

Anomaly Detection and Outlier Analysis

27

by Patrick Boily, with contributions from Youssouph Cissokho, Soufiane Fadel, and Richard Millson

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. [335–337]. 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 recently.

In this chapter, we review various detection methods, with particular attention paid to both supervised and unsupervised methods.

27.1 Overview¹

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.²

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

27.1.1 Basic Notions and Concepts

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.³

27.1 Overview	1305
Basic Notions and Concepts	1305
ML Framework	1310
Motivating Example	1317
27.2 Quantitative Approaches	1320
Distance Methods	1320
Density Methods	1331
27.3 Qualitative Approaches	1345
AVF Algorithm	1346
Greedy Algorithm	1347
27.4 High-Dimensional Data	1348
Definitions and Challenges	1349
Projection Methods	1349
Subspace Methods	1359
Ensemble Methods	1360
27.5 Exercises	1364

1: This section is an extension of Section 15.5.

2: Throughout, the definitions of terms like **normal** and **anomalous** will be kept purposely vague, to allow for increased flexibility.

3: Outlying observations may be anomalous along any of the individual variables, or in combinations of variables.

Observations could be anomalous in one context, but not in another.

Consider, for instance, an adult male who is 6-side tall. Such a man would fall in the 86th percentile among Canadian males [185], which, while on the tall side, is not unusual; in Bolivia, however, the same man would land in the 99.9th percentile, which would mark him as extremely tall and quite dissimilar to the rest of the population.

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?

In practice, an **outlier/anomalous observation** may arise as

- a **“bad” object/measurement**: data artifacts, spelling mistakes, poorly imputed values, etc.
- a **misclassified observation**: according to the existing data patterns, the observation should have been labeled differently;
- an observation whose measurements are found in the **distribution tails**, of a large enough number of features;
- an **unknown unknowns**: a completely new type of observations whose existence was heretofore unsuspected.

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.

4: Such as data transformation strategies.

When influential observations are identified, **remedial measures**⁴ 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.

5: And shrouded in **uncertainty** due to their relatively low numbers.

Anomaly Detection By definition, anomalies are **infrequent**,⁵ which makes it difficult to distinguish 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 box-plots, 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. These methods usually find the anomalies that **shout the loudest** [338].

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 observations (*cf.* **leverage**).

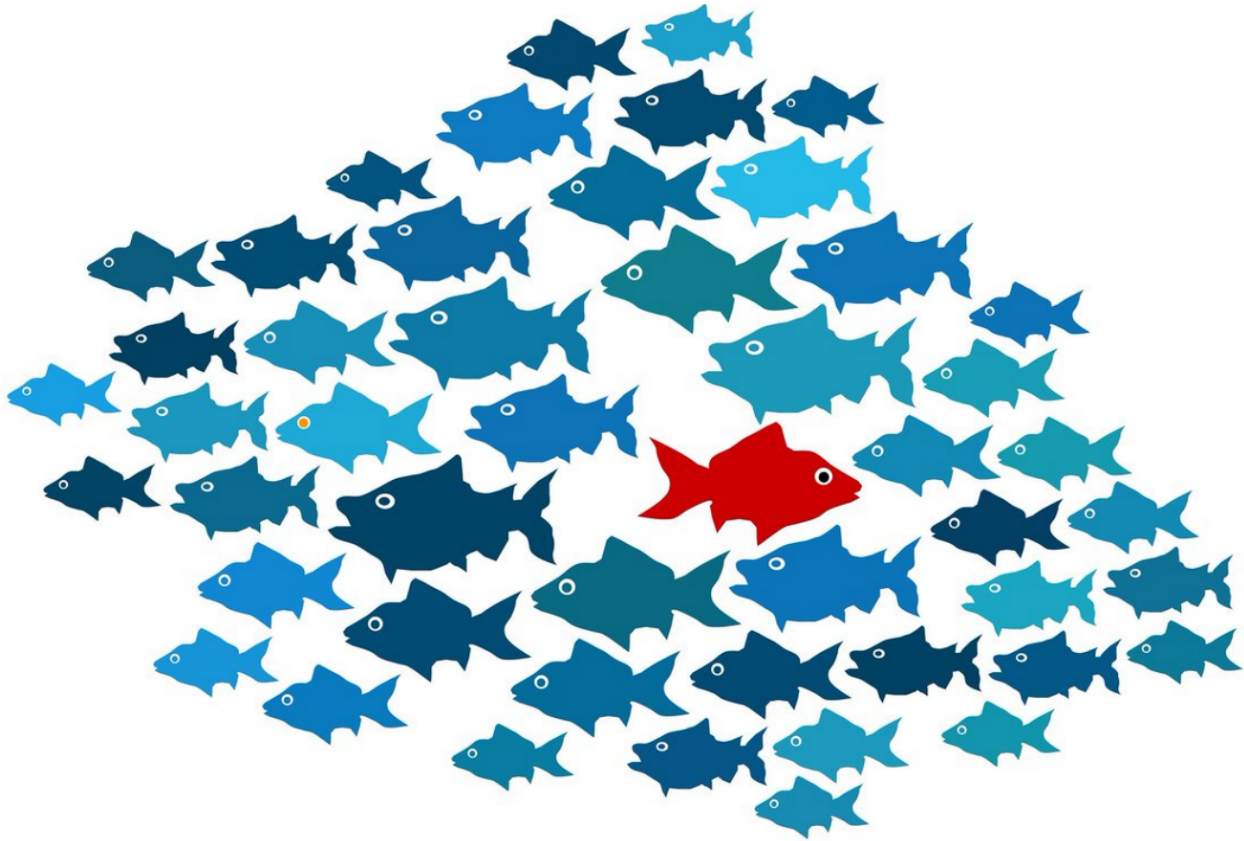


Figure 27.1: A school of fish: what jumps at you here? [author unknown]

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. 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.

In the early stages of anomaly detection, **simple data analyses** (such as descriptive statistics, 1- and 2-way tables, and traditional visualizations) 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.⁶

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 learning (SL) 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** (or class imbalance) problem.

6: Which, by the way, should always be seen as a welcomed development.

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.

For the **security agency**, the cost of wrongly thinking that a passenger:

- is smuggling a weapon \implies cost of a single search;
- is NOT smuggling a weapon \implies catastrophe (potentially).

The wrongly targeted individuals may have a . . . somewhat different take on this, however, either from a societal or a personal perspective.⁷

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

As an example, if all participants in a workshop except for one can view the video conference lectures, then the one individual/internet connection/computer is **anomalous** – it behaves in a manner which is different from the others.

It is **very important to note** that this **DOES NOT** mean that the different behaviour is the one we are actually interested in/searching for! In Figure 27.1, perhaps we were interested in the slightly larger red fish that swims in a different direction than the rest of the school, but perhaps we were really interested in the regular-sized teal fish that swims in the same direction as the others but that has orange eyes (can you spot it?).

Outlier Tests The following traditional methods and tests of outlier detection fall into this category:⁸

- 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 27.2 (a concrete example is provided in Section 15.5).

- The **Grubbs test** is another univariate test, which takes into consideration the number of observations in the dataset:

H_0 : no outlier in the data against H_1 : exactly one outlier in the data.

7: Or both.

8: Note that **normality** of the data is an assumption for some of them; how robust these tests are against departures from this assumption depends on the situation.

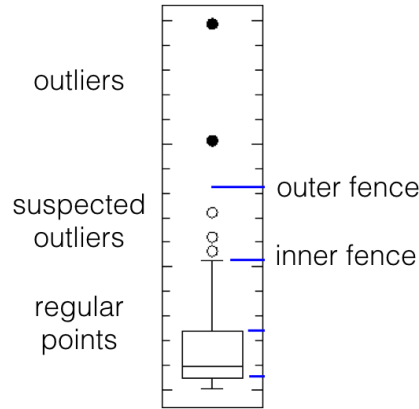


Figure 27.2: Tukey's boxplot test; suspected outliers are marked by white disks, outliers by black disks.

Let x_i be the value of feature X for the i^{th} unit, $1 \leq i \leq n$, let (\bar{x}, s_x) be the mean and standard deviation of feature X , let α be the desired significance level, and let $T(\frac{\alpha}{2n}, n)$ be the critical value of the Student t -distribution at significance $\frac{\alpha}{2n}$. The test statistic is

$$G = \frac{\max\{|x_i - \bar{x}| : i = 1, \dots, n\}}{s_x} = \frac{|x_{i^*} - \bar{x}|}{s_x}.$$

Under H_0 , G follows a special distribution with critical value

$$\ell(\alpha; n) = \frac{n-1}{\sqrt{n}} \sqrt{\frac{T^2(\frac{\alpha}{2n}, n)}{n-2 + T^2(\frac{\alpha}{2n}, n)}}.$$

At significance level $\alpha \in (0, 1)$, we reject the null hypothesis H_0 in favour of the alternative hypothesis that x_{i^*} is the **unique outlier along feature** if $G \geq \ell(\alpha; n)$. If we are looking for more than one outlier, it can be tempting to classify every observation x_i for which

$$\frac{|x_i - \bar{x}|}{s_x} \geq \ell(\alpha; n)$$

as an outlier, but this approach is **contra-indicated**.

▪ Other common tests include:

- 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;
- visual outlier detection (see Section 15.5 for some simple examples).

27.1.2 Statistical Learning Framework

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

$$P(\text{observation is anomalous} \mid I) > P(\text{observation is not anomalous} \mid I)?$$

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.

A Time Series Detour For time series data, this means that it may be necessary to first perform trend and seasonality extraction.⁹

Example: 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 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? Are either of the Feb 2013 or the July 2017 performances anomalous?

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.

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.

9: More information on these topics can be obtained in Chapter 9.

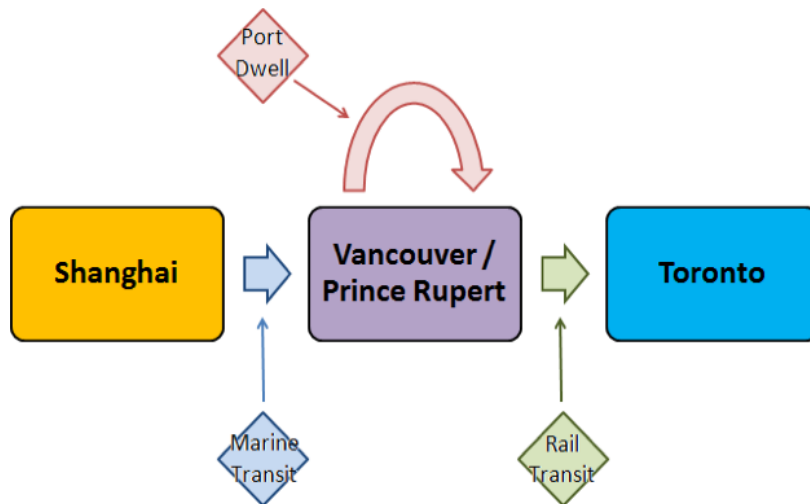


Figure 27.3: Multi-modal supply chain corridor.

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.

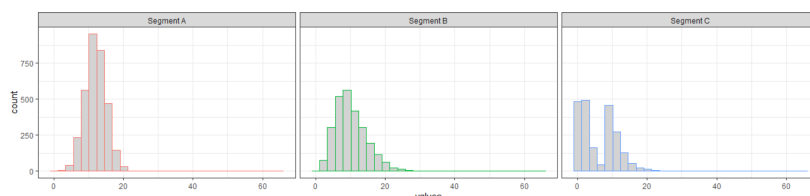
Each segment's performance is measured using **fluidity indicators**,¹⁰ which are computed using various statistics of the transit/dwelling time distributions for each of the supply chain segments, such as:

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.

10: In this case, compiled at a monthly scale.



Segmt	Freq	Mean	SD	C05	C95	RI	BI	CV
A	3286	12.10	3.33	7.06	17.00	2.41	0.41	0.27
B	2594	10.09	4.43	3.88	18.20	4.69	0.80	0.44
C	2142	5.96	5.08	0.19	14.40	77.12	1.41	0.85

Figure 27.4: Illustration of how to derive the various monthly fluidity indicators.

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);
- **irregular component**.

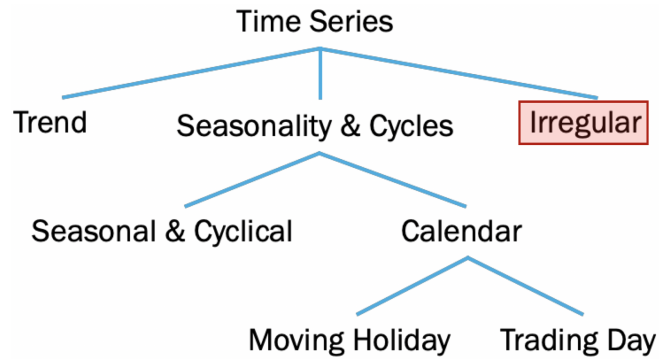


Figure 27.5: Components of a time series; the irregular component should be seasonally adjusted.

The trend and the seasonal components provide the **expected behaviour** of the indicator time series; 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.¹¹ The **irregular component** arises 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.¹² 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:

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

The multiplicative and pseudo-additive models are defined in similar ways (again, consult Chapter 9 for details).¹³

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 27.6. The size of the peaks and troughs seems fairly

11: Sources of these breaks may come from changes in government policies, strike actions, exceptional events, inclement weather, etc.

12: X12 is implemented in SAS and R, among other platforms.

13: 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.

CV data, from

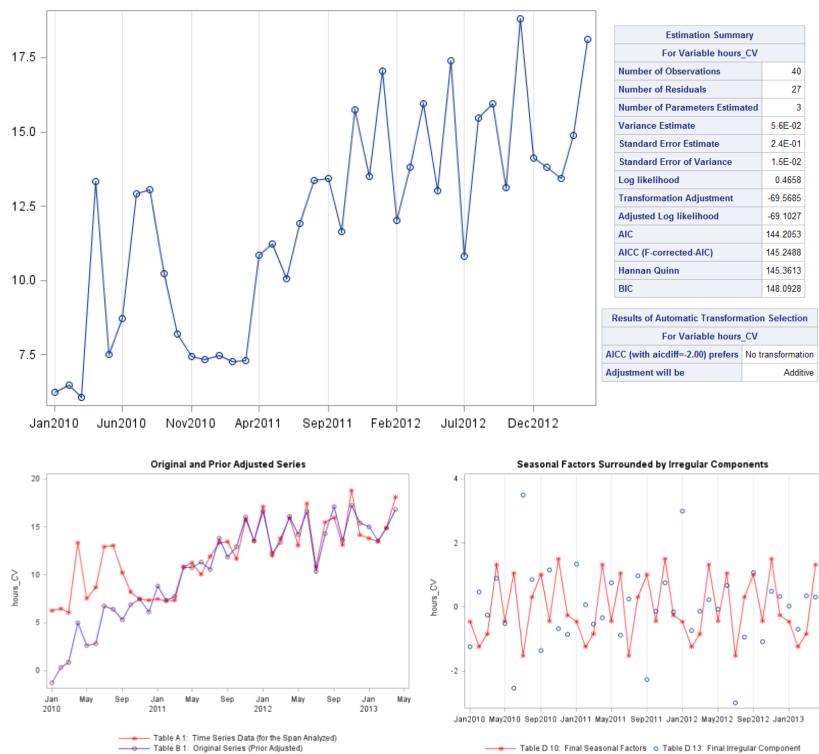


Figure 27.7: Diagnostic plot for marine transit CV data, from 2010 to 2013 (left); SI chart (right).

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 27.7: 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 in Figure 27.8.¹⁴ It is on the irregular component that detection anomaly would be conducted.

14: The trend and irregular components are also shown separately for readability.

This example showcases the importance of **domain understanding** and data preparation to the anomaly detection process. As the vast majority of observations in a general problem are typically “normal”, we can also view anomaly detection as a **rare occurrence learning** classification problem or as a **novelty detection** data stream problem.¹⁵

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**.

15: We discussed the former in Chapter 21; the latter will be tackled in Chapter 24.

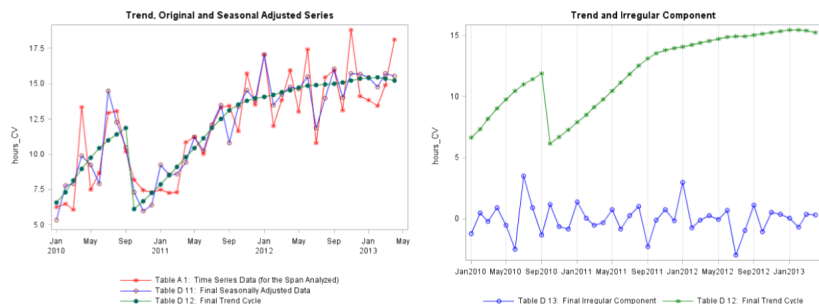


Figure 27.8: Adjusted plot for marine transit CV data, from 2010 to 2013.

