a core point, and $\mathbf{q}_1$ and $\mathbf{q}_2$ are border points.

Algorithm 2: DBSCAN

Input: dataset D, distance function d, neighbourhood radius $r > 0$, minimum number of points to be considered a cluster $m \in \mathbb{N}$
$Clusters = \{\}$
$Outliers = \{\}$
for $\mathbf{p} \in D$ **do**
  if $\mathbf{p} \in Outliers \cup (\cup_{C \in Clusters} C)$ **then**
    continue
  end
  Set $N(\mathbf{p}) = \{\mathbf{q} \in D : d(\mathbf{p}, \mathbf{q}) \leq r\}$
  if $|N(\mathbf{p})| < m$ **then**
    Add $\mathbf{p}$ to *Outliers*
    continue
  end
  else
    $Cluster = N(\mathbf{p})$
    for $\mathbf{q} \in Cluster \setminus \{\mathbf{p}\}$ **do**
      if $\mathbf{q} \in Outliers$ **then**
        Remove q from *Outliers*
      end
      else if $\mathbf{q} \in \cup_{C \in Clusters} C$ **then**
        continue
      end
      Set $N(\mathbf{q}) = \{q' \in D : d(q, q') \leq r\}$
      if $|N(\mathbf{q})| \geq m$ **then**
        Cluster = Cluster $\cup N(\mathbf{q})$
      end
    end
  end
  Add *Cluster* to *Clusters*
end
return *Outliers*
Output: a list of *outliers*

be set fairly intuitively, which is not the case for the parameters in general clustering algorithms: if the elements of D are n—dimensional, take $m \geq n + 1$ (larger values of m allow for better noise identification).

On the other hand, DBSCAN is not deterministic, as border points can be assigned to different clusters depending on the order in which core points are considered (this does not affect its use as an anomaly detection algorithm, however).

In high dimensions, the ability of any distance function based on Euclidean distance to distinguish near and distant points diminishes due to the **Curse of Dimensionality**; thus in high dimension spaces, it become ineffective (as do other clustering algorithms).

Finally, DBSCAN cannot handle differences in local densities as the radius of a neighbourhood r is fixed; this could lead to sparser clusters being labelled as outliers, or to outliers surrounding a denser cluster being included in the cluster. This issue is overcome in HDBSCAN.

Isolation Forest

The previously discussed approaches first construct models of what normal points look like, and then identify points that do not fit this model. The Isolation Forest algorithm [17] introduced in 2008 instead tries to explicitly identify outliers under the assumptions that there are few outliers and that these outliers have very different attributes compared to normal points. Doing so allows the use of sampling techniques that increase algorithmic speed while decreasing memory requirements.

The Isolation Forest algorithm tries to isolate anomalous points. It does this by randomly selecting an attribute and then randomly selecting a split value between that attribute's min and max values. This recursively partitions the points until every point is isolated in its own partition.

Recursive partitioning yields a binary tree called an **Isolation Tree**. The root of this tree is the entire dataset; each node is a subset of the observations, and each branch corresponds to one of the generated partitions. The leaf nodes are singleton sets containing a single isolated point. Each point is then assigned a score derived from how deep in the tree its singleton partition appears (see Figure 14 and Algorithm 3).

As points that are shallower in the tree were easier to separate from the rest, these are the likely **outliers**. Since only shallow points are of interest, once the height of the tree has reached a given threshold (the expected height of a random binary tree, say), further construction of the tree can be stopped to decrease computational cost.

Additionally, instead of building a tree from the entire dataset, a tree can be constructed from a subset. The location of any point within this smaller tree can then be estimated, again saving computational and memory resources. These two improvements are detailed in the original paper [17].

Once a number of Isolation Trees have been randomly generated (**Isolation Forest**), a score can be computed for each point. This is done by searching each tree for the location of a given point and noting the path length required to reach it. Once a point's path length in each tree has been computed, the average path length is taken to be its score.

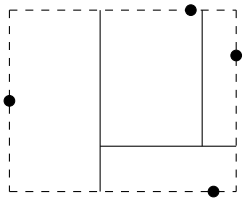


Figure 14. A partitioning constructed during Isolation Tree generation.

Algorithm 3: Recursive Isolation Tree Construction: iTree(D)

Input: dataset D
if $|D| \leq 1$ **then**
  return {}
end
else
  Let $\overline{A}$ be a list of attributes in D
  Randomly select an attribute $A \in \overline{A}$
  Randomly sample a point s from $[\min_{\mathbf{q}\in D} A(\mathbf{q}), \max_{\mathbf{q}\in D} A(\mathbf{q})]$
  Return

$$\text{Node}\begin{cases}\text{LeftChild} & = \text{iTree}(\{\mathbf{q} \in D : A(\mathbf{q}) \leq s\}) \\ \text{RightChild} & = \text{iTree}(\{\mathbf{q} \in D : A(\mathbf{q}) > s\}) \\ \text{NodeValue} & = D\end{cases}$$

end
Output: Binary tree with node values that are subsets of D

It can be desirable to construct a normalized anomaly score that is independent of the size of the dataset. In order to do this, the expected path length of a random point in an Isolation Tree (i.e. binary tree) must be estimated. With $n = |D|$, it can be shown that the expected length is

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n},$$

where $H(n-1)$ is the $(n-1)^{\text{th}}$ harmonic number, which can be approximated by $\ln(n-1)+0.577$; $c(n)$ is then used to normalize the final anomaly score $a(\mathbf{p})$ for $\mathbf{p} \in D$, which is given by

$$\log_2 a(\mathbf{p}) = -\frac{\text{average path length to } \mathbf{p} \text{ in the Isolation Trees}}{c(n)}.$$

Thus defined, $a(\mathbf{p}) \in [0,1]$, with $a(\mathbf{p}) \approx 1$ suggesting $\mathbf{p}$ is an **anomaly**, $a(\mathbf{p}) \leq 0.5$ suggesting $\mathbf{p}$ is a normal point; if all points receive a score around 0.5, this suggests that there are no anomalies present.

Isolation Forests have small time and memory requirements; can handle high dimensional data, and do not need observations to have been labeled anomalies in the training set, but the anomaly score assigned to a given point can have high variance over multiple runs of the algorithm. The authors of [24] propose some solutions.

In general, density-based schemes are more powerful than distance-based schemes when a dataset contains patterns with diverse characteristics, but less effective when the patterns are of comparable densities with the outliers [44].

Algorithm 4: Isolation Forest

Input: dataset D, integer t number of Isolation Trees
Forest $= \{\}$
for $i = 1$ *to* t **do**
  Tree $=$ iTree(D)
  Add *Tree* to *Forest*
end
for $\mathbf{p} \in D$ **do**
  PathLengths $= \{\}$
  for *Tree in Forest* **do**
    Find the path length ℓ from the root of Tree to node $\{\mathbf{p}\}$
    Add ℓ to *PathLengths*
  end
  $AveragePathLength = \frac{\sum_{\ell \in PathLengths} \ell}{t}$
  Set $a(\mathbf{p}) = 2^{-\frac{AveragePathLength}{c(|D|)}}$
end
Output: Anomaly score $a(\mathbf{p}) \in [0,1]$ for each $\mathbf{p} \in D$

3. Qualitative Methods of Anomaly Detection

New challenges are presented by non-numerical variables.