Basic Concepts A generic system (such as the monthly transit times example) may be realized in **normal** states or in **abnormal** states. Normality, perhaps counter-intuitively, 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 [339] 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 are likely to be perturbed as well; if anomalies arise from perturbed processes, being able to identify when the process is abnormal,¹⁶ may lead to **useful anomaly detection**.

16: That is to say, being able to capture the various normal and abnormal processes.

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

Table 27.1: Confusion matrix for anomaly detection

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

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

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.

Assume that for a testing set with $\delta = FN + TP$ **true outliers**, an anomaly detection algorithm identifies $\mu = FP + TP$ **suspicious observations**, of which $\nu = TP$ are **known** to be true outliers. Performance evaluation in this context is often measured using **precision** and **recall**.

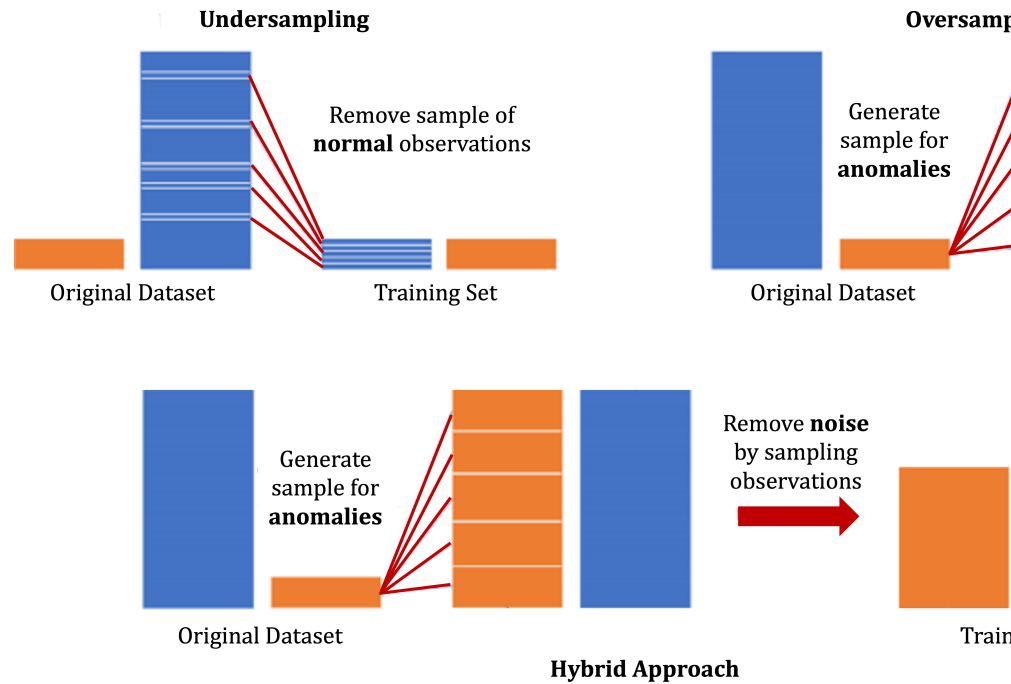


Figure 27.9: Oversampling, undersampling, and hybrid strategy for anomaly detection [342].

When the δ actual anomalies are ranked near the top δ suspicious ones, $RP \approx 1$. The metric is well-defined only when $\mu \geq \delta$; as with most performance evaluation metrics, it is meaningful only in **comparison** with the performance of other algorithms.¹⁹

19: Other SL performance evaluation metrics include:

- **AUC** – the probability of ranking a randomly chosen anomaly higher than a randomly chosen normal observation (higher is better);
- **probabilistic AUC** – a calibrated version of AUC.

The **rare occurrence** problem can be tackled by using:

- a **manipulated training set** (oversampling, undersampling, generating artificial instances);
- **specific SL AD algorithms** (CREDOS, PN, SHRINK);
- **boosting algorithms** (SMOTEBoost, RareBoost);
- **cost-sensitive classifiers** (MetaCost, AdaCost, CSB, SSTBoost),
- etc. [341]

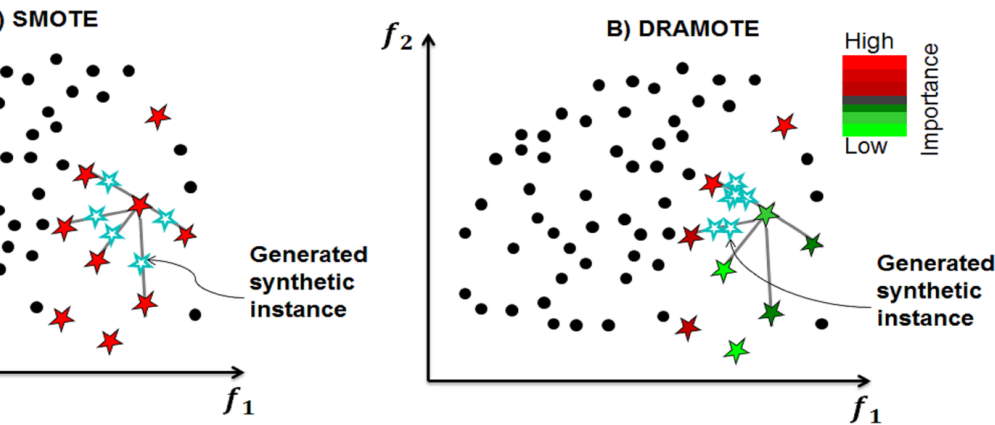
The rare (anomalous) class can be **oversampled** by duplicating the rare events until the data set is **balanced** (roughly the same number of anomalies and normal observations). This does not increase the overall level of information, but it will increase the mis-classification cost.

The majority class (normal observations) can also be **undersampled** by randomly removing:

- “**near miss**” observations or
- observations that are “**far**” from anomalous observations.

Some loss of information has to be expected, as are “overly general” rules. Common strategies are illustrated in Figures 27.9 and Figure 27.10.

Another modern approach rests on the concept of **dimension reduction** (see Chapter 23); **autoencoders** learn a compressed representation of the data. In a sense, the **reconstruction error** measures how much information is lost in the compression.



Artificial cases with SMOTE and DRAMOTE [343].

Anomaly detection algorithms are then applied to the compressed data: we look for anomalous patterns or anomalous reconstruction errors.

In the example of Figure 27.11, one observation is anomalous because its compressed representation does not follow the pattern of the other 8 observations, whereas another observation is anomalous because its reconstruction error is substantially higher than that of the other 8 observations.²⁰ We discuss autoencoders in more detail in Chapter ??.

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 19.18 for an illustration).²¹

27.1.3 Motivating Example

In this chapter, we will illustrate the concepts and the algorithms of anomaly detection on an artificial dataset.

Consider a dataset of 102 observations in \mathbb{R}^4 ; the first 100 observations $\mathbf{p}_1, \dots, \mathbf{p}_{100}$ are drawn from a multivariate $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, with

$$\boldsymbol{\mu} = (1, -2, 0, 1), \quad \boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0.5 & 0.7 & 0.5 \\ 0.5 & 1 & 0.95 & 0.3 \\ 0.7 & 0.95 & 1 & 0.3 \\ 0.5 & 0.3 & 0.3 & 1 \end{pmatrix}.$$

Setting-up the data

```
nobs = 100
mu = matrix(rep(c(1,-2,0,1),100), nrow=4)
Sigma = matrix(c(1, 0.5, 0.7, 0.5,
                0.5, 1, 0.95, 0.3,
                0.7, 0.95, 1, 0.3,
                0.5, 0.3, 0.3, 1), nrow=4, ncol=4)
```

We use $\boldsymbol{\Sigma}$'s Cholesky decomposition to generate random observations.

20: Can you hazard a guess as to which one is which?

21: There are a number of challenges associated to unsupervised anomaly detection, 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 may lead to different cluster assignments, as discussed in Chapter 22.

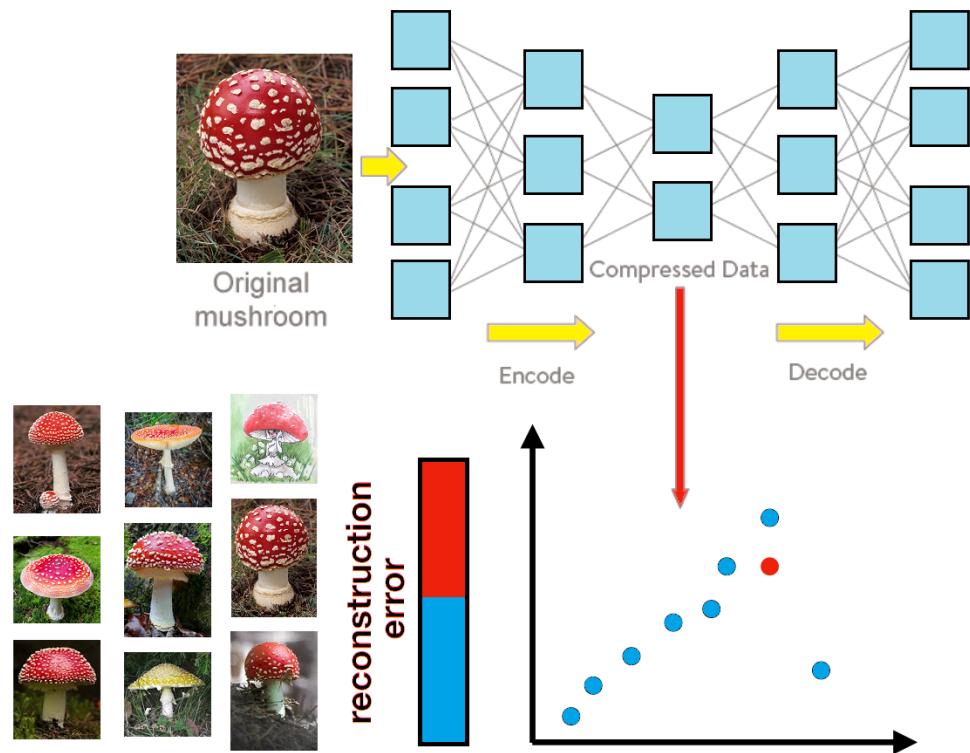


Figure 27.11: Illustration of autoencoder compression/reconstruction for anomaly detection, modified from [3]

Cholesky decomposition

```
L = chol(Sigma)
nvars = dim(L)[1]

set.seed(0) # for replicability
r = t(mu + t(L) %*% matrix(rnorm(nvars*nobs),
                           nrow=nvars, ncol=nobs))
```

The summary statistics for the 100 “regular” observations are given below:

Summarizing data

```
rdata = as.data.frame(r)
names(rdata) = c('x1', 'x2', 'x3', 'x4')
summary(rdata)
```

	x1	x2	x3	x4
Min.	:-1.90	Min. :-4.41	Min. :-2.53	Min. :-1.99
1st Qu.:	0.38	1st Qu.: -2.65	1st Qu.: -0.62	1st Qu.: 0.34
Median :	0.93	Median : -2.02	Median : -0.05	Median : 0.94
Mean :	0.94	Mean : -1.98	Mean : 0.01	Mean : 0.94
3rd Qu.:	1.46	3rd Qu.: -1.40	3rd Qu.: 0.63	3rd Qu.: 1.59
Max. :	3.44	Max. : 0.52	Max. : 2.03	Max. : 2.81

We now add two observations $\mathbf{z}_1 = (1, 1, 1, 1)$ and $\mathbf{z}_4 = (4, 4, 4, 4)$ not arising from $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

Setting-up some outliers

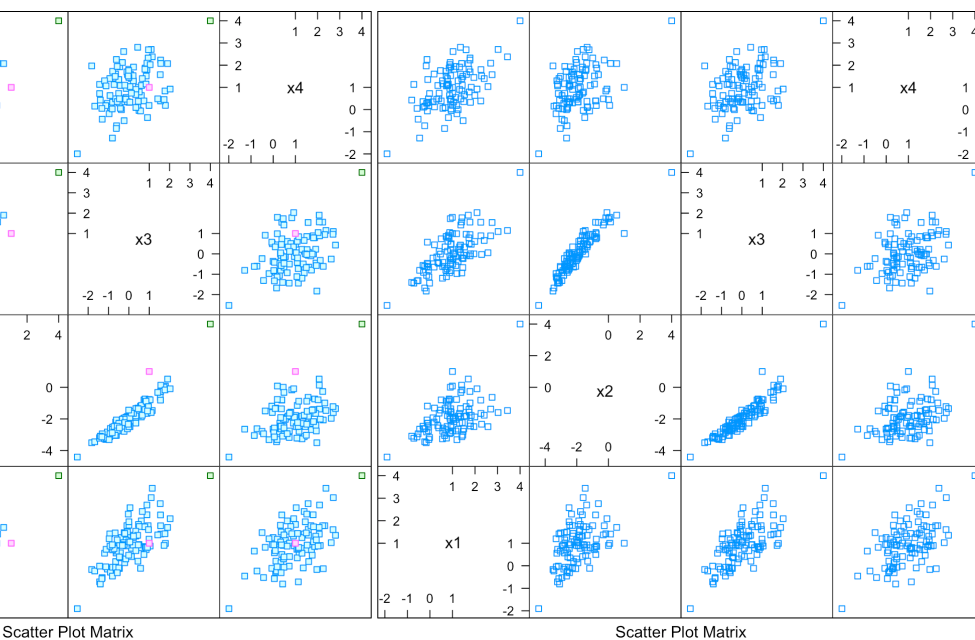
```
pt.1 = c(1,1,1,1)
pt.2 = c(4,4,4,4)

rdata = rbind(rdata,pt.1,pt.2)
group = c(rep(1,nobs),2,3)
rdata = cbind(rdata,group)
```

The complete dataset is displayed below, with z_1 in pink and z_4 in green. But since we will not usually know which observations are “regular” and which are “anomalous”, let us remove the colouring.

Plotting and anonymizing the data

```
lattice::splom(rdata[,1:4], groups=group, pch=22)
lattice::splom(rdata[,1:4], pch=22)
```



Evidently, a visual inspection suggests that there are in fact 3 outliers in the dataset: the two that were specifically added as such, but a 3rd observation that was naturally outlying!

Multiple references were consulted in the preparation of this chapter, in particular [335, 339]. Other good **survey documents** include [344, 345]. Specific methods and approaches are the focus of other papers: [346–348] (**high-dimensional data**), [349] (**DOBIN**), [350] (**outlier ensembles**), [351, 352] (**isolation forest**), [353, 354] (**DBSCAN**), [355] (**LOF**), [356–360] (**subspace method**), [361] (**time series data**).

On the practical side, we would be remiss if we did not also mention [362], but keep in mind that there is a plethora of quality **online anomaly detection tutorials** in the programming language of your choice.

27.2 Quantitative Approaches

Cluster-based methods are not the only types of UL anomaly detection methods. Generally, they come in two flavours: **distance-based** and **density-based**.

- **Distance-based methods** include distance to all observations, distance to k nearest neighbours (k NN), average distance to k NN, median distance to k NN, etc.
- **Density-based methods** include local outlier factor (LOF), isolation forest, HDBSCAN, etc.

27.2.1 Distance 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 observations to use in this comparison distinguishes the different distance-based algorithms.

Notation Let $D \subset \mathbb{R}^n$ be an n -dimensional dataset, $\mathbf{p}, \mathbf{q} \in D$, $P \subset D$ be a subset of D . Assume that $d : D \times D \rightarrow \mathbb{R}$ gives the distance between \mathbf{p} and \mathbf{q} , written $d(\mathbf{p}, \mathbf{q})$.

An anomaly detection algorithm provides a function $a : D \rightarrow \mathbb{R}$ that describes how anomalous a given observation is. This induces an ordering on the observations 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 an observation 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 observations \mathbf{p}, \mathbf{q} as

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

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

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 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

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

for continuous random variables, it is defined by

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

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

$$\sum_{\Omega} \sqrt{f_X f_Y} = 1 \quad \text{or} \quad \int_{\Omega} \sqrt{f_X f_Y} dz = 1$$

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 $f_Y^* = \sqrt{f_Y}$:

$$\begin{aligned} 0 &\leq \int_{\Omega} \sqrt{f_X f_Y} dz \\ &= \int_{\Omega} f_X^* f_Y^* dz \\ &\leq \left(\int_{\Omega} |f_X^*|^2 dz \right)^{1/2} \left(\int_{\Omega} |f_Y^*|^2 dz \right)^{1/2} \\ &= \left(\int_{\Omega} f_X dz \right)^{1/2} \left(\int_{\Omega} f_Y dz \right)^{1/2} = 1; \end{aligned}$$

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 observation \mathbf{p} to represent a probability distribution. In that case, the Hellinger distance between that observation and any other distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ can be studied using the framework above, using the **Mahalanobis distance**:

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

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

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

Example In general, we do not know the true mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ from which the data could arise, and the mean vector and the covariance structure must be estimated from the data.

In the example of Section 27.1, we have:

```
(mu.1 <- colMeans(rdata[,1:4]))
cov(rdata[,1:4])
```

```

           x1           x2           x3           x4
0.96801577 -1.89096069  0.05602349  0.97433766

           x1           x2           x3           x4
x1 0.8999038 0.5690744 0.6646093 0.5033570
x2 0.5690744 1.3124241 1.0685066 0.4694309
x3 0.6646093 1.0685066 0.9921537 0.3969461
x4 0.5033570 0.4694309 0.3969461 0.9043493
```

These are distinct from the true underlying collection of parameters μ and Σ , but close enough to be explained by **sampling variation** and because $\mathbf{z}_1, \mathbf{z}_4 \sim \mathcal{N}(\mu, \Sigma)$.

We first attempt to identify the anomalous observations by computing the Mahalanobis distance from the empirical distribution to all observations in the dataset.

```

Sigma.inv = matlib::inv(cov(rdata[,1:4]))

M_d<-vector()

for(j in 1:nrow(rdata)){
  M_d[j] <- sqrt(as.matrix(rdata[j,1:4]-mu.1) %*%
                  Sigma.inv %*%
                  t(as.matrix(rdata[j,1:4]-mu.1)))
}

rdata <- data.frame(rdata,M_d)
summary(M_d)
```

```

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.4622  1.3479  1.6764  1.7980  2.1010  6.6393
```

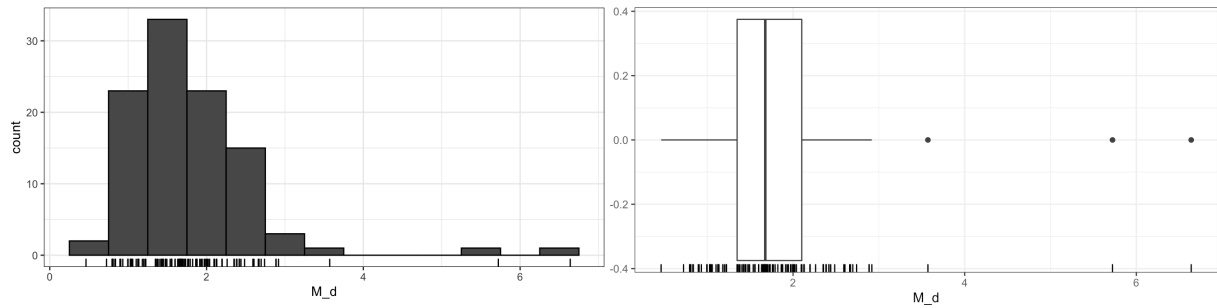
The summary suggests that there is (at least) one observation for which the Mahalanobis distance to the empirical distribution is quite high.

```

library(dplyr) # we always assume that these
library(ggplot2) # two packages have been loaded

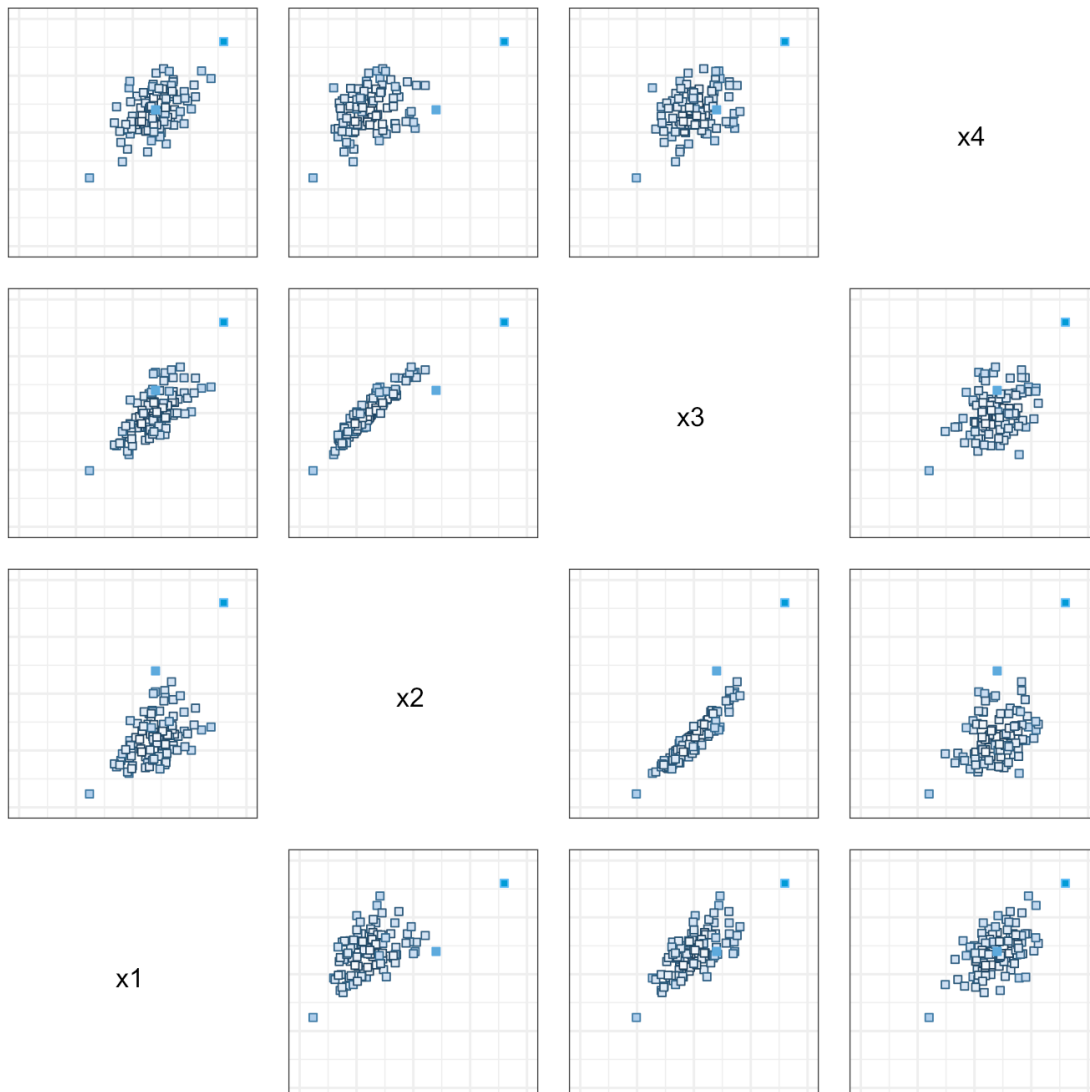
rdata |> ggplot(aes(x=M_d)) +
  geom_histogram(colour="black",binwidth = 0.5) +
  geom_rug() + theme_bw()

rdata |> ggplot(aes(x=M_d)) +
  geom_boxplot() + geom_rug(color="black")
```



The histogram of Mahalanobis distances shows that most observations are fairly “regular”, but that two of the observations have substantially larger distances. The boxplot confirms it, but identifies a potential third outlying observation.

Below, we display the scatter plot matrix of the 102 observations, with colour intensity mapped to the Mahalanobis distance of the observation from the empirical distribution (the code is omitted for readability).



It certainly seems as though z_1 and z_4 could be the two anomalies.

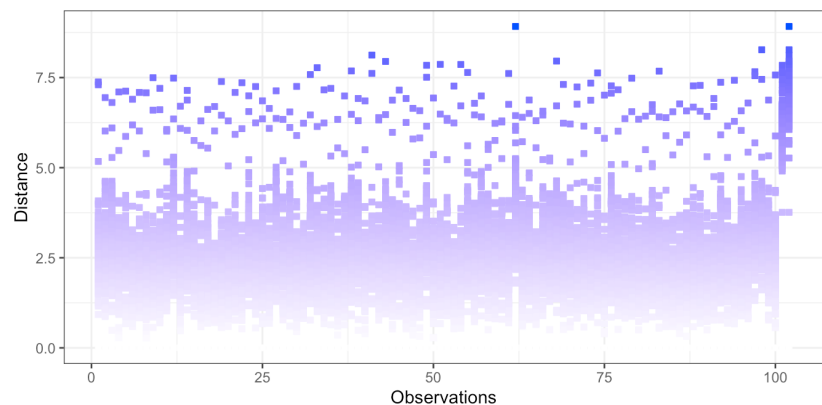
Next, we plot the Mahalanobis distance from each observation to every other observation.

```
M_pq<-matrix(nrow=nrow(rdata), ncol=nrow(rdata))

for(j in 1:nrow(rdata)){
  for(i in 1:nrow(rdata)){
    M_pq[j,i]<-sqrt(as.matrix(rdata[j,1:4]-rdata[i,1:4]) %*%
                    Sigma.inv %*%
                    t(as.matrix(rdata[j,1:4]-rdata[i,1:4])))
  }
}

M_pq<-as.data.frame.table(M_pq)
M_pq[,1:2]<-lapply(M_pq[,1:2],as.numeric)

M_pq |> ggplot(aes(x=Var1,y=Freq)) +
  geom_point(aes(fill=Freq,colour=Freq),pch=22) +
  scale_fill_continuous(high = "#0033FF", low = "#FFFFFF") +
  scale_colour_continuous(high = "#0033FF", low = "#FFFFFF") +
  scale_x_continuous(name="Observations") +
  scale_y_continuous(name="Distance") +
  theme_bw() + theme(legend.position = "none")
```



Note the differing patterns for observations 101 and 102, as well as the diffuse cloud of points above the distance 5.0 for the other observations. There are a few other observations for which the distances to other observations seem to be larger than in a majority of the cases.

Next, we display the same distributions with the help of boxplots.

```
median.value <- M_pq |>
  group_by(Var1) |>
  summarise(meanDist=mean(Freq)) |>
  summarise(median_value=median(meanDist))

test <- M_pq |>
  group_by(Var1) |>
  summarise(meanDist=mean(Freq)) |>
  summarise(std=sd(meanDist))
```

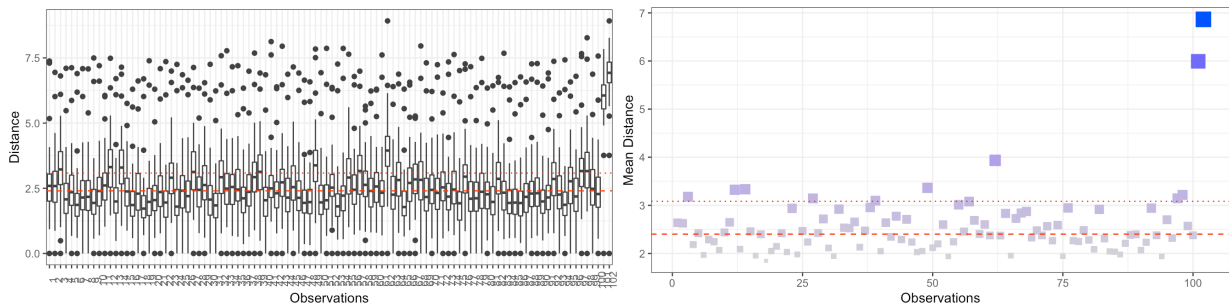
```
med.sd = test+median.value

M_pq |> ggplot(aes(x=as.factor(Var1),y=Freq)) +
  geom_boxplot() +
  scale_x_discrete(name="Observations") +
  scale_y_continuous(name="Distance") +
  theme_bw() + theme(legend.position = "none") +
  geom_hline(yintercept=as.numeric(median.value),
    linetype = "dashed", color = "red") +
  geom_hline(yintercept=as.numeric(med.sd),
    linetype = "dotted", color = "red") +
  theme(axis.text.x = element_text(angle=90))
```

The long-dashed red line (see below) represents the median of all the mean distances per observation; the short-dashed red line lies 1 standard deviation above the median.

To simplify the reading of the situation, we plot only the mean distance per observation, linking the **colour intensity** and the **marker size** to the mean distance (blue corresponding to larger distances, as do larger markers).

```
M_pq |> group_by(Var1) |> summarise(meanDist=mean(Freq)) |>
  ggplot(aes(x=Var1,y=meanDist)) +
  scale_x_continuous(name="Observations") +
  scale_y_continuous(name="Mean Distance") +
  geom_point(aes(fill=meanDist,colour=meanDist,
    size=meanDist),pch=22) +
  scale_fill_continuous(high = "#0033FF",
    low = "#CCCCCC") +
  scale_colour_continuous(high = "#0033FF",
    low = "#CCCCCC") +
  theme_bw() + theme(legend.position = "none") +
  geom_hline(yintercept=as.numeric(median.value),
    linetype = "dashed", color = "red") +
  geom_hline(yintercept=as.numeric(med.sd),
    linetype = "dotted", color = "red")
```



Do any other observations strike you as potential outliers?

Similarity Measures (Reprise) 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** (also called the **Chebychev distance**)

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

Note that the Minkowski distance d_p is only a distance function (i.e., a **metric**) when $p \geq 1$.²²

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 $1 - J(P, Q)$.²³

Finally, let $\mathbf{p}, \mathbf{q} \neq \mathbf{0}$. Recall that $\mathbf{p} \cdot \mathbf{q} = \|\mathbf{p}\| \|\mathbf{q}\| \cos \theta$, 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

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

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-based methods for anomaly detection; they will also eventually be useful for density-based anomaly detection.

22: 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.

23: 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.

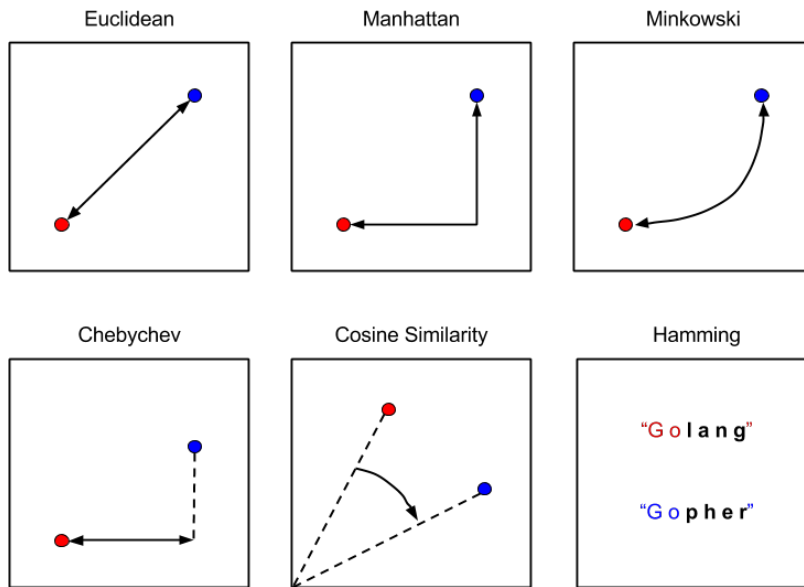


Figure 27.12: 2D visualization of various similarity metrics [363].

Distance-Based Anomaly Detection All these distance functions (similarity measures) 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 $k, v \leq |D|$, the **distance to all points** (DTAP) anomaly detection algorithm considers each observation \mathbf{p} in D and adds the distance from \mathbf{p} to every other observation in D , i.e.

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

The v observations 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.

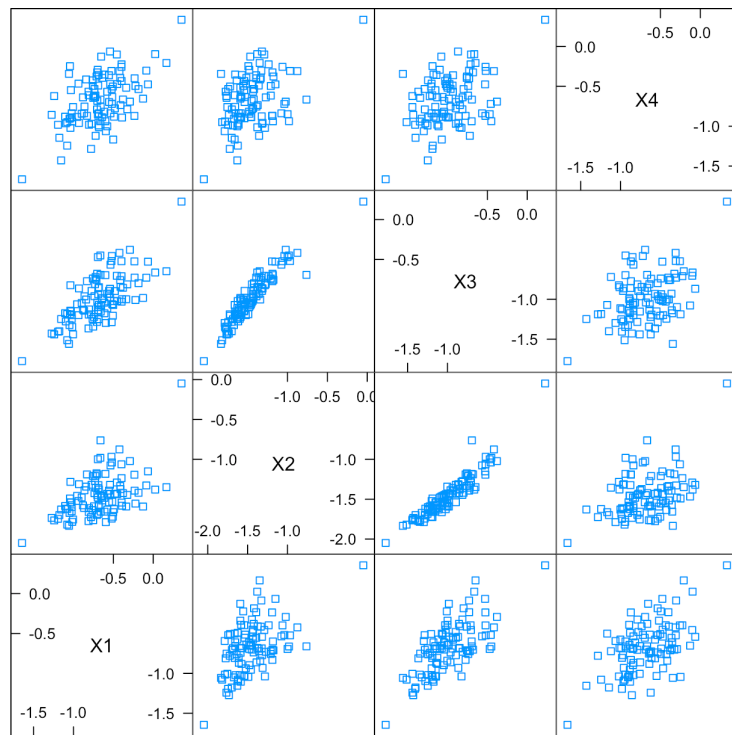
The **distance to nearest neighbour** (DTNN) algorithm defines

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

with a similar definition for the v anomalous observations. The **average distance to k nearest neighbours** and **median distance to k nearest neighbours** are defined similarly.