3.1 Definitions and Challenges

Categorical Variables

A **categorical variable** (or qualitative variable) is one whose levels are measured on a nominal scale; examples include an object's colour, the mother tongue of an individual, her favourite meal, and so forth.

The **central tendency** of the values of a categorical variable is usually given by its **mode**; measures of spread are harder to define consistently (the proportion of levels with more than a certain percentage of the observations above a given threshold could be used as rough gauge, but difficulties with this approach are readily apparent).

We often associate qualitative feature to numerical values, but with the caveat that these should not be interpreted as numerals; if we use the code "red" $= 1$ and "blond" $= 2$ to represent hair colour, for instance, we obviously cannot conclude that "blond" $>$ "red", even though $2 > 1$.

A categorical variable that has exactly two levels is called a **dichotomous feature** (or a binairy variable); those with more than two levels are called **polytomous variables**.

Challenges of Anomaly Detection with Categorical Data

Representing categorical variables with numerical features can lead to traps; consequently, using anomaly detection methods based on distance metrics or on density is not recommended in the qualitiative context, unless they have first been modified appropriately.

3.2 Review of Two Methods

We present two of the categorical methods below.

AVF Algorithm

The **Attribute Value Frequency** (AVF) algorithm offers a fast and simple way to detect outlying observations in categorical data, which minimizes the amount of data analyses, without having to create or search through various combinations feature levels (which increase the search time).

Intuitively, outlying observations are points which occur relatively infrequently in the (categorical) dataset; an "ideal" anomalous point is one for which **each feature value is extremely anomalous** (or relatively infrequent).

The **rarity** of an attribute level can be measured by summing the number of times the corresponding feature takes that value in the dataset.

Let's say that there are n observations in the dataset: $\{\mathbf{x}_i\}$, $i = 1, \ldots, n$, and that each observation is a collection of m features. We write

$$\mathbf{x}_i = (x_{i,1}, \cdots, x_{i,\ell}, \cdots, x_{i,m}),$$

where $x_{i,\ell}$ ℓth feature's level. Using the reasoning presented above, the **AVF score** (shown below) is a good tool to determine whether $\mathbf{x}_i$ should be considered an outlier or not:

$$\text{AVFscore}(\mathbf{x}_i) = \frac{1}{m}\sum_{\ell=1}^{m} f(x_{i,\ell}),$$

where $f(x_{i,\ell})$ is the number of observations $\mathbf{x}_i$ for which the ℓth feature takes on the level $x_{i,\ell}$. A low AVF score indicates that the observation is more likely to be an outlier.

Since AVFscore($\mathbf{x}_i$) is essentially a sum of m positive numbers, it is minimized when each of the sum's term is minimized, individually. Thus, the "ideal" anomalous observation described above minimizes the AVF score; the minimal score is reached when each of the observation's features' levels occurs only once in the dataset.

As shown by the AVF pseudocode (see Algorithm 5), once the AVF score is calculated for all points, the k outliers returned by the algorithms are the k observations with the smallest AVF scores (the algorithm's complexity is $\mathcal{O}(nm)$).

Algorithm 5: AVF

Inputs: dataset D (n observations, m features), number of anomalous observations k
while $i \leq n$ **do**
  j = 1
  AVFscore($\mathbf{x}_i$) = $f(x_{i,j})$
  while $j \leq m$ **do**
    AVFscore($\mathbf{x}_i$) = AVFscore($\mathbf{x}_i$) + $f(x_{i,j})$; $j = j + 1$
  end
  AVFscore($\mathbf{x}_i$) =Mean(AVFscore($\mathbf{x}_i$))
  $i = i + 1$
end
Outputs: k observations with smallest AVF scores

Greedy Algorithm

The **greedy** algorithm "greedyAlg1" is an algorithm which identifies the set OS of candidate anomalous observations in an efficient manner.

The mathematical formulation of the problem is simple – given a dataset D and a number k of anomalous observations to identify, we solve the optimization problem

$$\text{OS} = \arg\min_{O \subseteq D}\{H(D \setminus O)\}, \quad \text{subject to } |O| = k,$$

where the **entropy** of the subset $D \setminus O$ is the sum of the entropy of each of feature on $D \setminus O$:

$$H(D \setminus O) = H(X_1; D \setminus O) + \cdots + H(X_m; D \setminus O)$$

et

$$H(X_\ell; D \setminus O) = -\sum_{z_\ell \in S(X_\ell; D \setminus O)} p(z_\ell) \log p(z_\ell),$$

where $S(X_\ell; D \setminus O)$ is the set of levels that the ℓth feature takes in $D \setminus O$.

The "greedyAlg1" algorithm solves the optimization problem as follows:

1. The set of outlying and/or anomalous observations OS is initially set to be empty, and all observations of $D \setminus$ OS are identified as normal (or regular).

2. Compute $H(D \setminus \text{OS})$.

3. Scan the dataset in order to select a candidate anomalous observation: every normal observation $\mathbf{x}$ is temporarily taken out of $D \setminus$ OS to create a subset $D'_{\mathbf{x}}$, whose entropy $H(D'_{\mathbf{x}})$ is also computed.

4. The observation $\mathbf{z}$ which provides the **maximal entropy impact**, i.e. the one that minimizes

$$H(D \setminus \text{OS}) - H(D'_{\mathbf{x}}), \quad \mathbf{x} \in D \setminus \text{OS},$$

   is added to OS.

5. Repeat steps 2-4 another $k-1$ times to obtain a set OS of k candidate anomalous observations.

You can find more details in the in the source article [46]; an interesting detail is that the complexity of the algorithm is expected to be $\mathcal{O}(nmp)$, which implies that it is **scalable**.

4. Anomalies in High-Dimensional Datasets

Anomaly detection is a broad field of study that has been applied to a large number of areas. Nowadays, many real datasets are very large; in some scenarios, the observations may contain **hundreds or thousands of features** (or dimensions).

Many classical methods use proximity (distance) concepts for anomaly detection (see Section 2 for a sample of such methods) and can only be applied in cases where the sample size n is larger than the dimension p ($n > p$).

The management of **high-dimensional data** ($n < p$) offers specific difficulties: indeed, in such spaces observations are often **isolated** and **scattered** (or sparse) and the notion of proximity fails to maintain its relevance.

In that case, the notion of defining significant outliers is much more complex and not obvious: many conventional methods of detecting outliers are simply not efficient in the high-dimensional context, due to this **curse of dimensionality**.

The remainder of this section is organized as follows: first, an attempt is made to define the concept and the challenges; then, anomaly detection techniques are discussed; finally, we end with a detailed description of ensembles and subspace methods. Our approach mainly follows those found in [1, 9, 10, 12, 16].

4.1 Definitions and Challenges

As we have seen previously, an anomalous observation is one that deviates or behaves differently from other the observations in the dataset, which makes us suspect that it was generated by some other mechanism [1]; such an observation would, of course, be considered to be irregular.

The challenges of anomaly and outlier detection in high-dimensional data lie in the facts that:

- the notion of distance fails to retain its relevance due to the curse of dimensionality (whence "the problem of detecting outliers is like finding a needle in a haystack" [16]);
- every point in such datasets has a tendency to be an outlier, and
- datasets become more sparse as the dimension of the feature space increases.