Example: Distance to All Points We start by building the DTAP anomaly detector for the Euclidean distance (`method="euclidean"`) on the **scaled** artificial data, which is shown below.

```
rdata.scaled=data.frame(matrix(ncol = 4, nrow = nobs+2))
for(i in 1:4){
  rdata.scaled[,i] <-
    2/(max(rdata[,i]) - min(rdata[,i])) * rdata[,i] - 1
}
lattice::splom(rdata.scaled[,1:4], pch=22)
```



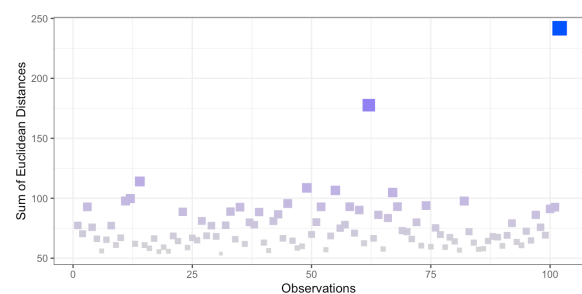
Scatter Plot Matrix

The top $\nu = 6$ anomalous observations are obtained as follows, with accompanying plot:

```
m.L2 <- as.matrix(dist(rdata.scaled[,1:4],
                      method="euclidean"))
adoa.L2 <- data.frame(1:(nobs+2), rowSums(m.L2))
colnames(adoa.L2) <- c("obs", "dist")
adoa.L2 <- adoa.L2[order(-adoa.L2$dist),]
rownames(adoa.L2) <- NULL
head(adoa.L2)

adoa.L2 |>
  ggplot(aes(x=obs,y=dist)) +
    scale_x_continuous(name="Observations") +
    scale_y_continuous(name="Sum of Euclidean Distances") +
    geom_point(aes(fill=dist, colour=dist, size=dist),
              pch=22) +
    scale_fill_continuous(high = "#0033FF", low = "#CCCCCC") +
    scale_colour_continuous(high = "#0033FF", low = "#CCCCCC") +
    theme_bw() + theme(legend.position = "none")
```

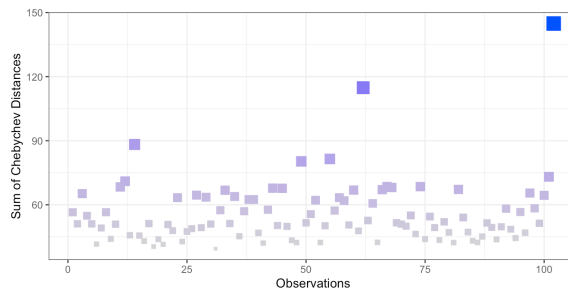
```
obs dist
102 241.7556
62 177.6135
14 113.9903
49 108.6464
55 106.5156
67 104.7870
```



We can repeat this process for a variety of metrics.²⁴

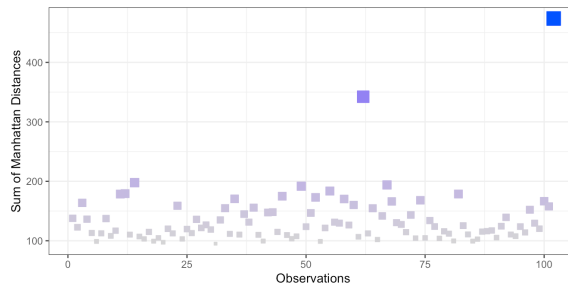
- **Chebychev** (replace `method="euclidean"` by `method="maximum"`)

```
obs dist
102 144.85387
62 114.83273
14 88.25016
55 81.47169
49 80.33274
101 73.11895
```



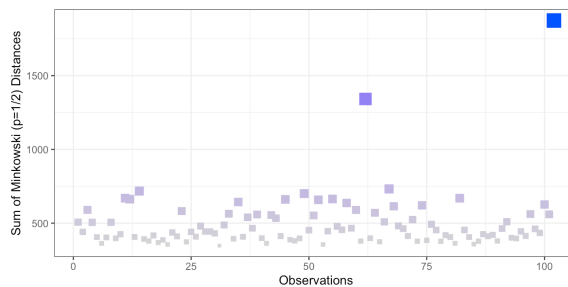
- **Manhattan** (`method="euclidean"` \mapsto `method="manhattan"`)

```
obs dist
102 473.7469
62 342.2531
14 197.7493
67 193.9257
49 191.7707
55 183.6509
```



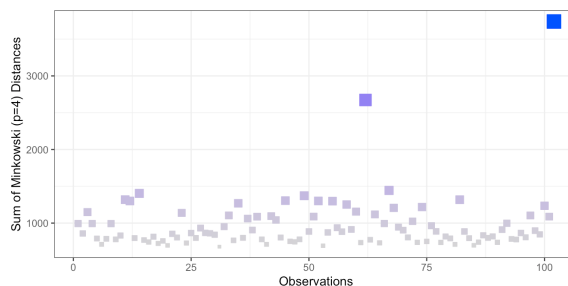
- **Minkowski**, $p = 1/2$ (`"euclidean"` \mapsto `"manhattan"`, $p=0.5$):

```
obs dist
102 1873.7488
62 1341.9020
67 731.9811
14 717.3451
49 700.5557
11 669.4313
```



- **Minkowski**, $p = 4$ (`"euclidean"` \mapsto `"manhattan"`, $p=4$):

```
obs dist
102 3738.781
62 2672.940
67 1444.177
14 1403.799
49 1372.067
11 1318.830
```



We see that while observation 102 is always the most anomalous according to DTAP, the ranking is affected by the choice of distance metric. Is this surprising?

Example: Distance to Nearest Neighbour We next build the DTNN anomaly detector for the Euclidean distance, again on the **scaled** artificial data.

As before, we display the top $v = 6$ anomalous observations and the accompanying charts for 5 different metrics.

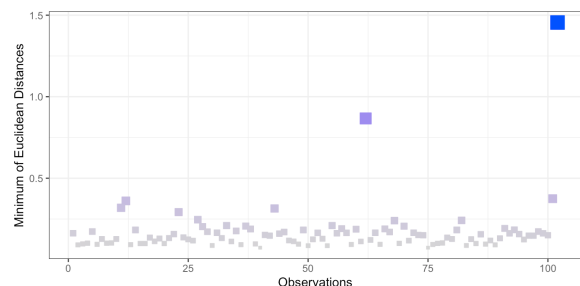
24: We do not display the code that is used for these other metrics; it can be obtained with simple modifications from the Euclidean code.

As above, we only present the code in the Euclidean case; the remaining metrics require only slight modifications. The factor 1000000 is used to create a matrix with a strongly dominant diagonal, to exclude observations being found nearest to themselves. Depending on the dataset, this factor could be reduced or may need to be increased.

```
m.L2 <- m.L2 + 1000000*diag(nobs+2)
adoa.L2 <- data.frame(1:(nobs+2),apply(m.L2,1,min))
colnames(adoa.L2) <- c("obs","dist")
adoa.L2 <- adoa.L2[order(-adoa.L2$dist),]
rownames(adoa.L2) <- NULL
head(adoa.L2)

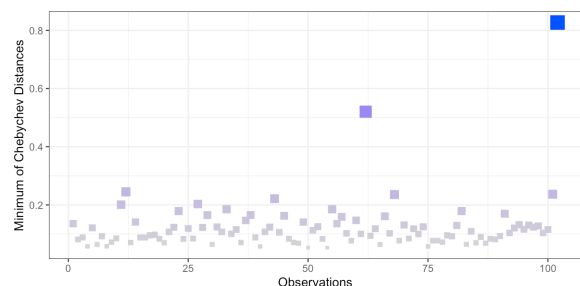
adoa.L2 |>
  ggplot(aes(x=obs,y=dist)) +
  scale_x_continuous(name="Observations") +
  scale_y_continuous(name="Minimum of Euclidean Distances") +
  geom_point(aes(fill=dist, colour=dist, size=dist),
             pch=22) +
  scale_fill_continuous(high = "#0033FF",
                       low = "#CCCCCC") +
  scale_colour_continuous(high = "#0033FF",
                         low = "#CCCCCC") +
  theme_bw() + theme(legend.position = "none")
```

```
obs dist
102 1.4549129
62 0.8667313
101 0.3746565
12 0.3611265
11 0.3186967
43 0.3145305
```



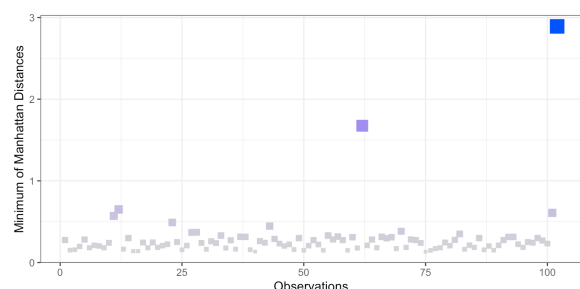
▪ **Chebychev** (replace method="euclidean" by method="maximum")

```
obs dist
102 0.8269178
62 0.5203870
12 0.2449443
101 0.2368307
68 0.2356369
43 0.2214142
```



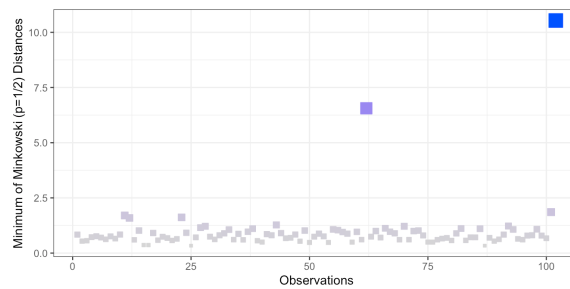
▪ **Manhattan** (method="euclidean" ↦ method="manhattan")

```
obs dist
102 2.8914743
62 1.6745725
12 0.6499116
101 0.6082916
11 0.5702036
23 0.4903152
```



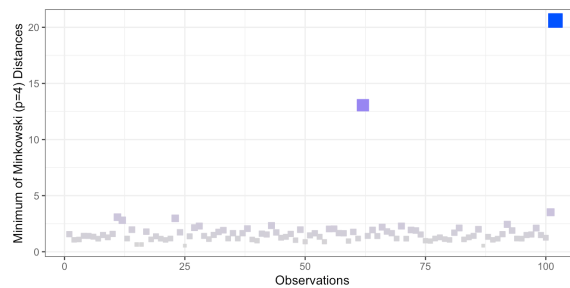
- Minkowski, $p = 1/2$ ("euclidean" \mapsto "manhattan", $p=0.5$):

```
obs dist
102 10.534441
62  6.559370
101 1.857222
11  1.704864
23  1.618388
12  1.591956
```



- Minkowski, $p = 4$ ("euclidean" \mapsto "manhattan", $p=4$):

```
obs dist
102 20.605432
62  13.059856
101 3.521821
11  3.081000
23  2.979090
12  2.809583
```



There are commonalities: certain observations come back repeatedly as likely anomalous observations.

Note, however, that the anomaly rankings change according to the **selected distance function** and the **choice of algorithm**; the choice of data scaling approach could also have an impact.

This is par for the course in the anomaly detection context.

27.2.2 Density Methods

Density-based approaches view observations as anomalous if they occur in **low-density regions**.

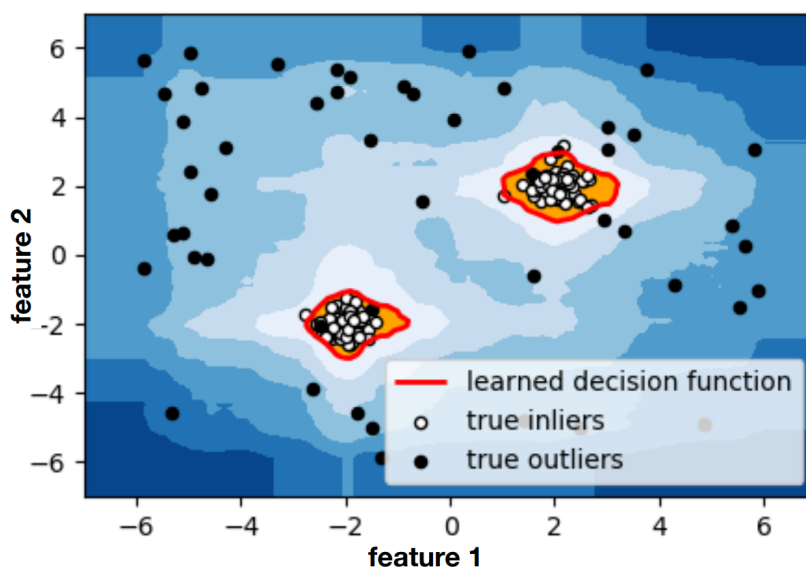


Figure 27.13: Low-density areas as outlier nurseries [338].

Density-based methods include:

- **local outlier factors;**
- **DBSCAN,** and
- **isolation forests.**

Local Outlier Factor The **Local Outlier Factor** (LOF) algorithm was proposed in 2000 by [355] (a summary can be found in Section 6.4.2 of [339]). LOF works by measuring the local deviation of each observation in a dataset from its k nearest neighbours, with a point said to be anomalous if this **deviation is large**.

A **local k -region** $N_k(\mathbf{p})$ around an observation \mathbf{p} is defined as the k nearest neighbours of \mathbf{p} . The density of observations in each of their respective local k -neighbourhoods is estimated (the **local density**), and compared to the density of the local k -neighbourhoods of the observations 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 27.14.

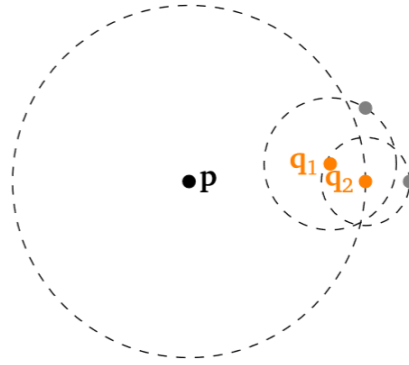


Figure 27.14: k -local density of \mathbf{p} compared to $\mathbf{q}_1, \mathbf{q}_2$.

The formal procedure is implemented in the algorithm of Figure 27.15.

Any observation with a LOF $a_k(\mathbf{p})$ above some threshold τ is a **local outlier**, but selecting is not obvious. LOF introduces the idea of a **reachability distance**, which improves the stability of results within clusters/regions: within $N_k(\mathbf{p})$, it is

$$d_{\text{reach}}(\mathbf{p}, \mathbf{q}) = \max_{\ell} \{d(\mathbf{p}, \mathbf{q}_{\ell}); \mathbf{q}_{\ell} \in N_k(\mathbf{p})\},$$

the maximal distance to its k -neighbours; outside of $N_k(\mathbf{p})$, it is

$$d_{\text{reach}}(\mathbf{p}, \mathbf{q}) = d(\mathbf{p}, \mathbf{q}),$$

the actual distance.

In Figure 27.16, assuming $k = 3$, the observations $\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3$ 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 observation \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} .

Algorithm 1: Local Outlier Factor (LOF)

-
- 1 **Input:** dataset D , point $\mathbf{p} \in D$, integer k for number of nearest neighbours to consider, distance function d
 - 2 Compute the distance between all points in D
 - 3 **for** $\mathbf{p} \in D$ **do**
 - 4 **for** $\mathbf{q} \in D \setminus \{\mathbf{p}\}$ **do**
 - 5 | Compute $d(\mathbf{p}, \mathbf{q})$
 - 6 **end**
 - 7 Order D by increasing distance from \mathbf{p}
 - 8 Set $d_k(\mathbf{p}) = d(\mathbf{p}, \mathbf{q}_k)$
 - 9 **end**
 - 10 Find the k nearest neighbours of \mathbf{p}
 - 11 Set $N_k(\mathbf{p}) = \{\mathbf{q} \in D \setminus \{\mathbf{p}\} : d(\mathbf{p}, \mathbf{q}) \leq d_k(\mathbf{p})\}$
 - 12 Define the reachability distance

$$d_{\text{reach}}(\mathbf{p}, \mathbf{q}) = \max\{d_k(\mathbf{q}), d(\mathbf{p}, \mathbf{q})\}$$
 - 13 Define the average reachability distance

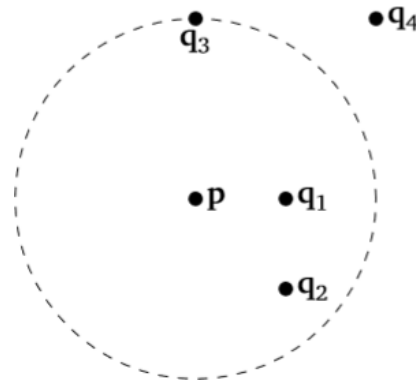
$$\overline{d_{\text{reach}}}(\mathbf{p}) = \frac{\sum_{\mathbf{q} \in N_k(\mathbf{p})} d_{\text{reach}}(\mathbf{p}, \mathbf{q})}{|N_k(\mathbf{p})|}$$
 - 14 Define the local reachability density

$$\ell_k(\mathbf{p}) = \left(\overline{d_{\text{reach}}}(\mathbf{p})\right)^{-1}$$
 - 15 Compute the local outlier factor $a_k(\mathbf{p}) = \frac{\sum_{\mathbf{q} \in N_k(\mathbf{p})} \frac{\ell_k(\mathbf{q})}{\ell_k(\mathbf{p})}}{|N_k(\mathbf{p})|}$
 - 16 **Output:** LOF $a_k(\mathbf{p})$
-

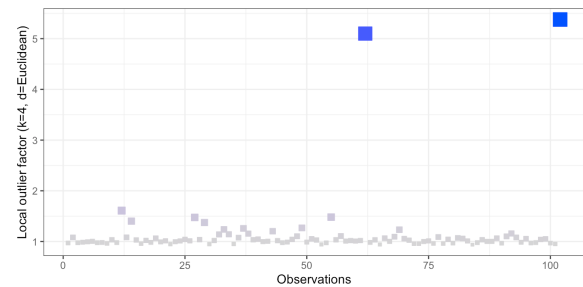
Example LOF is implemented in R via the Rlof package; we apply it to the scaled data using the Euclidean and the Chebychev distances.

```
dist.L2 = dist(rdata.scaled[,1:4], method="euclidean")
lof <- Rlof::lof(dist.L2, k=4)
rdata.lof.L2 = data.frame(rdata.scaled[,1:4], lof)
rdata.lof.obs.L2 = data.frame(1:(nobs+2), lof)
names(rdata.lof.obs.L2) = c("obs", "lof")
rdata.lof.obs.L2 <-
  rdata.lof.obs.L2[order(-rdata.lof.obs.L2$lof),]
rownames(rdata.lof.obs.L2) <- NULL
head(rdata.lof.obs.L2)

rdata.lof.obs.L2 |>
  ggplot(aes(x=obs, y=lof)) +
  scale_x_continuous(name="Observations") +
  scale_y_continuous(name="Local outlier factor"
    (k=4, d=Euclidean)) +
  geom_point(aes(fill=lof, colour=lof, size=lof), pch=22) +
  scale_fill_continuous(high = "#0033FF", low = "#CCCCCC") +
  scale_colour_continuous(high = "#0033FF", low = "#CCCCCC") +
  theme_bw() + theme(legend.position = "none")
```

Figure 27.16:
with $k = 3$.

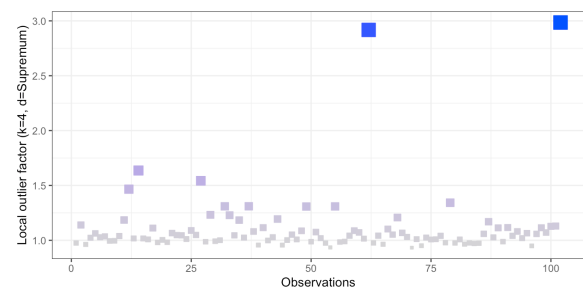
```
obs lof
102 5.377542
62 5.100218
12 1.609463
55 1.481152
27 1.475414
14 1.402112
```



```
dist.sup = dist(rdata.scaled[,1:4], method="maximum")
lof <- Rlof::lof(dist.sup, k=4)
rdata.lof.sup = data.frame(rdata.scaled[,1:4], lof)
rdata.lof.obs.sup = data.frame(1:(nobs+2), lof)
names(rdata.lof.obs.sup) = c("obs", "lof")
rdata.lof.obs.sup <-
  rdata.lof.obs.sup[order(-rdata.lof.obs.sup$lof),]
rownames(rdata.lof.obs.sup) <- NULL
head(rdata.lof.obs.sup)

rdata.lof.obs.sup |>
  ggplot(aes(x=obs, y=lof)) +
    scale_x_continuous(name="Observations") +
    scale_y_continuous(name="Local outlier factor
                        (k=4, d=Supremum)") +
    geom_point(aes(fill=lof, colour=lof, size=lof), pch=22) +
    scale_fill_continuous(high = "#0033FF", low = "#CCCCCC") +
    scale_colour_continuous(high = "#0033FF",
                           low = "#CCCCCC") +
    theme_bw() + theme(legend.position = "none")
```

```
obs lof
102 2.984990
62 2.917587
14 1.636228
27 1.542470
12 1.466011
79 1.342456
```

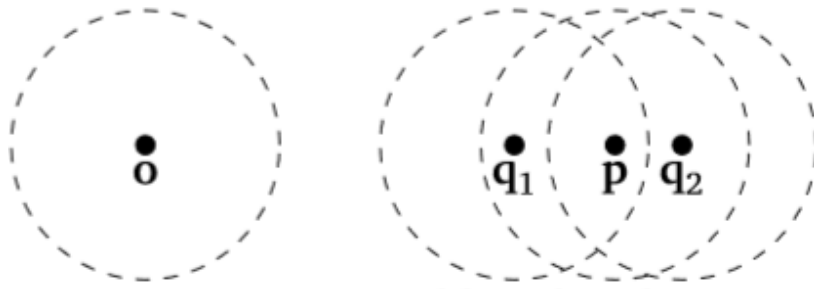


DBSCAN Density-Based Spatial Clustering of Applications with Noise (DBSCAN) was proposed in 1996 by [353] (a summary can be found in Section 4.1.5 of [339], as well as in Section 22.4.1). As its name suggests, it is a density-based clustering algorithm that groups nearby observations together and labels observations that do not fall in the clusters as **anomalies/outliers**.

Hierarchical DBSCAN (HDBSCAN) [354] 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 [364].

In DBSCAN,

- an observation **p** is a **core point** if there is a minimum of m observations within distance r of **p**;
- an observation **q** is a **border point** (or non-core point) if it is not itself a core point but is within distance r of one, and
- an observation **o** is an **outlier** if it is neither a core point nor a border point.



DBSCAN considers each observation in the dataset individually. If that observation 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 observation in this r -neighbourhood is then considered in turn, with the r -neighbourhoods of other core observations contained in the neighbourhood being **added to the cluster**.

This expansion repeats until all observations have been examined. During this step, observations that were previously labelled as outliers may be updated as they become border points in the new cluster. This process continues until every observation has either been assigned to a cluster or labelled as an outlier.

The formal procedure is implemented in the algorithm of Figure 27.18.

While DBSCAN’s dual use as a **clustering algorithm** may seem irrelevant in the outlier detection setting, it is its ability to successfully identify clusters that is crucial for labeling the remaining observations as outliers.

DBSCAN/HDBSCAN Strengths:

- the number of clusters does not need to be known beforehand (unlike in k -means and other clustering algorithms);
- clusters of arbitrary shape can be detected;
- when using HDBSCAN, only the parameter for the **minimum cluster size** m is required, which can be set fairly intuitively.²⁵

25: This is not the case for the parameters in general clustering algorithms: if the elements of D are n -dimensional, the only restriction is that $m \geq n + 1$ (larger values of m allow for better noise identification).

Algorithm 2: DBSCAN

```

1 Input: dataset  $D$ , distance function  $d$ ,
   neighbourhood radius  $r > 0$ , minimum number of
   points to be considered a cluster  $m \in \mathbb{N}$ 
2  $Clusters = \{\}$ 
3  $Outliers = \{\}$ 
4 for  $p \in D$  do
5   if  $p \in Outliers \cup (\cup_{C \in Clusters} C)$  then
6     continue
7   end
8   Set  $N(p) = \{q \in D : d(p, q) \leq r\}$ 
9   if  $|N(p)| < m$  then
10    Add  $p$  to  $Outliers$ 
11    continue
12  end
13  else
14     $Cluster = N(p)$ 
15    for  $q \in Cluster \setminus \{p\}$  do
16      if  $q \in Outliers$  then
17        Remove  $q$  from  $Outliers$ 
18      end
19      else if  $q \in \cup_{C \in Clusters} C$  then
20        continue
21      end
22      Set  $N(q) = \{q' \in D : d(q, q') \leq r\}$ 
23      if  $|N(q)| \geq m$  then
24         $Cluster = Cluster \cup N(q)$ 
25      end
26    end
27  end
28  Add  $Cluster$  to  $Clusters$ 
29 end
30 return  $Outliers$ 
31 Output: a list of outliers

```

Figure 27.18:

DBSCAN/HDBSCAN Limitations:

- it is not deterministic, as border points can be assigned to different clusters depending on the order in which core observations are considered – this does not affect its use as an anomaly detection algorithm, however;
- in high-dimensional spaces, the ability of any Euclidean-based distance function to distinguish near and distant observations diminishes due to the **Curse of Dimensionality**; in such spaces, it becomes ineffective (as do other clustering algorithms);

- it 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).



Example We use the R implementation of DBSCAN, HDBSCAN, and OPTICS (another density-based clustering algorithm) found in the `dbscan` package; we apply various parameters to the scaled artificial data, using the Euclidean distance in all instances.²⁶

```
scaled = scale(rdata[,1:4])
set.seed(1) # for replicability
```

- DBSCAN, `eps = 0.4`, `minPts = 4`

```
(db.1 <- dbscan::dbscan(scaled, eps = .4, minPts = 4))
lattice::splom(scaled, groups=db.1$cluster + 1L, pch=22)
```

The clustering contains 1 cluster(s) and 96 noise points.

```
0 1
96 6
```

Evidently, 0.4 is too small a value for `eps` or 4 is too large a value for `minPts` (or both).

- DBSCAN, `eps = 1`, `minPts = 4`

```
(db.2 <- dbscan::dbscan(scaled, eps = 1, minPts = 4))
lattice::splom(scaled, groups=db.2$cluster + 1L, pch=22)
```

The clustering contains 1 cluster(s) and 6 noise points.

```
0 1
6 96
```

The results are reversed with a larger value of `eps`.

26: We display all the accompanying charts in Figures 27.19 and 27.20.

- DSBCAN, $\text{eps} = 1, \text{minPts} = 10$

```
(db.3 <- dbscan::dbscan(scaled, eps = 1, minPts = 10))
lattice::splom(scaled, groups=db.3$cluster + 1L, pch=22)
```

The clustering contains 1 cluster(s) and 24 noise points.

```
0 1
24 78
```

Are the clustering results (i.e., the anomaly discovery rate) as expected?
The interaction between the parameters can have unpredictable effects.

- DSBCAN, $\text{eps} = 2, \text{minPts} = 10$

```
(db.4 <- dbscan::dbscan(scaled, eps = 2, minPts = 10))
lattice::splom(scaled, groups=db.4$cluster + 1L, pch=22)
```

The clustering contains 1 cluster(s) and 2 noise points.

```
0 1
2 100
```

- HDBSCAN, $\text{minPts} = 4$

```
(hdb <- dbscan::hdbscan(scaled, minPts = 4))
lattice::splom(scaled, groups=hdb$cluster + 1L, pch=22)
```

The clustering contains 4 cluster(s) and 71 noise points.

```
0 1 2 3 4
71 10 6 4 11
```

Note the absence of the eps parameter.

- OPTICS, $\text{eps} = 1, \text{minPts} = 4, \text{eps_cl} = 1, \text{xi} = .05^{27}$

```
opt <- dbscan::optics(scaled, eps = 1, minPts = 4)
(opt.1 <- dbscan::extractDBSCAN(opt, eps_cl = 1))
lattice::splom(rdata[,1:4], groups=opt.1$cluster + 1L, pch=22)
(opt.2 <- dbscan::extractXi(opt, xi = .05))
lattice::splom(scaled, groups=opt.2$cluster + 1L, pch=22)
```

The clustering contains 1 cluster(s) and 6 noise points.

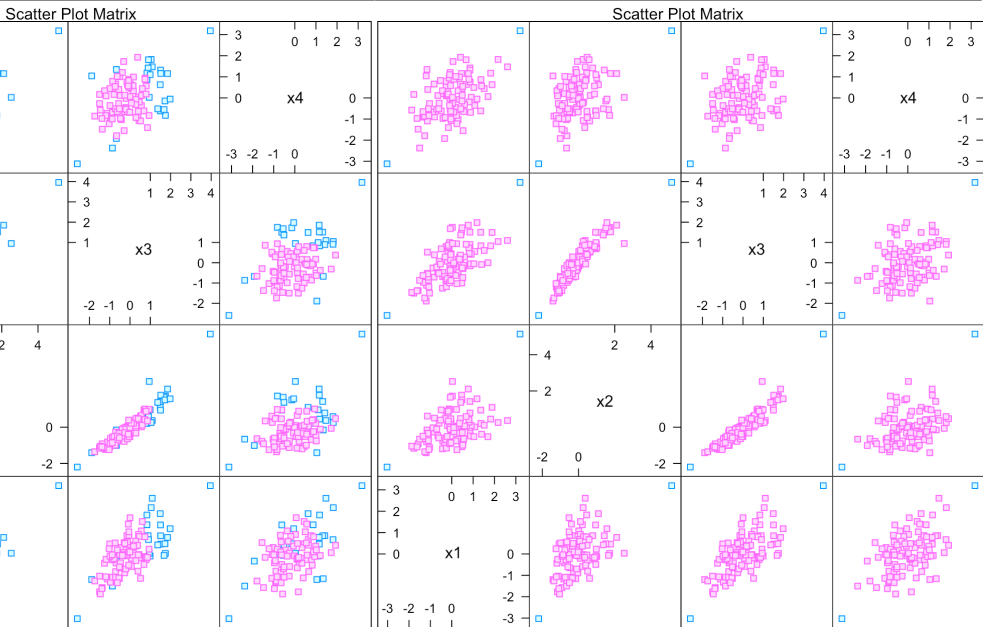
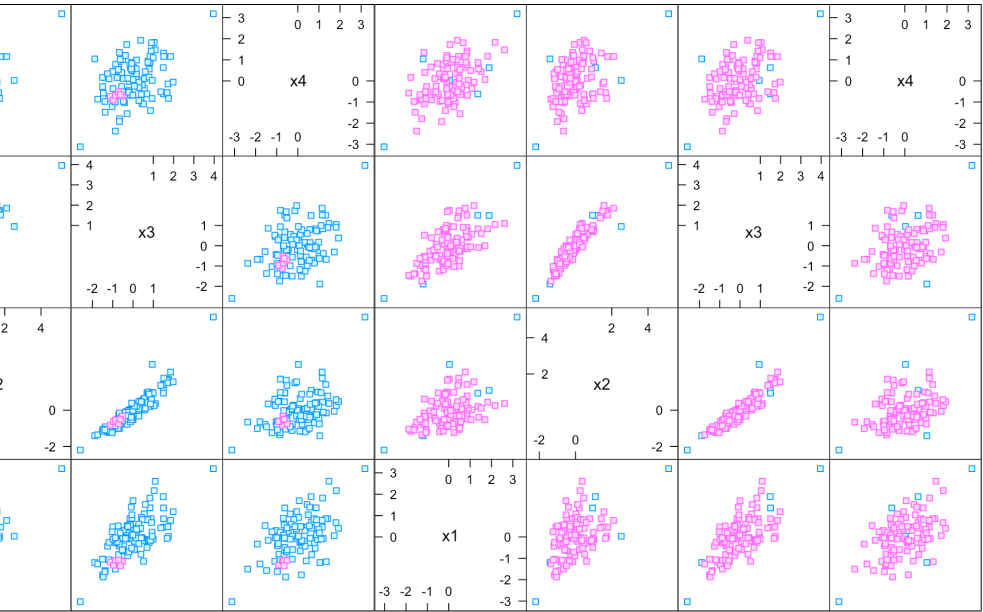
```
0 1
6 96
```

The clustering contains 4 cluster(s) and 7 noise points.

```
0 1 2 3 4
7 4 4 11 76
```

Are there any surprises?

27: Read the `dbscan` package documentation for details.



Outcomes (outliers in blue): DBSCAN, top left ($\text{eps} = 0.4, \text{minPts} = 4$); DBSCAN, top right ($\text{eps} = 1, \text{minPts} = 10$); DBSCAN, bottom right ($\text{eps} = 2, \text{minPts} = 10$).