The authors of [2] consider that in order to deal properly with large datasets, detection methods should:

1. allow for effective management of sparse data issues;
2. provide interpretability of the discrepancies (i.e. how the behaviour of such observations is different);
3. allow anomalie measurements to be compared, and
4. consider the local data behaviour to determine whether an observation is abnormal or not.

4.2 Projection-Based Methods

Nowadays, it is common to deal with very large data sets known as HDLSS (**high dimension, low sample size**), which can contain hundreds of variables (or even more).

As a result, the curse of dimensionality affects the efficiency of conventional anomaly/outlier detection methods.

One solution to this problem is to **reduce the dimensionality** of the dataset while preserving its essential characteristics. Such projecion-based methods

- principal component analysis,
- linear discriminant analysis,
- feature selection, etc.

In this section, We provide details on one such method: PCA.

Principal Components Analysis

Principal components analysis (PCA) aims to find a representation of the original dataset in a lower-dimensional subspace (such as a line or a plane) containing the greatest possible variation.

PCA corresponds to an orthogonal linear transformation of the data into a new coordinate system, such that the largest variance resulting from a scalar projection of the data is on the first coordinate (the **first principal component**), the second largest variance on the second coordinate, and so forth.

PCA is used in various contexts:

- as a dimension reduction method used during the data pre-processing step;
- as a data visualization aid, and, in the scenario of interest for this report,
- as an anomaly and outlier detection method.

Let the dataset be represented by a numerical, centered, and scaled $n \times p$ matrix $\mathbf{X} = [\mathbf{X}_1, \cdots, \mathbf{X}_p]$ with n observations (number of rows) and p features (number of columns). The **principal components** can be written as linear combinations of the variables

$$\mathbf{Y}_i = \ell_i^\top \mathbf{X} = \ell_{1,i}\mathbf{X}_1 + \cdots + \ell_{p,i}\mathbf{X}_p; \quad i = 1, \cdots, k,$$

with $k \leq p$, yielding the largest variance subjet to the constraint $\|\ell_i\| = 1$ (where $\|\cdot\|$ represents the Euclidean norm). We can thus deduce that

$$\begin{aligned} \mathrm{Var}(Y_i) &= \mathrm{Var}\left(\ell_i^\top \boldsymbol{X}\right) = \ell_i^\top \boldsymbol{\Sigma} \ell_i\,, \\ \mathrm{Cov}(Y_i, Y_k) &= \mathrm{Cov}\left(\ell_i^\top \boldsymbol{X}, \ell_k^\top \boldsymbol{X}\right) = \ell_i^\top \boldsymbol{\Sigma} \ell_k\,. \end{aligned}$$

In other words, PCA finds the **loadings vector** ℓ_1 which maximizes the variance of Y_1, i.e.

$$\ell_1 = \underset{\|\ell_1\|=1}{\arg\max} \left\{ \ell_1^\top \mathbf{X}^\top \mathbf{X} \ell_1 \right\},$$

then the loadings vector ℓ_2 (not correlated with ℓ_1) which maximizes the variance of Y_2, i.e.

$$\ell_2 = \underset{\|\ell_2\|=1,\, \ell_1^\top \ell_2 = 0}{\arg\max} \left\{ \ell_2^\top \mathbf{X}^\top \mathbf{X} \ell_2 \right\}.$$

Similarly, the loadings vector ℓ_k is not correlated with any of the ℓ_i, $i < k$, and maximizes the variance of Y_k, i.e.

$$\ell_k = \underset{\substack{\|\ell_k\|=1, \\ \ell_i^\top \ell_k = 0,\ \forall\, i<k}}{\arg\max} \left\{ \ell_k^\top \mathbf{X}^\top \mathbf{X} \ell_k \right\}. \tag{1}$$

We solve (1) for all $i < k$ through the Lagrangian

$$L = \ell_k^\top \mathbf{X}^\top \mathbf{X} \ell_k - \lambda_k (\ell_k^\top \ell_k - 1) - w \ell_i^\top \ell_k.$$

The critical points are found by differentiating with respect to each of the entries of ℓ_k, λ_k and w, and setting the result to 0. Simplifying, we obtain

$$\mathbf{X}^\top \mathbf{X} \ell_k = \lambda_k \ell_k$$
$$\ell_k^\top \ell_k = 1 \quad \text{et} \quad \ell_k^\top \ell_i = 0, \quad \text{for all } i < k.$$

The loadings vector ℓ_k is thus the **eigenvector** of the design matrix $\mathbf{X}^\top \mathbf{X}$ associated to the kth largest eigenvalue.

The **proportion of the variance which can be explained** by the PCA can be calculated by first noting that

$$\sum_{i=1}^{p} \mathrm{Var}(Y_i) = \sum_{i=1}^{p} \ell_i^\top \mathbf{\Sigma} \ell_i = \sum_{i=1}^{p} \lambda_i.$$

Consequently, the proportion of the total variance explained by the ith principal component is

$$0 \leq \frac{\lambda_i}{\sum_{i=1}^{p} \lambda_i} \leq 1$$

The quality of the PCA results is strongly dependent on the number of retained principal components, that is, on the dimension k of the subspace on which the observations are projected. There are multiple ways to select the "right" k – we will briefly present two of them.

The proportion of the total variance explained by the first k principal components is given by

$$p_k = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{p} \lambda_i}.$$

One approach is to retain k principal components, where k is the smallest value for which p_k surpasses some pre-established threshold (often taken between 80% and 90%).

The **scree plot method**, on the other hand, consists in drawing the curve given by the decreasing eigenvalues (the **scree plot**), and to identify the curve's "elbows". These points correspond to principal components for which the variance decreases at a slower rate with added components. If such an elbow exists, we would retain the eigenvalues up to it (and thus, the corresponding principal components).

Example 6. PCA is applied on a dataset of genetic expression measurements, for $n = 72$ leukemia patients and $p = 7128$ genes [47]. The scree plot suggests that only one principal component should be retained; the projection on the first 3 principal components is also shown in Figure 15 (on the right). Some R code is given below.

```
leukemia.big <-
    read.csv("http://web.stanford.edu/~hastie/
    CASI_files/DATA/leukemia_big.csv")
leukemia.big <- t(leukemia.big)
leukemia.big.scaled <-
    scale(leukemia.big)
pca.leukemia <-
    prcomp(leukemia.big.scaled)
plot(pca.leukemia)
pca.leukemia.s <- summary(pca.leukemia)
plot(pca.leukemia.s$importance[3,])
```

There are other PCA-associated dimension reduction methods, such as the singular value decomposition, kernel PCA, and so forth; more details are available in [48].

What is the link with anomaly and/or outlier detection? Once the dataset has been projected on a lower-dimensional subspace, the curse of dimensionality is mitigated – it is on the projected data that the traditional detection methods are applied.

Note, however, that any such reduction necessarily leads to a loss of information, which can affect the accuracy of the detection procedure, especially if the presence/absence of anomalies is not aligned with the dataset's principal components.

Distance-Based Outlier Basis Using Neighbours

Using PCA for anomaly detection is potentially problematic, however: whether an observation is anomalous or not does not figure in the construction of the principal component basis $\{\mathrm{PC}_1, \ldots, \mathrm{PC}_k\}$ – there isn't necessarily a correlation between the axes of heightened variance and the presence or absence of anomalies.