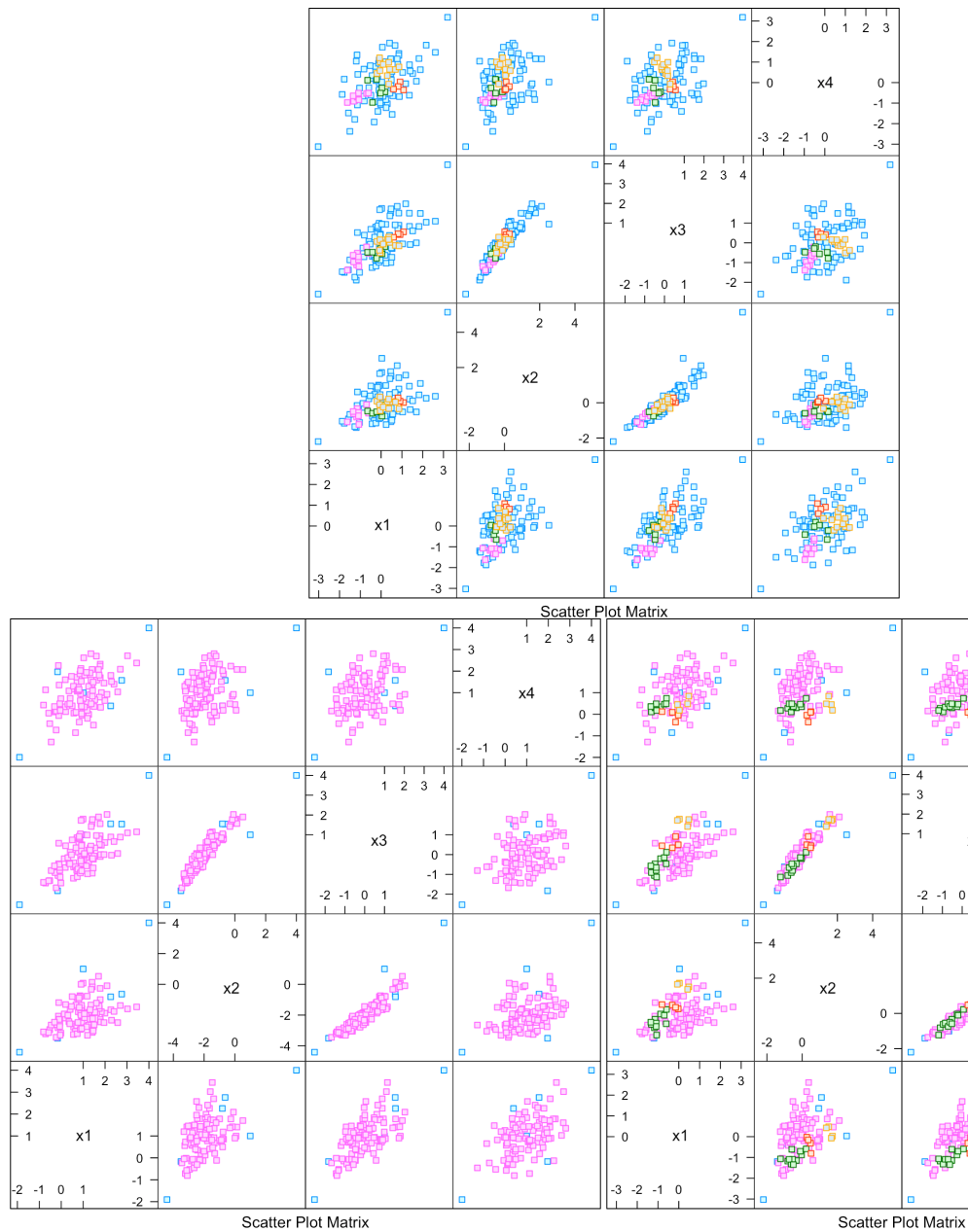


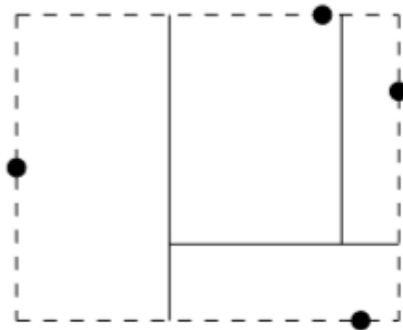
Figure 27.20: Clustering outcomes (outliers in blue): HDBSCAN, top (minPts = 4); OPTICS, bottom left (eps = 1, xi = 0.05); OPTICS, bottom right (eps = 0.4, minPts = 4, eps_cl = 1, xi = 0.05).

Isolation Forest The approaches that were previously discussed first construct models of what normal observations look like, and then identify observations that do not fit this model.

The **Isolation Forest** algorithm [351] introduced in 2008 instead tries to explicitly identify outliers under the assumptions that **there are few outliers** in the data and that these outliers have **very different attributes** compared to normal (or regular) observations.

This allows the use of sampling techniques that increase algorithmic speed while decreasing memory requirements.

The algorithm attempts to **isolate** anomalous observations by **randomly** selecting an attribute and then **randomly** selecting a split between that attribute's min and max values. This recursively partitions the observations until every observation is isolated in its own partition component.



Recursive partitioning yields a binary tree called an **Isolation Tree** (IsoTree):

- the **root** of this tree is the entire dataset;
- each **node** is a subset of the observations;
- each **branch** corresponds to one of the generated partitions, and
- the **leaves** are sets containing a single isolated observation.

Each observation is then assigned a score derived from **how deep in the tree** its singleton partition appears. Observations that are shallower in the tree are easier to separate from the rest – these are likely **outliers**.

Since only shallow observations are of interest, once the height of the tree has reached a **given threshold**,²⁸ further construction of the tree can be stopped to decrease computational cost.

28: The expected height of a random binary tree, say.

Instead of building a tree from the entire dataset, a tree can be constructed from a subset. The location of any observation within this smaller tree can then be estimated, again saving computational and memory resources. These two improvements are detailed in the original paper [351].

The formal procedure is implemented in the algorithm of Figure 27.21.

Once a number of Isolation Trees have been randomly generated (an **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 an observation's path length in each

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

```

1 Input: dataset  $D$ 
2 if  $|D| \leq 1$  then
3   | return  $\{\}$ 
4 end
5 else
6   | Let  $\bar{A}$  be a list of attributes in  $D$ 
7   | Randomly select an attribute  $A \in \bar{A}$ 
8   | Randomly sample a point  $s$  from
      |  $[\min_{q \in D} A(q), \max_{q \in D} A(q)]$ 
9   | Return
      |  $\begin{cases} \text{LeftChild} & = iTree(\{q \in D : A(q) \leq s\}) \\ \text{RightChild} & = iTree(\{q \in D : A(q) > s\}) \\ \text{NodeValue} & = D \end{cases}$ 
10 end
11 Output: Binary tree with node values that are
      subsets of  $D$ 

```

Figure 27.21: Isolation Tree algorithm.

tree has been computed, the average path length is taken to be its score. In isolated forests, low scores are indicative of outlying behaviour.

The formal procedure is implemented in the algorithm of Figure 27.23.

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

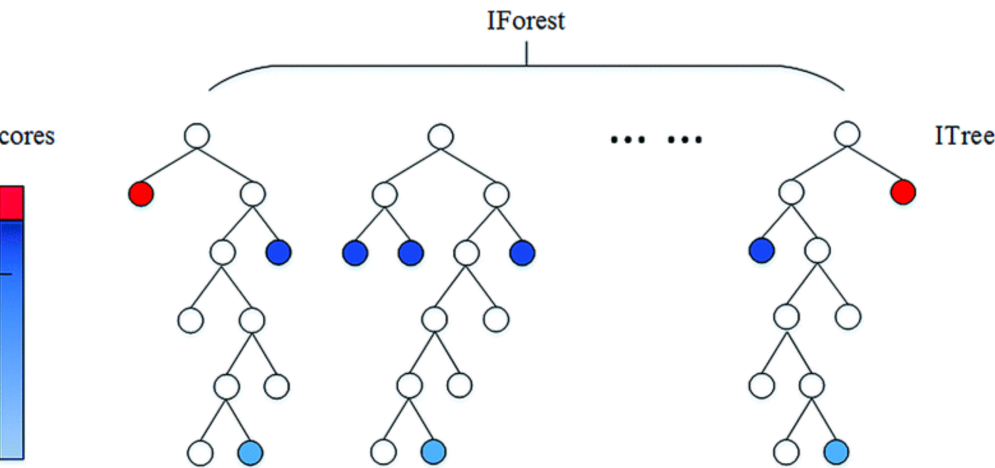
$$\log_2 a(\mathbf{p}) = -\frac{\text{average path length to } \mathbf{p} \text{ in the Isolation Trees}}{c(n)}.$$

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 observation; if all observations receive a score ≈ 0.5 , this suggests that there are no anomalies present.

IsoForest Strengths:

- small time and memory requirements;
- can handle high dimensional data, and
- do not need observations to have been labeled anomalies in the training set.

IsoForest Main Limitation:



st schematics [338].

- the anomaly score assigned to a given point can have high variance over multiple runs of the algorithm. The authors of [352] propose some solutions.

Example We use the R implementation of IsoForest found in the `solitude` package; we apply various parameters to the scaled artificial data, using the Euclidean distance in all instances. Note that this implementation uses a different scoring system, in which high scores are indicative of anomalous observations.

```
#library(solitude)
set.seed(1) # for replicability
index = 1:102
```

We initiate an isolation forest:

```
iso = solitude::isolationForest$new(
  sample_size = length(index))
iso$fit(dataset = rbind(scaled[index,1:4],c(0,0,0,0)))
test<-iso$predict(scaled[index,1:4]) # scores for Tr data
```

The top $v = 6$ IsoForest anomaly scores are given below:

```
rdata.iso = data.frame(1:(nobs+2),test$anomaly_score)
names(rdata.iso) = c("obs","anomaly_score")
rdata.iso <- rdata.iso[order(-rdata.iso$anomaly_score),]
rownames(rdata.iso) = NULL
head(rdata.iso)

rdata.iso |>
  ggplot(aes(x=obs,y=anomaly_score)) +
  scale_x_continuous(name="Observations") +
  scale_y_continuous(name="IsoForest Anomaly Score") +
  geom_point(aes(fill=anomaly_score, colour=anomaly_score,
```

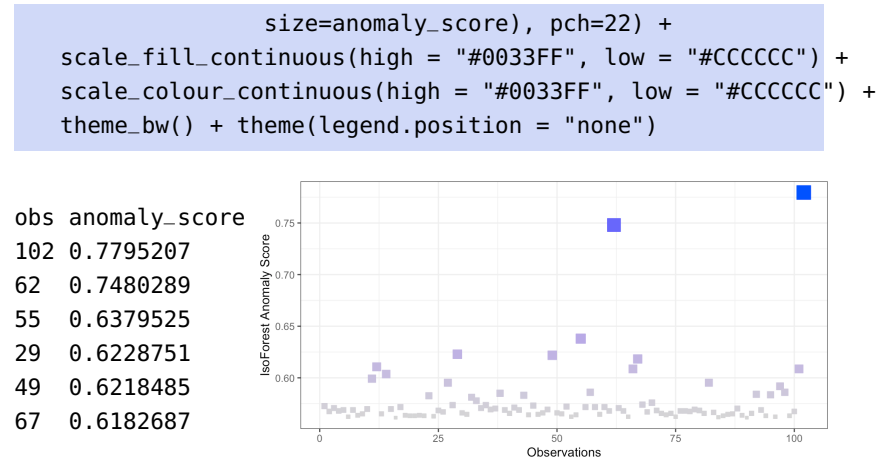
Algorithm 4: Isolation Forest

```

1 Input: dataset  $D$ , integer  $t$  number of Isolation
  Trees
2  $Forest = \{\}$ 
3 for  $i = 1$  to  $t$  do
4    $Tree = iTree(D)$ 
5   Add  $Tree$  to  $Forest$ 
6 end
7 for  $p \in D$  do
8    $PathLengths = \{\}$ 
9   for  $Tree$  in  $Forest$  do
10    Find the path length  $\ell$  from the root of  $Tree$ 
      to node  $\{p\}$ 
11    Add  $\ell$  to  $PathLengths$ 
12  end
13   $AveragePathLength = \frac{\sum_{\ell \in PathLengths} \ell}{t}$ 
14  Set  $a(p) = 2^{-\frac{AveragePathLength}{c(|D|)}}$ 
15 end
16 Output: Anomaly score  $a(p) \in [0, 1]$  for each
     $p \in D$ 

```

Figure 27.23:



The profile of anomaly scores has a fairly distinct look (although we recognize quite a few of the usual suspects).

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** [365].

Hair Colour			Mother Tongue	Hair Colour			
Black	Blond	Red		Black	Brown	Blond	Red
4	12	2	French	34	70	32	5
4	3	6	English	40	111	13	18
2	1	0	Mandarin	46	6	2	0
1	0	0	Arabic	30	4	0	2
7	13	1	Other	37	26	32	9

Mother Tongue				
French	English	Mandarin	Arabic	Other
141	182	54	36	104
27.3%	35.2%	10.4%	7.0%	20.1%

Age			
24-	25-44	45-64	65+
175	244	66	32
33.8%	47.2%	12.8%	6.2%

Mother Tongue	Age			
	24-	25-44	45-64	65+
French	49	66	22	4
English	65	83	26	8
Mandarin	19	27	4	4
Arabic	10	19	4	3
Other	32	49	10	13

Total Number of Observations:	517	
Percentage of Levels Above:	15%	25%
Hair Colour	75%	50%
Mother Tongue	60%	60%
Age	50%	50%

... and 1–way tables for the artificial example; the percentages of the levels above certain thresholds provide a summary of each of the categorical variables.

27.3 Qualitative Approaches

Non-numerical variables present new challenges when it comes to anomaly detection. A **categorical variable**²⁹ is one whose levels are measured on a nominal scale; examples include an object's colour, an individual's mother tongue, her favourite meal, and so on.

The **central tendency** of a categorical variable is usually given by its **mode**; measures of spread are harder to define consistently.³⁰

Consider a dataset with $n = 517$ observations and $p = 3$ predictors:

- **age** (24–, 24–44, 45–64, 65+);
- **mother tongue** (French, English, Mandarin, Arabic, Other), and
- **hair colour** (black, brown, blond, red).

The respective modes are 24–44, English, and brown. We can see the distribution tables in Figure 27.24, as well as the number of levels that contain more than 15% and more than 25% of the observations.

We often associate qualitative features to numerical values, but with the caveat that these **should not be interpreted as numerals**.³¹

A categorical variable that has exactly two is called a **dichotomous feature** (or a binary variable); those with more than two levels are called **polytomous variables**.³²

Commonly-used distances (apart from the 0–1 distance and the related **Hamming distance**) typically require numerical inputs. Anomaly detection methods based on distance or on density are **not recommended** in the qualitative context.³³ Another option is to look at combinations of feature levels, but this can prove computationally expensive.

We present two specific **categorical anomaly detection** methods below: the attribute value frequency algorithm, and the greedy algorithm.

29: Or qualitative variable.

30: One possibility is to use the proportion of levels with more than a certain percentage of the observations above a given threshold.

31: If we use the code “red” = 1, “blond” = 2, “brown” = 3, and “black” = 4 to represent hair colour, we **cannot conclude** that “blond” > “red”, even though $2 > 1$, or that “black” – “brown” = “red”, even though $4 - 3 = 1$.

32: While we are on the topic, regression on categorical variables is called **multinomial logistic regression**.

33: Unless the distance function has been modified appropriately, but that is harder to do than one may expect.

27.3.1 Attribute Value Frequency Algorithm

The **Attribute Value Frequency** (AVF) algorithm offers a fast and simple way to detect outlying observations in categorical data; it can be conducted without having to create ir which minimizes the amount of data analyses, without having to create or search through various combinations of feature levels, which reduces the overall runtime.

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 anomalous** (or relatively infrequent). The **rarity** of an attribute level can be measured by adding the number of times the corresponding feature takes that value in the dataset.

Consider a p -dimensional dataset with n observations: $\{\mathbf{x}_i\}, i = 1, \dots, n$ (each observation is a vector of p features). We write

$$\mathbf{x}_i = (x_{i,1}, \dots, x_{i,\ell}, \dots, x_{i,p}),$$

where $x_{i,\ell}$ is \mathbf{x}_i 's ℓ th feature's level.

In the artificial categorical dataset presented previously above, we (perhaps) have

$$\begin{aligned} \mathbf{x}_1 &= (x_{1,1}, x_{2,1}, x_{3,1}) = (24-, \text{French}, \text{brown}) \\ &\vdots \\ \mathbf{x}_{517} &= (x_{517,1}, x_{517,2}, x_{517,3}) = (24-, \text{Mandarin}, \text{black}). \end{aligned}$$

Using the reasoning presented above, the **AVF score** is a good tool to determine whether \mathbf{x}_i should be considered an outlier or not:

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

where $f(x_{i,\ell})$ is the number of observations \mathbf{x} for which the ℓ th feature takes on the level $x_{i,\ell}$.

A **low** AVF score indicates that the observation is more likely to be an outlier. Since $\text{AVFscore}(\mathbf{x}_i)$ is essentially a sum of p positive values, it is minimized when each of the sum's term is minimized, individually.³⁴

Thus, the “**ideal**” anomalous observation (in the sense described above) minimizes the AVF score; the minimal score is reached when each of the observation's features' levels occur only once in the dataset.³⁵

For an integer ν , the suggested outliers are the ν observations signatures with smallest AVF score; the algorithm's complexity is $O(np)$.

The formal procedure is implemented in the algorithm of Figure 27.25.

The 10 lowest AVF scores in the artificial categorical dataset are highlighted on the next page.

34: What does this assume, if anything at all, about the features' independence.

35: Strictly speaking, the AVF score would be minimized when each of the observation's features' levels occur zero time in the dataset, but then ... the observation would not *actually* be in the dataset.

Algorithm 5: AVF

```

1 Inputs: dataset  $D$  ( $n$  observations,  $m$  features),
   number of anomalous observations  $k$ 
2 while  $i \leq n$  do
3    $j = 1$ 
4    $\text{AVFscore}(\mathbf{x}_i) = f(x_{i,j})$ 
5   while  $j \leq m$  do
6      $\text{AVFscore}(\mathbf{x}_i) = \text{AVFscore}(\mathbf{x}_i) + f(x_{i,j});$ 
7      $j = j + 1$ 
8   end
9    $\text{AVFscore}(\mathbf{x}_i) = \text{Mean}(\text{AVFscore}(\mathbf{x}_i))$ 
10   $i = i + 1$ 
11 end
12 Outputs:  $k$  observations with smallest AVF scores

```

Figure 27.25: AVF algorithm.

Age	Mother Tongue	Hair Colour			
		Black	Brown	Blond	Red
24-	French	167.7	177.7	131.7	116.7
	English	181.3	191.3	145.3	130.3
	Mandarin	138.7	148.7	102.7	87.7
	Arabic	132.7	142.7	96.7	81.7
	Other	155.3	165.3	119.3	104.3
25-44	French	190.7	200.7	154.7	139.7
	English	204.3	214.3	168.3	153.3
	Mandarin	161.7	171.7	125.7	110.7
	Arabic	155.7	165.7	119.7	104.7
	Other	178.3	188.3	142.3	127.3
45-64	French	131.3	141.3	95.3	80.3
	English	145.0	155.0	109.0	94.0
	Mandarin	102.3	112.3	66.3	51.3
	Arabic	96.3	106.3	60.3	45.3
	Other	119.0	129.0	83.0	68.0
65+	French	120.0	130.0	84.0	69.0
	English	133.7	143.7	97.7	82.7
	Mandarin	91.0	101.0	55.0	40.0
	Arabic	85.0	95.0	49.0	34.0
	Other	107.7	117.7	71.7	56.7

For instance,

$$\begin{aligned}
 \text{AVFscore}(24-, \text{French}, \text{blond}) &= \frac{1}{3}(f(24-) + f(\text{French}) + f(\text{blond})) \\
 &= \frac{1}{3}(175 + 141 + 79) = 131.7
 \end{aligned}$$

For anomaly detection purposes, individual raw AVF scores are not as meaningful as relative AVF scores.

27.3.2 Greedy Algorithm

The **greedy** algorithm “greedyAlg1” is an algorithm which identifies the set of candidate anomalous observations in an efficient manner.³⁶ The mathematical formulation of the problem is simple – given a dataset D and a number ν of anomalous observations to identify, we solve the optimization problem

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

36: Greedy in the sense that the algorithm picks the next step according to what is best there-and-now, and not with a long-term view.

where the **entropy** of the subset $D \setminus O$ is the sum of the entropy of each feature on $D \setminus O$ (see Section @ref(ML-CVE-dt)):

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

and

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

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

The “greedyAlg1” algorithm solves the 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 $H(D \setminus OS)$.
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

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

is added to OS.

5. Repeat steps 2-4 another $\nu - 1$ times to obtain a set of ν candidate anomalous observations.

More details can be found in the source article [366]; an interesting detail is that it is **scalable** – it will also work for big datasets, provided the right framework is used.

27.4 Anomalies in High-Dimensional Data

Nowadays, real datasets are often quite **large**; in some scenarios, the observations may contain **100s or 1000s of features** (or dimensions).

Many classical methods use **proximity** (distance) concepts for anomaly detection (see Section 27.2) and can only be expected to work reasonably well in cases where the sample size n is larger than the dimension p .

The management of **high-dimensional data** ($n < p$) offers specific difficulties: 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**: many conventional methods of detecting outliers are simply not efficient in the high-dimensional context, due to the **curse of dimensionality**. Consequently, **high-dimensional anomaly detection methods** are linked with dimension reduction and feature selection.

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 [335, 339, 350, 367, 368].

27.4.1 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 [335]; such an observation would, of course, be considered to be **irregular**.

The challenges of anomaly and outlier detection in **high-dimensional data** (HDD) are due to:

- the notion of distance becoming **irrelevant** due to the curse of dimensionality (whence “the problem of detecting outliers is like finding a needle in a haystack” [367]);
- **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 [346] 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 anomaly measurements to be **compared** (“apples-to-apples”);
4. consider the **local data behaviour** to determine whether an observation is abnormal or not.

27.4.2 Projection Methods

HDLSS (**high dimension, low sample size**) datasets can contain 100+ variables; the curse of dimensionality affects the efficiency of conventional anomaly/outlier detection methods.

One way to counter the problem is to **reduce the dataset’s dimensionality** while preserving its essential characteristics. We have discussed such **projection methods** in Chapter 23. Let us see how one of them, **principal components analysis**, can be applied to anomaly detection.³⁷

37: We take advantage of this reprise to present PCA using a different formalism.

PCA (Reprise) As we know, **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, etc.

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 section,
- as an **anomaly and outlier detection** approach.

Let the dataset be represented by a numerical, centered, and scaled $n \times p$ matrix $\mathbf{X} = [\mathbf{X}_1, \dots, \mathbf{X}_p]$ with n observations (number of rows) and p features (number of columns).

The **principal components** are linear combinations of the variables:

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

with $k \leq p$, yielding the largest variance subject to the constraint $\|\ell_i\| = 1$ (where $\|\cdot\|$ represents the Euclidean norm).

We can thus deduce that

$$\begin{aligned} \text{Var}(\mathbf{Y}_i) &= \text{Var}(\ell_i^T \mathbf{X}) = \ell_i^T \mathbf{X}^T \mathbf{X} \ell_i = \ell_i^T \mathbf{X}^T \mathbf{X} \ell_i \\ \text{Cov}(\mathbf{Y}_i, \mathbf{Y}_k) &= \text{Cov}(\ell_i^T \mathbf{X}, \ell_k^T \mathbf{X}) = \ell_i^T \mathbf{X}^T \mathbf{X} \ell_k. \end{aligned}$$

PCA finds the **loadings vector** ℓ_1 which maximizes the variance of \mathbf{Y}_1 :

$$\ell_1 = \arg \max_{\|\ell_1\|=1} \{ \ell_1^T \mathbf{X}^T \mathbf{X} \ell_1 \},$$

then the uncorrelated loadings vector ℓ_2 which maximizes the variance of \mathbf{Y}_2 :

$$\ell_2 = \arg \max_{\|\ell_2\|=1, \ell_1^T \ell_2=0} \{ \ell_2^T \mathbf{X}^T \mathbf{X} \ell_2 \}.$$

Similarly, the loadings vector ℓ_k is not correlated with any of the $\ell_i, i < k$, and maximizes the variance of \mathbf{Y}_k :

$$\ell_k = \arg \max_{\substack{\|\ell_k\|=1, \\ \ell_i^T \ell_k=0, \forall i < k}} \{ \ell_k^T \mathbf{X}^T \mathbf{X} \ell_k \}.$$

We solve this optimization problem for all $i < k$ through the Lagrangian

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

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, which translates to:

$$\begin{aligned} \mathbf{X}^T \mathbf{X} \ell_k &= \lambda_k \ell_k \\ \ell_k^T \ell_k &= 1 \quad \text{and} \quad \ell_k^T \ell_i = 0, \quad \text{for all } i < k. \end{aligned}$$

The loadings vector ℓ_k is thus the **eigenvector** of the covariance matrix $\mathbf{X}^T \mathbf{X}$ associated to the k th largest eigenvalue. The **proportion of the variance which can be explained** by the PCA can be calculated by first noting that

$$\sum_{i=1}^p \text{Var}(\mathbf{Y}_i) = \sum_{i=1}^p \ell_i^T \mathbf{X}^T \mathbf{X} \ell_i = \sum_{i=1}^p \lambda_i.$$

Consequently, the proportion of the total variance explained by the i th 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.³⁸

38: 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.³⁹

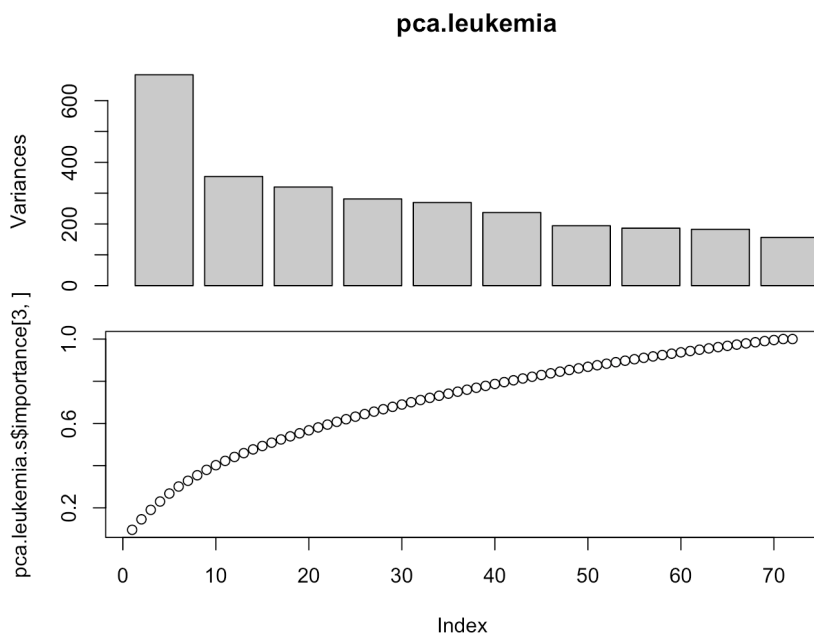
39: And thus, the corresponding principal components.

Example The `leukemia.big.csv` [↗](#) contains genetic expression measurements for $n = 72$ leukemia patients and $p = 7128$ genes [369].

```
leukemia.big <- read.csv("leukemia_big.csv")
# scale and format data
leukemia.big <- t(leukemia.big)
leukemia.big.scaled <- scale(leukemia.big)
```

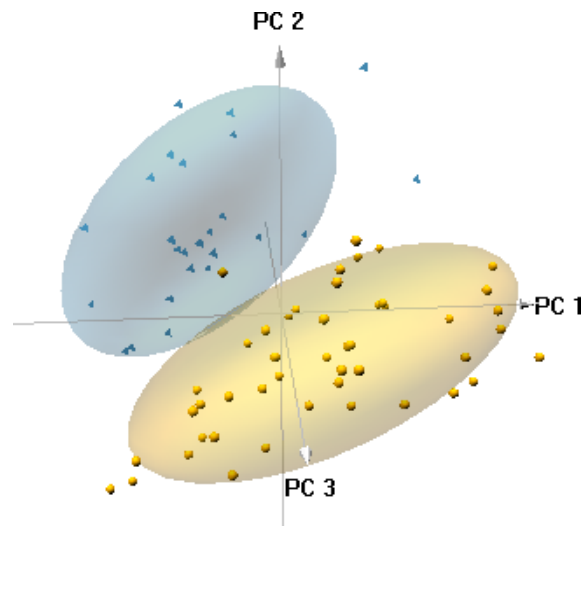
We find the PCA decomposition and display the plots.

```
pca.leukemia <- prcomp(leukemia.big.scaled)
plot(pca.leukemia)
pca.leukemia.s <- summary(pca.leukemia)
plot(pca.leukemia.s$importance[3,])
```



The scree plot suggests that only two principal components should be retained, but that does not explain an awful lot of the variation. Even keeping the first 3 principal components only explains roughly 20% of the variation.

The projection on the first 3 PC is shown below [author unknown]:



It is not obvious, however, that the methods presented here to find an optimal k are appropriate for **anomaly detection purposes**: simply put, a good k is one which **allows for good anomaly detection**.

There are other PCA-associated dimension reduction methods: independent components analysis, singular value decomposition, kernel PCA, etc. (see Chapter 23).

But 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 (hopefully) **mitigated** – traditional methods are then applied to the **projected data**.

Dimension reduction usually leads to a loss of information, however, 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.

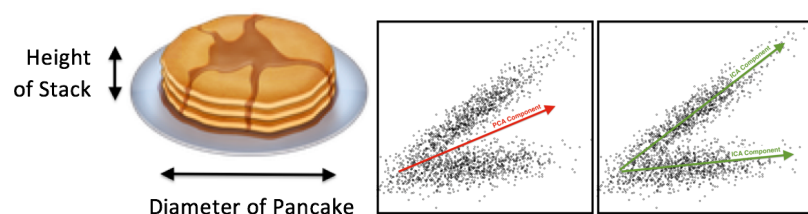
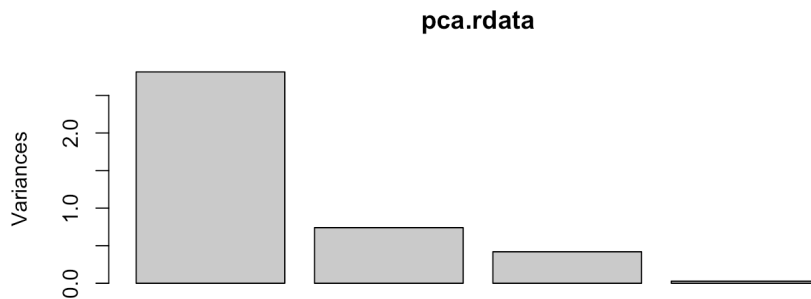


Figure 27.26: Principal component analysis (PCA) is a statistical task that finds the principal components of a dataset. PCA: pancake. Interesting property: the number of principal components (PCA components) is not aligned with the number of features. (Adapted from [21].)

Example Let us re-visit the artificial dataset from the earliest sections. We start by computing the PCA decomposition of the scaled dataset.

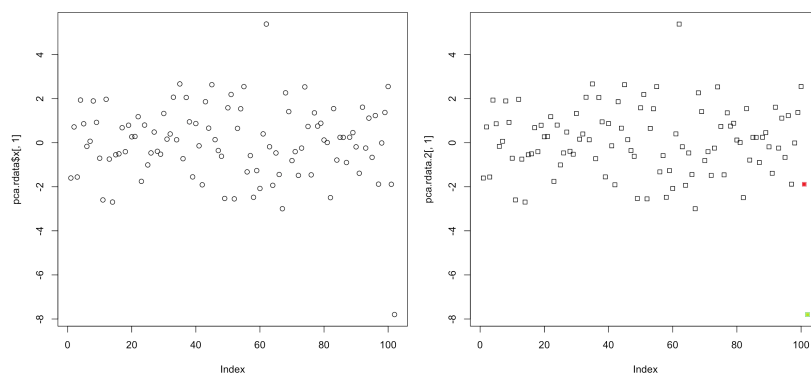
```
pca.rdata <- prcomp(scaled)
plot(pca.rdata)
```



This suggests that data is almost 1-dimensional.⁴⁰ We display the data on the first PC and highlight the “true” anomalies in the data.