The **distance-based outlier basis using neighbours** algorithm (DOBIN) builds a basis which is better suited for the eventual detection of outlying observations. DOBIN's main idea is to search for nearest neighbours that are in fact relatively distant from one another:

1. We start by building a space $\mathbf{Y} = \{\mathbf{y}_\ell\}$ which contains $M \ll n(n+1)/2$ vectors of the form

   $$\mathbf{y}_\ell = (\mathbf{x}_i - \mathbf{x}_j) \odot (\mathbf{x}_i - \mathbf{x}_j),$$

   where $\odot$ is the element-by-element Hadamard multiplication, and for which the 1—norm

   $$\|\mathbf{y}_\ell\|_1 = (x_{1,1} - x_{2,1})^2 + \cdots + (x_{1,p} - x_{2,p})^2$$

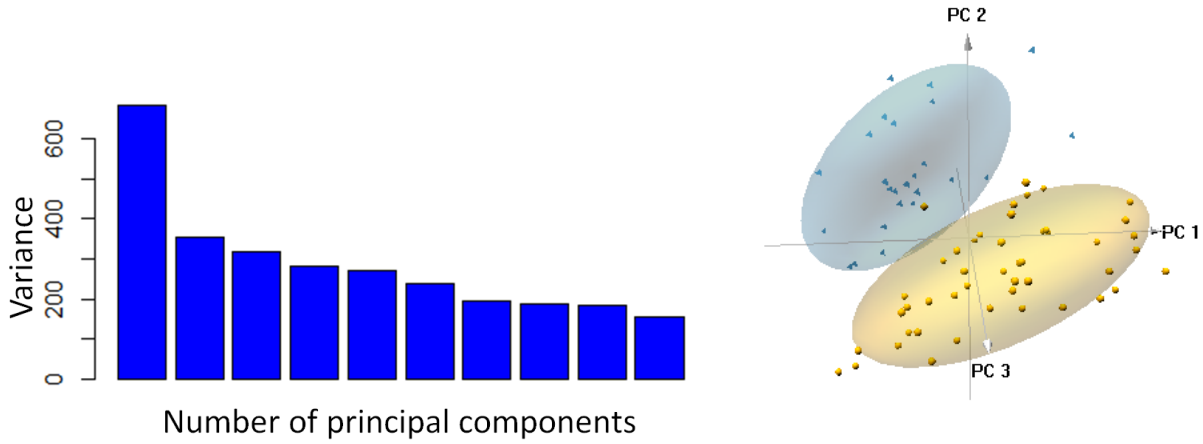


Figure 15. Scree plot (left); projection on the first 3 principal components (right).

is the square of the distance between $\mathbf{x}_i, \mathbf{x}_j \in \mathbf{X}$ (the selection of each of the M observation pairs is made according to a rather complex procedure which only considers $\mathbf{x}_i$ and $\mathbf{x}_j$ if they are part of one another's k—neighbourhood, for $k \in \{k_1, \ldots, k_2\}$); the set $\mathbf{Y}$ thus contains points for which $\|\mathbf{y}_\ell\|_1$ is relatively large, which is to say that the observations $\mathbf{x}_i$ are $\mathbf{x}_j$ fairly distant from one another even if they are k—neighbours of each other;

2. we next build a basis $\{\eta_1, \ldots, \eta_p\} \subset \mathbb{R}^p$ where each η_i is a unit vector given by a particular linear combination of points in $\mathbf{Y}$; they can be found using a Gram-Schmidt-like procedure:

$$\begin{aligned} \mathbf{y}_{\ell_0} &= \mathbf{y}_\ell, \quad \ell = 1, \ldots, M \\ \mathbf{y}_{\ell_{b-1}} &= \mathbf{y}_{\ell_{b-2}} - \langle \eta_{b-1} \mid \mathbf{y}_{\ell_{b-2}} \rangle, \quad \ell = 1, \ldots, M \\ \eta_b &= \frac{\sum_{\ell=1}^{M} \mathbf{y}_{\ell_{b-1}}}{\left\| \sum_{\ell=1}^{M} \mathbf{y}_{\ell_{p-1}} \right\|_2}, \end{aligned}$$

for $b = 1, \ldots, p$,

3. and we tranform the original dataset $\mathbf{X}$ according to $\hat{\mathbf{X}} = \mathscr{T}(\mathbf{X})\Theta$, where $\mathscr{T}(\mathbf{X})$ normalizes each feature of $\mathbf{X}$ according to a problem-specific scheme (Min-Max or Median-IQR, say) and

$$\Theta = [\eta_1 \mid \cdots \mid \eta_p]$$

is a orthogonal $p \times p$ matrix.

It is on the transformed space (which plays an analogous role to the subspace projection of $\mathbf{X}$ in PCA) that we apply the various outlier and anomaly detection algorithms.

The full details contain a fair number of technical complications; the interested reader is invited to consult the original documentation [7] (note that the algorithm is implemented in R *via* the module *dobin*).

4.3 Ensembles Methods

In the preceding sections, we have described various anomaly detection algorithms whose relative performance varies with the type of data being considered. It's usually impossible to come up with an algorithm that outperforms all the others.

This is because a particular anomaly detection algorithm may be well adapted to a data set and may be successful in detecting abnormal or outlier observations, but it may not work with other data sets whose characteristics do not match the first data set.

The impact of such a mismatch between algorithms can be mitigated by using **ensemble methods**, where the results of several algorithms are considered before making a final decision. Such an approach often provides the best results and thus improves the performance of the base anomaly detection algorithms [12].

We will consider two tyes of ensemble methods: **sequential ensembles** (boosting) and **independent ensembles**,

Sequential Ensembles

Sequential ensembles requires a given algorithm (or a set of algorithms) to be applied to a dataset in a sequential manntter, each time on a slightly different dataset derived from the previous step's dataset based on the previous steps' results, and so forth. At each step, the weight associated with each observation is modified according to the preceding results using some "boosting" method (such as AdaBoost or XGBoost, for instance).

The final result is either some weighted combination of all preceding results, or simply the results output by the last step in the sequence (see Algorithme 6).

The details are out-of-scope for this report, but can be studied in [49].

Algorithm 6: SequentialEnsemble

Inputs: dataset D, base algorithms $A_1, \ldots, A_r$
$j = 1$;
while *stopping criteria are not met* **do**
  Select an algorithm A_j based on the results from the preceding steps;
  Create a new dataset D_j from D by modifying the weight of each observation based on the results from the preceding steps;
  Apply A_j to D_j;
  $j = j + 1$;
end
Output: anomalous observations obtained by weighing the results of all previous steps

Algorithm 7: IndependantEnsemble

Inputs: dataset D, base algorithms $A_1, \ldots, A_r$
$j = 1$;
while *stopping criteria are not met* **do**
  Select an algorithm A_j;
  Create a new dataset D_j from D by (potential) re-sampling, but independently of the preceding steps' results;
  Apply A_j to D_j;
  $j = j + 1$;
end
Output: anomalous observations obtained by combining the results of all previous steps

Independent Ensembles

In an independent ensemble, we instead apply different algorithms (or different instanciations of the same algorithm) to the dataset (or some resampled dataset).

Choices made at the data and algorithm level are independent of the results obtained in previous runs (unlike in a sequential ensemble). The results are then combined to obtain more robust outliers (see Algorithm 7).

Every base anomaly detection algorithm provides an anomaly score (or an abnormal/regular classification) for each observation in D; observations with higher scores are considered to be more anomalous, observations with lower scores more normal.

The results are then combined using a task-specific method in order to provide a more robust classification of anomalous or outlying observations.

Many such combination techniques used in practice:

- **majority vote**,
- **average**,
- **minimal rank**, etc.

Let $\alpha_i(\mathbf{p})$ represent the (normalized) **anomaly score** of $\mathbf{p} \in D$, according to algorithm A_i. If $\alpha_i(\mathbf{p}) \approx 0$, it is unlikely that $\mathbf{p}$ is an anomaly according to A_i, whereas if $\alpha_i(\mathbf{p}) \approx 1$, it is quite likely that $\mathbf{p}$ according to A_i.

The **rank** of $\mathbf{p} \in D$ according to A_i, on the other hand, is denoted by $r_i(\mathbf{p})$: the higher the rank (smaller number), the higher the anomaly score *vice versa*. In a dataset with n observations, the rank varies from 1 to n (ties are allowed).

If the base detection algorithms are $A_1, \ldots, A_m$, the anomaly score and the rank of an observation $\mathbf{p} \in D$ according to the independent ensemble method are, respectively,

$$\alpha(\mathbf{p}) = \frac{1}{m}\sum_{i=1}^{m} \alpha_i(\mathbf{p}) \qquad \text{and} \qquad r(\mathbf{p}) = \min_{1 \le i \le m} \{r_i(\mathbf{p})\}.$$

If $n = m = 3$, for instance, we could end up with

$$\alpha_1(\mathbf{p}_1) = 1.0,\ \alpha_1(\mathbf{p}_2) = 0.9,\ \alpha_1(\mathbf{p}_3) = 0.0;$$
$$\alpha_2(\mathbf{p}_1) = 1.0,\ \alpha_2(\mathbf{p}_2) = 0.8,\ \alpha_2(\mathbf{p}_3) = 0.0;$$
$$\alpha_3(\mathbf{p}_1) = 0.1,\ \alpha_3(\mathbf{p}_2) = 1.0,\ \alpha_3(\mathbf{p}_3) = 0.0.$$

Using the mean as the combination techniques, we obtain

$$\alpha(\mathbf{p}_1) = 0.7,\ \alpha(\mathbf{p}_2) = 0.9,\ \alpha(\mathbf{p}_3) = 0.0,$$

whence

$$\mathbf{p}_2 \succeq \mathbf{p}_1 \succeq \mathbf{p}_3,$$

that is, $\mathbf{p}_2$ is more anomalous than $\mathbf{p}_1$, which is itself more anomalous than $\mathbf{p}_3$ (see the notation introduced on page 8).

Using the minimal rank method, we obtain

$$r_1(\mathbf{p}_1) = 1,\ r_1(\mathbf{p}_2) = 2,\ r_1(\mathbf{p}_3) = 3;$$
$$r_2(\mathbf{p}_1) = 1,\ r_2(\mathbf{p}_2) = 2,\ r_2(\mathbf{p}_3) = 3;$$
$$r_3(\mathbf{p}_1) = 2,\ r_3(\mathbf{p}_2) = 1,\ r_3(\mathbf{p}_3) = 3,$$

from which

$$r(\mathbf{p}_1) = r(\mathbf{p}_2) = 1,\ r(\mathbf{p}_3) = 3,$$

whence $\mathbf{p}_1 \succeq \mathbf{p}_3$ and $\mathbf{p}_2 \succeq \mathbf{p}_3$, but $\mathbf{p}_1$ and $\mathbf{p}_2$ have the same anomalous levels.

Evidently, the results depend not only on the data set under consideration and on the base algorithms that are used in the ensemble, but also on how the results are combined.

In the context of HDLSS data, ensemble methods can sometimes allow the analyst to mitigate some of the effects of the curse of dimensionality by selecting fast base algorithms (which can be run multiple times) and focusing on building robust relative anomaly scores.

Another suggested approach is to use a different subset of the original dataset's features at each step, in order to de-correlate the base detection models.

4.4 Subspace Methods

Subspace methods have been used particularly effectively by analysts for anomaly and outlier detection in high-dimensional data sets [9, 15, 16]; it is often easier to find the sought-after observations by exploring lower-dimensional subspaces (rather than the original set).

There is thus an intrinsic interest in exploring subspaces in their own right [1, 10]. This approach eliminates **additive noise effects** often found in high dimensional spaces and leads to more robust outliers (that is, outliers which are identified as such even when using different methods).

The problem is rather difficult to solve effectively and efficiently, since the potential number of subspace projections of high-dimensional data is related exponentially to the number of features in the dataset.

The **Feature Bagging** algorithm formalizes the idea presented at the end of the preceding sub-section; it officially uses the LOF algorithm of Section 2, but any fast anomaly detection algorithm can be used instead. The anomaly scores and rankings from each run are aggregated as they are in the Independent Ensemble approach.

Algorithm 8: FeatureBagging

Input: dataset D
$j = 1$;
while *stopping criteria are not met* **do**
  Sample an integer r between $p/2$ et $p-1$;
  Randomly select r features (variables) of D in order to create a projected dataset $\tilde{D}_r$ in the corresponding r—dimensional sub-space;
  Compute the LOF result for each observation in the projected $\tilde{D}_r$;
  $j = j + 1$;
end
Output: anomaly scores given by the independent ensemble method (average, minimal rank, etc.).

There are other, more sophisticated, subspace anomaly detection methods, including:

- **High-dimensional Outlying Subspaces** (HOS) [37];
- **Subspace Outlier Degree** (SOD) [38];
- **Projected Clustering Ensembles** (OutRank) [40];
- **Local Selection of Subspace Projections** (OUTRES) [42].

It should be noted that anomaly detection and outlier analysis is still very active as an area of research, with numerous challenges. The "No Free Lunch" Theorem suggests that, importantly, there is no magic method: all methods have strengths and limitations, and the results depend heavily on the data.

5. Applications to Time Series

In this section, we discuss outliers and anomalies in time series. A **time series** is a sequential set of values tracked over a time period. The additional structure of time series makes detection of anomalies challenging; yet many algorithms attempt the task (R provides implementations of a number of these approaches). In this section, we discuss two of the most commonly-used methods.

5.1 Outliers and Anomalies in Time Series

Outliers in time series are sudden changes in the dynamics of the data that can be temporary or permanent. These anomalous recordings are usually inconsistent with the rest of the series and cannot be explained by standard time series models. If left as is, they can have a tremendous impact on the analysis (on model selection and parameter estimation, for instance).

They may, therefore, affect the forecasting power of the fitted model. It is thus important to detect and treat outliers in time series before fitting a model.

Outliers that only change the mean level of the series are said to be **deterministic**. A simple procedure can be used to detect deterministic outliers in time series: compare a time series model with no outliers to a model that includes the outliers [34]. We can then estimate the effect of processing the anomalies by looking at the differences between the models.

In general, detecting outliers in applied time series consists of determining the location, type, and magnitude of any existing outliers. There are several types of outliers:

- an **additive outlier** (AO) is an abrupt change for only one observed value – such an outlier has no effect on the subsequent observations;[10]
- an **innovational outlier** (IO) is an unsual innovation[11] in the generating process that affects all later observation – the influence of such outliers may increase with the passage of time;
- a **level shift outlier** (LS) affects the mean level of observations so that all the observations after the outlier shift to a new level – clearly such outliers have a permanent effect on the time series, and it is important to detect and process them prior to building any forecasting model;[12]
- a **transient change outlier** (TC) is similar to a LS but its effect is not permanent and disappears over subsequent observations.

[10] An AO is a **seasonal additive outlier** (SAO) when the additive outlier reappears at regular intervals.

[11] The **innovations** in time series play the same role as errors in cross-sectional analysis (such as OLS).

[12] A LS is called a **seasonal level shift outlier** (SLS) when the mean level shift occurs at regular intervals.

Essentially, we view **anomalies** as outliers; regularly occuring events are part of the the series trend, while rarely occuring deviances from the trend are anomalous.

The rest of this section is devoted to application of R packages to detect outliers and anomalies.

5.2 R Package: `tsoutliers`

The package `tsoutlier` automatically detects outliers in time series based on the procedure outlined in [35]. In this approach, the outlier effects are estimated simultaneously using multiple ARIMA-based regression, and the model parameters and the outlier effects are estimated jointly.

The main interface to the automatic detection procedure is through the funtion `tso(x, types)`, where `x` is a time series object and `types` is a character vector indicating the type of outliers to be considered by the procedure. All five types of outliers described in the previous section can be considered; `AO`, `IO`, `LS`, `SLS` and `TC`. If `types` is not specified, then `AO`, `LS`, and `TC` are be considered by default.

Example 7. The **DJ30** index consists of 30 stocks that are meant to reflect US market performance. Historical data of all stocks currently involved in the Dow Jones Industrial Average is available online on Kaggle [36]. For each of the 30 components of the index, there is one CSV file named by the stock's symbol (e.g. AAPL for Apple Inc.). Each file provides historically adjusted market-wide data (daily, max. 5 years back).

After reading all the CSV files in R *via* `read.csv()`, the daily closing return for each component is sorted by date and stored in a data frame named `data`. Outliers are identified with `tsoutliers`'s `tso()` function. For Apple Inc.'s stock, for instance, the outliers are found to be:

```
tso(ts(data$AAPL),
   types=c("TC","AO","LS","IO","SLS"))
```

```
Outliers:
   type  ind time coefhat  tstat
1    TC  159  159   4.040  4.074
2    AO  302  302  -6.116 -4.405
3    AO  305  305   5.735  4.130
4    AO  410  410  -6.571 -4.732
5    TC  473  473  -4.521 -4.560
6    AO  536  536   6.496  4.678
7    AO  666  666   6.098  4.392
8    AO  791  791   6.629  4.774
9    AO 1043 1043   5.891  4.243
10   AO 1109 1109  -6.633 -4.777
11   AO 1144 1144   7.042  5.072
12   AO 1149 1149  -9.961 -7.174
13   AO 1167 1167   6.833  4.921
14   AO 1238 1238  -5.812 -4.186
```

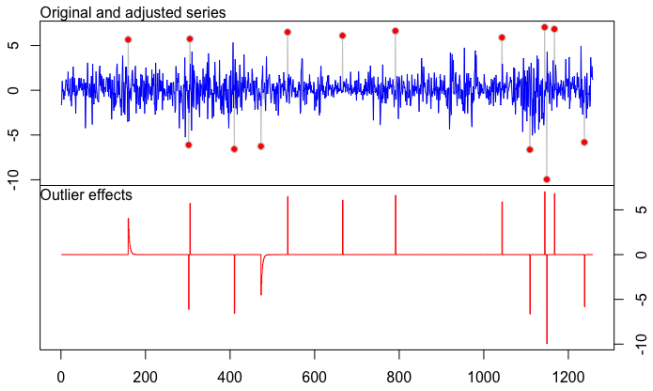


Figure 16. Outliers detection for Apple Inc. daily closing return. The $x-$axis is labelled by the date index.

The algorithm finds 14 outliers, with two being of transient change (TC) type while the rest are additive outliers (AO). The outliers are shown in Figure 16;

The isolated sharp spikes represent AO, while a spike that takes a few periods to disappear represents a TC. Note that the $x-$axis is labelled by the time (date) index, not the actual date. The latter can easily be extracted using the following code:

```
ots=out_tso$outliers
cbind.data.frame(type=ots$type,
   date=data[ots$ind,1])
```

yields

```
   type       date        type       date
1    TC 2015-01-28     8    AO 2017-08-01
2    AO 2015-08-21     9    AO 2018-08-01
3    AO 2015-08-26    10    AO 2018-11-02
4    AO 2016-01-27    11    AO 2018-12-26
5    TC 2016-04-27    12    AO 2019-01-03
6    AO 2016-07-27    13    AO 2019-01-30
7    AO 2017-02-01    14    AO 2019-05-13
```

5.3 R Package: `anomalize`

In this section, we will show how to use `anomalize` to detect anomalies in time series data. The package is available on CRAN, with the latest version always available on github.

It is recommended to first install the package from CRAN (so that the dependencies are also installed locally), then update the package using devtools as shown below:

```
install.packages('anomalize')
library(devtools)
install_github("business-science/anomalize")
library(anomalize)
library(tidyverse)
```

The `anomalize()` function is used to detect outliers in a distribution with no trend or seasonality for tidy data, and returns three columns:

- `remainder_l1` (lower limit for anomalies);
- `remainder_l2` (upper limit for anomalies, and
- `anomaly` (Yes/No).

The first argument of the function is the "tibble" or "tbl_time" object `data`; the second argument is the column `target`, to which the function is applied; the third is the anomaly detection `method`, either `iqr` `geds`.

The **IQR method** (inter-quartile range $Q_3 - Q_1$) is a generalization of Tukey's test (see Section 1). By defaults, the limits are set at 3 times the IQR above Q_3 and below Q_1 (corresponding to $\alpha = 0.05$); anything beyond those limits is considered to be an anomalous observation.

The alpha parameter can be adjusted; at $\alpha = 0.025$, the limits are 6 times the IQR above Q_3 and below Q_1, making it more difficult for data to be an anomaly. Conversely, $\alpha = 0.1$ contracts the limits to 1.5 times the IQR above Q_3 and below Q_1, making it more likely that observations will be deemed anomalous.

The IQR method does not depend on loops and is therefore fast and easily scaled, but it may not be as accurate in detecting anomalies since the high leverage anomalies can skew the centerline (median) of the IQR.

The **GESD method** (generlized extreme studentized deviate test) progressively eliminates outliers using a Student's T—test comparing the test statistic to a critical value. Each time an outlier is removed, the test statistic is updated. Once the test statistic drops below the critical value, all outliers are considered removed.

The α parameter adjusts the width of the critical values. By default, $\alpha = 0.05$. Because this method involves continuous updating via a loop, it is slower than the IQR method. However, it tends to outperform IQR for outlier detection and removal.

Other arguments include `max_anoms` (the maximum percentage of observations that can be identified as anomalies) and `verbose` (boolean linked to the type of output).

Example 8. In the previous sub-section, we used the function `tsoutliers()` to detect outliers of daily closing return for one of the components of DJ30, namely Apple Inc. An alternative manner to detect anomalies in R is to use `anomalize()`, as follows:

```
data_tb=data %>% as.tibble()
data_tb %>%
   time_decompose(AAPL,
         method="stl",
         frequency=10,
         trend="auto") %>%
   anomalize(remainder,
         method="gesd",
         alpha=0.05,
         max_anoms=0.2) %>%
   plot_anomaly_decomposition()
```

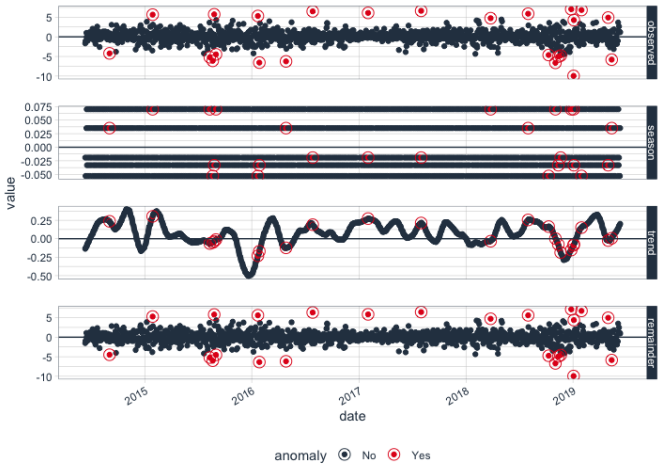


Figure 17. Anomaly detection for Apple Inc.'s daily closing returns, computed with `anomalize()`.

The output is shown in Figure 17, where the top plot displays the observations, the second and third plot displays the trend and seasonality components, respectively, and the bottom plot displays the extracted data on which anomalies are detected. The red markers show the anomalies found by `anomalize()`. The recomposed series can also be plotted *via* `time_recomposed()`:

```
data_tb %>% time_decompose(AAPL) %>%
  anomalize(remainder) %>%
  time_recompose() %>%
  plot_anomalies(time_recomposed=TRUE,
         ncol=3,
         alpha_dots=0.5)
```

The output is shown in Figure 18. Anomalous points that are shown in red can be extracted by the following code:

```
anomalies=data_tb %>%
time_decompose(AAPL) %>%
 anomalize(remainder) %>%
 time_recompose() %>%
 filter(anomaly=='Yes')
```

The output is a time tibble with 16 rows (with `date` and `observed` as the first two variables); these are the 16 observations that `anomalize()` reports. Recall that in previous section, `tsoutliers()` detected 14 outliers.

It is interesting to compare the outlier/anomaly dates for Apple Inc. data from both approaches. This can be done with the code on the following page:

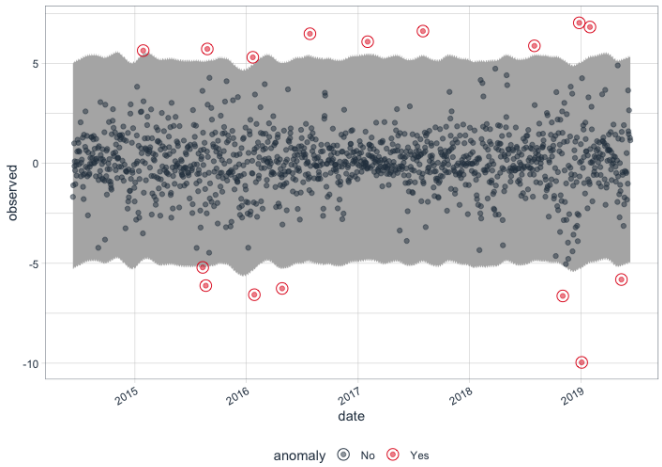


Figure 18. Anomaly detection for Apple Inc.'s daily closing returns with recomposed data; the grey portion represent the expected (normal) trend.

```
odates=cbind.data.frame(type=ots$type,
            date=data[ots$ind,1])
adates=cbind.data.frame(date=anomalies$date,
        observed=anomalies$observed)
left_join(adates,odates,by="date")
```

The list of detected outliers is found below:

```
         date observed type
 1 2015-01-28    5.653   TC
 2 2015-08-11   -5.204 <NA>
 3 2015-08-21   -6.116   AO
 4 2015-08-26    5.735   AO
 5 2016-01-22    5.317 <NA>
 6 2016-01-27   -6.571   AO
 7 2016-04-27   -6.258   TC
 8 2016-07-27    6.496   AO
 9 2017-02-01    6.098   AO
10 2017-08-01    6.629   AO
11 2018-08-01    5.891   AO
12 2018-11-02   -6.633   AO
13 2018-12-26    7.042   AO
14 2019-01-03   -9.961   AO
15 2019-01-30    6.833   AO
16 2019-05-13   -5.812   AO
```

All the observations that have been detected by the original approach (`tsoutliers()`) match the ones reported by this new approach (`anomalize()`), but `anomalize()` also detects two extra points, for which the type cannot be specified. The comparison is also illustrated in Figure 19.

5.4 Summary

For time series data, anomaly detection is usually performed on time series **remainders**, where both the **seasonal** and the **trend** components were removed; the former is the presence of variations that occur at specific regular intervals shorter than a year, such as daily, weekly, monthly, or quarterly while the later consists of longer term growth patterns.

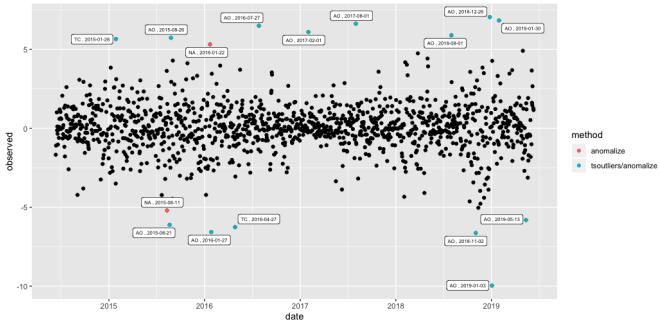


Figure 19. Comparing two methods of outlier/anomaly detection for Apple Inc.'s daily closing returns.

The first task in the anomaly detection of a time series is thus to generate its remainders.

There are different ways to decompose a time series to produce remainders: ARIMA and X12 are popular algorithms to do so.[13]

In general, high performance machine learning techniques are not recommended for anomaly detection since the overfitting reduces the difference between the observed and fitted values whereas in anomaly detection this difference is essential to highlight the anomaly.

On the other hand, seasonal decomposition performs best for this task by removing the right features (i.e. seasonal and trend components) while preserving the characteristics of anomalies in the remainders.

Finally, we note that there are other popular R outlier analysis packages in R, such as `AnomalyDetection`, which uses a method similar to `anomalize`'s GESD. Interested readers are invited to try this package and compare their results with the functions reviewed in this section.

6. Project

There is no substitute for practice: for the accompanying project, we ask you to test the performance and limitations of four outlier detection algorithms (LOF, IsolationForest, *k*NN, and PCA) on five datasets:

- the **Airline** dataset;
- the **Distracted Driving Fatality** dataset;
- the **House Prices** dataset;
- the **Melbourne Temperature** dataset, and
- the **Sale Transactions** dataset.

Consult the project statement for a series of guided steps.

[13] `tsoutliers()` uses ARIMA while `anomalize()` uses seasonal decomposition.

References

[1] Aggarwal, C.C. [2017], *Outlier Analysis* (2nd ed.), *Springer.*

[2] Aggarwal, C. C. et YU, P. S. (2001). Outlier detection for high dimensional data. In ACM Sigmod Record, pages 37–46. ACM.

[3] Maimon, O., Rokach, L. [2010], *Data Mining and Knowledge Discovery Handbook*, *Springer*.

[4] Prasanta, G., et al. [2011], A Survey of Outlier Detection Methods in Network Anomaly Identification. *Oxford University Press, 54 (4), 570–588*

[5] Ranga, S.N.N.R., [2019], Outlier Detection: Techniques and Applications: A Data Mining Perspective. *Springer Nature Switzerland AG; 1st ed*

[6] Manish, G., et al., [2014], Outlier Detection for Temporal Data, IEEE Transactions on Knowledge and Data Engineering 26(9), 2250–2267.

[7] Kandanaarachchi, S., Hyndman, R.J. [2019], Dimension reduction for outlier detection using DOBIN, Monash Business School.

[8] Priyanga, D.T., et al., [2019], Anomaly detection in high-dimensional data.

[9] Aggarwal, C.C., Sathe, S. [2017], Outlier Ensembles, an Introduction, *Springer.*

[10] Aurore Archimbaud (2018). Détection non-supervisée d'observations atypiques en contrôle de qualité : un survol. Journal de la Société Française de Statistique, Vol. 159 No. 3 1-39

[11] Badr, W., **5 ways to detect outliers that every data scientist should know**, towardsdatascience.com

[12] Mehrotra, K.G., Mohan, C.K., Huang, H. [2017], Anomaly Detection Principles and algorithms, *Springer*.

[13] Arora, L. [2019], **An Awesome Tutorial to Learn Outlier Detection in Python using PyOD Library**, on Analitics Vidhya.

[14] Santoyo, S. [2017], **A Brief Overview of Outlier Detection Techniques**, on towardsdatascience.com.

[15] He, Z., Deng, S., Xu, X. [2005], A Unified Subspace Outlier Ensemble Framework for Outlier Detection, Advances in Web Age Information Management.

[16] Lazarevic, A., Kumar, V. [2005], Feature Bagging for Outlier Detection, ACM KDD Conference.

[17] Fei, T.L., Ting, K.M., Zhou, Z.H. [2008], Isolation Forest, 2008 Eighth IEEE International Conference on Data Mining: 413–422.

[18] Ester, M., Kriegel, H.P., Sander, J., Xu, X. [1996], A density-based algorithm for discovering clusters in large spatial databases with noise, AAAI Press: 226–231.

[19] Orchard, T., Woodbury, M. [1972], A Missing Information Principle: Theory and Applications, Berkeley Symposium on Mathematical Statistics and Probability, University of California Press.

[20] Torgo, L. [2017], Data Mining with R (2nd edition), CRC Press.

[21] Chandola, V., Banerjee, A., Kumar, V. [2007], Outlier detection: a survey, Technical Report TR 07-017, Department of Computer Science and Engineering, University of Minnesota.

[22] Hodge, V., Austin, J. [2004], A survey of outlier detection methodologies, Artif.Intell.Rev., 22(2):85-126.

[23] **Height Percentile Calculator, by Age and Country**, on tall.life

[24] Hariri, S., Kind, M. C., Brunner, R. J. [2018], Extended Isolation Forest, Computing Research Repository.

[25] Campello, R., Moulavi, D., Sander, J. [2013], Density-Based Clustering Based on Hierarchical Density Estimates, Advances in Knowledge Discovery and Data Mining, Springer Berlin Heidelberg: 160–172.

[26] **How HDBSCAN Works** [2016] McInnes, L., Healy, J., Astels, S.

[27] Findley, D.F., Hood, C.C., X-12-ARIMA and its Application to Some Italian Indicator Series, U.S. Bureau of the Census.

[28] Findley, D.F., Monsell, B.C., Bell, Otto and Chen [1998], New Capabilities and Methods of the X-12-ARIMA Seasonal Adjustment Program, U.S. Bureau of the Census.

[29] An Introductory Course on Time Series Analysis, Australian Bureau of Statistics.

[30] Seasonal Adjustment of Economic Time Series, Singapore Department of Statistics.

[31] T.Jackson, M.Leonard, Seasonal Adjustment Using The X12 Procedure, SAS Institute.

[32] **Matthews Correlation Coefficient** (MCC) on Wikipedia.

[33] **Principal Component Analysis** on Wikipedia

[34] Ruey S. Tsay [1988], level shifts, and variance changes in time series.

[35] Chen, C. and Liu, Lon-Mu (1993). Joint Estimation of Model Parameters and Outlier Effects in Time Series. Journal of the American Statistical Association, 88(421), pp. 284-297. doi: 10.2307/2290724

[36] **EOD data for all Dow Jones stocks**

[37] Zhang, J., Lou, M., Ling, T. W. et Wang, H.(2004). Hosminer : a system for detecting outlyting subspaces of high-dimensional data. In Proceedings of the Thirtieth International Conference on Very Large Data Bases, volume 30, pages 1265–1268. VLDB Endowment

[38] Zimek, A., Kriegel, H.-P., Kröger, P., Schubert, E. (2009). Outlier detection in axis-parallel subspaces of high dimensional data. In Pacific-Asia Conference on Knowledge Discovery and Data Mining, pages 831–838. Springer.

[39] Kriegel H.-P. et al. (2000) "LOF: Identifying density-based local outliers," in Proceedings of the ACM SIGMOD International Conference on Management of Data (ACM, New York), pp. 93-104

[40] Müller, E., Assent, I., Iglesias S, P., Mulle, Y. et Bohm, K. (2012). Outlier ranking via subspace analysis in multiple views of the data. In IEEE 12th International Conference on Data Mining (ICDM), pages 529-538. IEEE.

[41] Müller, E., Assent, I., Steinhausen, U. et Seidl T. (2008). OutRank : ranking outliers in high dimensional data. In ICDEW 2008, IEEE 24th International Conference on Data Engineering Workshop, pages 600–603. IEEE

[42] Müller, E., Schiffer, M. et Seidl, T. (2010b). Adaptive outlierness for subspace outlier ranking. In Proceedings of the 19th ACM International Conference on Information and Knowledge Management, pages 1629–1632. ACM.

[43] Müller, E., Schiffer, M. et Seidl T. (2011). Statistical selection of relevant subspace projections for outlier ranking. In IEEE 27th International Conference on Data Engineering (ICDE), pages 434–445. IEEE.

[44] Jian T., Zhixiang C., Ada W. F., David, W C., Capabilities of outlier detection schemes inlarge datasets, framework and methodologies, Springer-Verlag London Limited 2006

[45] **ROC curve**

[46] He, Z., Xu, X., Deng, S. [2005], **A Fast Greedy Algorithm for Outlier Mining**.

[47] Hastie, T., **Leukemia dataset**.

[48] Leduc, O., Macfie, A., Maheshwari, A., Pelletier, M., Boily, P. [2019], Feature Selection and Dimension Reduction, *Data Science Report Series*, Data Action Lab blog.

[49] Leduc, O., Boily, P. [2019], **Boosting with AdaBoost and Gradient Boosting**, Data Action Lab blog.

[50] Lei, X. [2020], **Distributed LOF: Density-Sensitive Anomaly Detection With MapReduce**, on medium.com.

[51] Krishna, G. [2020] **Performing Real-time Anomaly Detection using AWS**, on medium.com.

[52] Baron, D. [2018], **Outlier Detection**, XXX Winter School of Astrophysics on Big Data in Astronomy, GitHub repository.

[53] Baron, D. [2016], **Outlier Detection Algorithm on Galaxy Spectra**, GitHub repository.