40: Although the kink is on the 2nd principal component; we will revisit this soon.

```
plot(pca.rdata$x[,1])
pca.rdata.2=data.frame(pca.rdata$x[,1:4],rdata[,5])
plot(pca.rdata.2[,1], col=group, pch=22)
```



We now recreate the Mahalanobis framework on the reduced 1D data. First, we compute the empirical parameters. Next, we display the Mahalanobis distances of observations to the empirical distribution.

```
mu.2 <- mean(pca.rdata.2[,1])
Sigma.inv.2 = 1/var(pca.rdata.2[,1])

M_d.2<-vector()

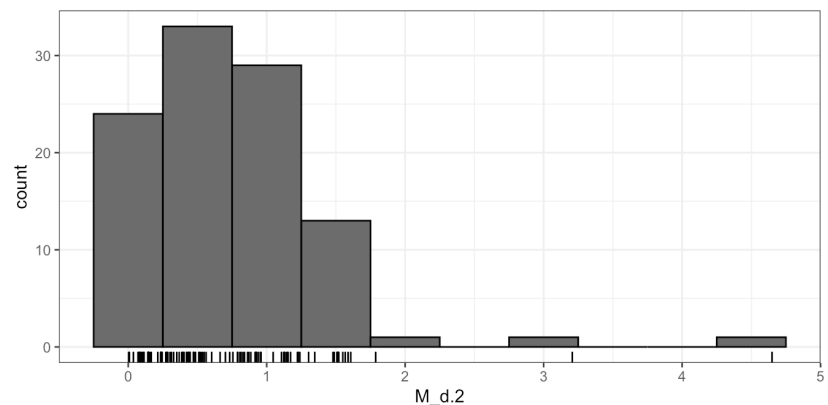
for(j in 1:nrow(rdata)){
  M_d.2[j]<-sqrt(as.matrix(pca.rdata.2[j,1]-mu.2) %*%
                Sigma.inv.2 %*%
                t(as.matrix(pca.rdata.2[j,1]-mu.2)))
}

pca.rdata.3 <- data.frame(pca.rdata.2,M_d.2)
summary(M_d)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.4622	1.3479	1.6764	1.7980	2.1010	6.6393

The corresponding histogram is given below.

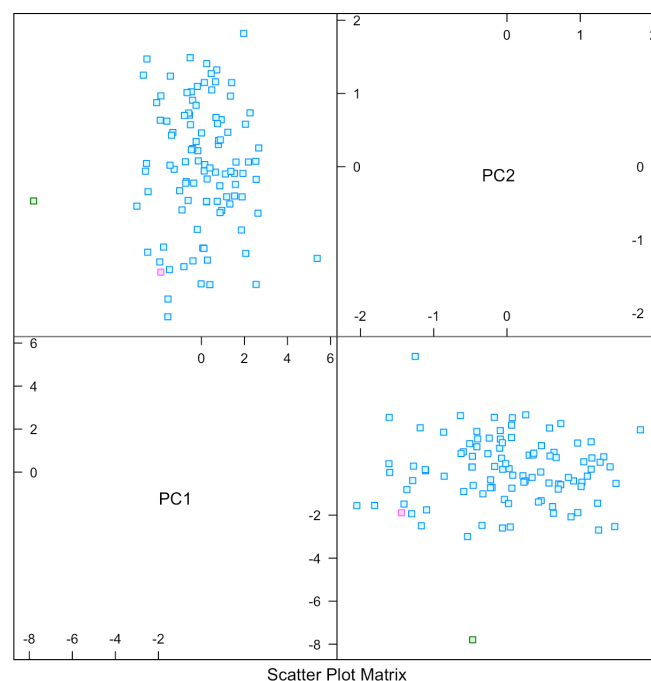
```
pca.rdata.3 |> ggplot(aes(x=M_d.2)) +
  geom_histogram(colour="black",binwidth = 0.5) +
  geom_rug() + theme_bw()
```



One outlier (observation 102, as it turns out) is still clearly visible, but observation 101 (which the Mahalanobis approach on the full unscaled dataset clearly picks out, see Section 27.2)) gets lost in the cloud of points when we only focus on the first PC. Whatever makes the latter an outlier is **out of alignment** with PC1.

Let us use PC1 **and** PC2 to see if we can find out what is going on. The anomalies are highlighted below:

```
pca.rdata.2=data.frame(pca.rdata$x[,1:4],rdata[,5])
lattice::splom(pca.rdata.2[,1:2], groups=group, pch=22)
```



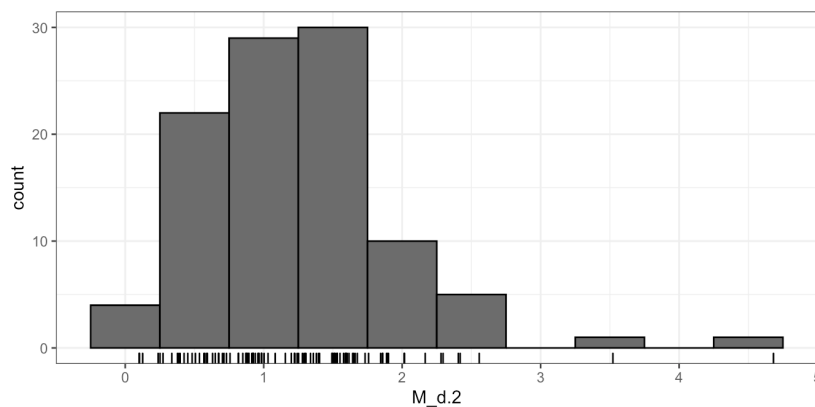
We now recreate the Mahalanobis framework on the reduced 2D data. First, we compute the empirical parameters. Next, we display the histogram of Mahalanobis distances of observations to the empirical distribution.

```
mu.2 <- colMeans(pca.rdata.2[,1:2])
Sigma.inv.2 = matlib::inv(cov(pca.rdata.2[,1:2]))

M_d.2<-vector()

for(j in 1:nrow(rdata)){
  M_d.2[j]<-sqrt(as.matrix(pca.rdata.2[j,1:2]-mu.2) %*%
                Sigma.inv.2 %*%
                t(as.matrix(pca.rdata.2[j,1:2]-mu.2)))
}

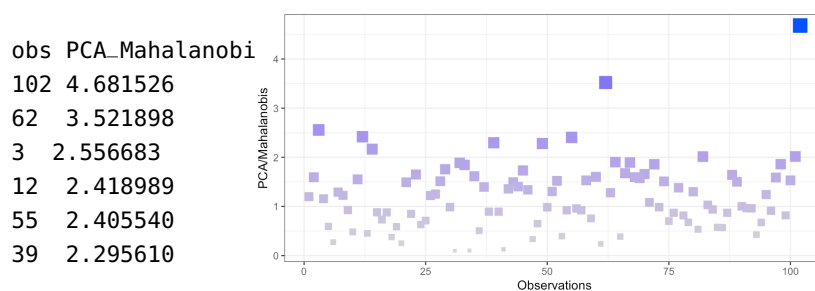
pca.rdata.3 <- data.frame(pca.rdata.2,M_d.2)
pca.rdata.3 |> ggplot(aes(x=M_d.2)) +
  geom_histogram(colour="black",binwidth = 0.5) +
  geom_rug() + theme_bw()
```



This suggests (again) that there are 2 anomalies in the data.⁴¹

41: The chart code is omitted – see previous examples for a baseline.

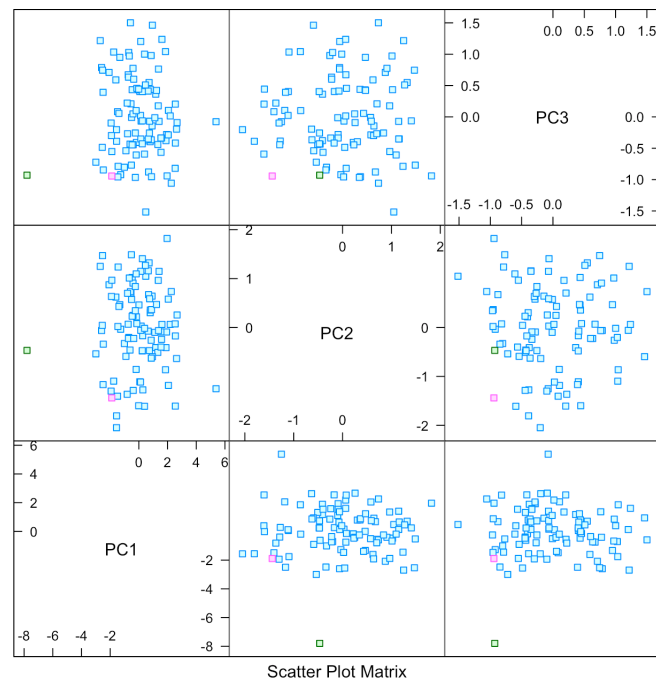
```
pca.rdata.4 = data.frame(1:(nobs+2),pca.rdata.3$M_d.2)
names(pca.rdata.4) = c("obs","PCA_Mahalanobis")
pca.rdata.4 <- pca.rdata.4[order(-pca.rdata.4$PCA_Mahalanobis),]
rownames(pca.rdata.4) <- NULL
head(pca.rdata.4)
```



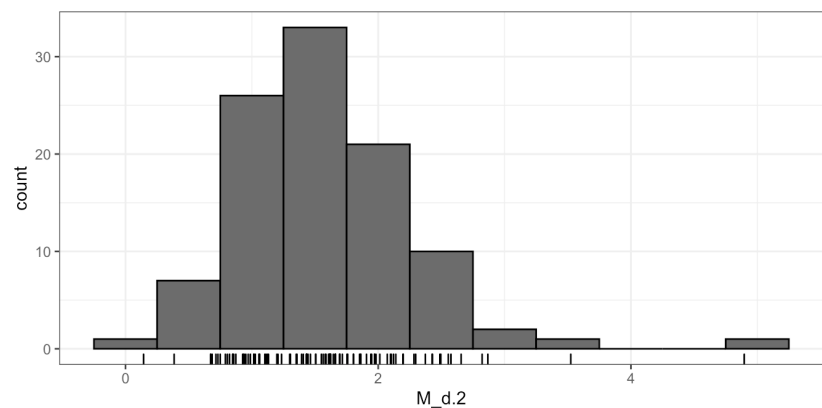
This again fails to capture observation 101 as an anomaly.

42: We omit the code as it is similar to the previous examples. Modify it as needed.

Perhaps if we use 3 PC? The anomalies are highlighted below.⁴²

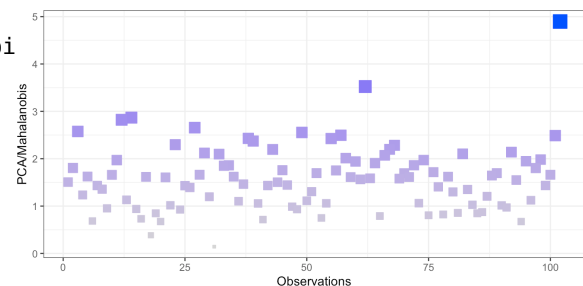


The histogram once against suggests that there are 2 anomalies in the data.



Unfortunately, the reduced 3D data once again fails to capture observation 101 as an anomaly.

```
obs PCA_Mahalanobi
102 4.896637
62 3.523881
14 2.867856
12 2.822241
27 2.656084
3 2.575923
```



Is the fact that observation 101 not captured here related to **dimension reduction**, or is it an issue related to **scaling**? Is there something that can be done to separate the two procedures? Can we get the benefits of PCA dimension reduction without having to scale the data?

Distance-Based Outlier Basis Using Neighbours As we have seen, using PCA for anomaly detection is potentially problematic: whether an observation is anomalous or not does not figure in the construction of the **principal component basis** $\{PC_1, \dots, PC_k\}$ – there is not 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 concept 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

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

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 $k \in \{k_1, \dots, k_2\}$); 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 $\{\eta_1, \dots, \eta_p\} \subset \mathbb{R}^p$ 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:

$$\begin{aligned} \mathbf{y}_{\ell_0} &= \mathbf{y}_\ell, \quad \ell = 1, \dots, M \\ \eta_1 &= \frac{\sum_{\ell=1}^M \mathbf{y}_\ell}{\left\| \sum_{\ell=1}^M \mathbf{y}_\ell \right\|_2} \\ \mathbf{y}_{\ell_1} &= \mathbf{y}_\ell - \langle \eta_1 | \mathbf{y}_\ell \rangle, \quad \ell = 1, \dots, M \\ \eta_2 &= \frac{\sum_{\ell=1}^M \mathbf{y}_{\ell_1}}{\left\| \sum_{\ell=1}^M \mathbf{y}_{\ell_1} \right\|_2} \\ &\vdots \\ \mathbf{y}_{\ell_{b-1}} &= \mathbf{y}_{\ell_{b-2}} - \langle \eta_{b-1} | \mathbf{y}_{\ell_{b-2}} \rangle, \quad \ell = 1, \dots, M \\ \eta_b &= \frac{\sum_{\ell=1}^M \mathbf{y}_{\ell_{b-1}}}{\left\| \sum_{\ell=1}^M \mathbf{y}_{\ell_{b-1}} \right\|_2}, \end{aligned}$$

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

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

$$\Theta = [\eta_1 | \dots | \eta_p]$$

is an 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 [349].

Example DOBIN is implemented in R *via* the `dobin` package. In the example below, note that the data is **not scaled**.

```
out <- dobin::dobin(rdata[,1:4], frac=0.9, norm=3)

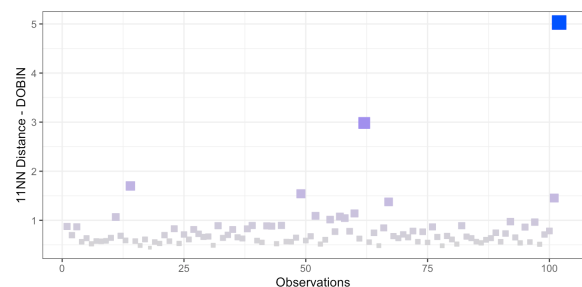
# arbitrary choice to keep kk not too large
# but respond to dataset size
kk <- min(ceiling(dim(rdata)[1]/10),25)

# dimension reduction
knn_dist <- FNN::knn.dist(out$coords[, 1:3], k = kk)
knn_dist <- knn_dist[,kk]
ord <- order(knn_dist, decreasing=TRUE)

knn_dist.dobin <- data.frame(1:(nobs+2),knn_dist)
names(knn_dist.dobin) = c("obs","knn_dobin")
knn_dist.dobin <-
  knn_dist.dobin[order(-knn_dist.dobin$knn_dobin),]
rownames(knn_dist.dobin) <- NULL
head(knn_dist.dobin)

knn_dist.dobin |>
  ggplot(aes(x=obs,y=knn_dobin)) +
    scale_x_continuous(name="Observations") +
    scale_y_continuous(name="11NN Distance - DOBIN") +
    geom_point(aes(fill=knn_dobin, colour=knn_dobin,
                  size=knn_dobin), pch=22) +
    scale_fill_continuous(high = "#0033FF",
                          low = "#CCCCCC") +
    scale_colour_continuous(high = "#0033FF",
                            low = "#CCCCCC") +
    theme_bw() + theme(legend.position = "none")
```

```
obs knn_dobin
102 5.029217
62 2.982912
14 1.700037
49 1.540807
101 1.453970
67 1.375444
```



Look: observation 101 got caught in the DOBIN net!

Algorithm 8: FeatureBagging

```

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

```

ng algorithm.

27.4.3 Subspace Methods


Subspace methods have been used particularly effectively by analysts for anomaly and outlier detection in high-dimensional datasets [350, 367, 370]; it is often easier to find the sought-after observations by exploring **lower-dimensional subspaces** (rather than the original set) – in that sense, subspace methods are akin to **feature selection** methods, whereas projection methods are closer to **dimension reduction** methods.

There is thus an intrinsic interest in exploring subspaces in their own right [335, 368]. 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 by multiple 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 formally uses the LOF algorithm of Section 27.2, but any fast anomaly detection algorithm could also be used instead. The anomaly scores and rankings from each run are aggregated as they are in the **Independent Ensemble** approach (see Section 27.4.4).

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

- **High-dimensional Outlying Subspaces (HOS)** [356];
- **Subspace Outlier Degree (SOD)** [357];
- **Projected Clustering Ensembles (OutRank)** [358];
- **Local Selection of Subspace Projections (OUTRES)** [359].

Example The feature bagging algorithm is implemented in R *via* the HighDimOut package (not available on CRAN as of May 2022, but it has been [archived](#) ). We apply it to our trusty artificial dataset.

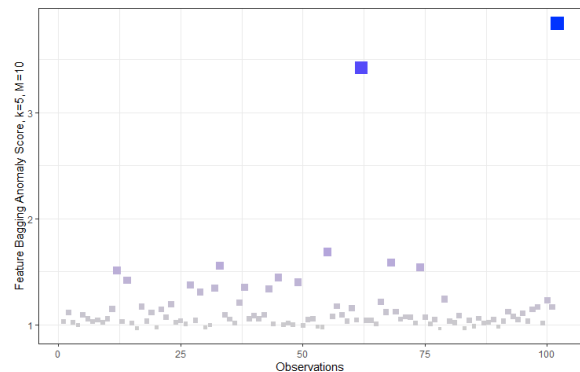
```

res.FBOD <- HighDimOut::Func.FBOD(data = rdata[,1:4],
                                iter=10, k.nn=5)
rdata.FBOD <- data.frame(1:(nobs+2),res.FBOD)
names(rdata.FBOD) = c("obs","FBOD")
rdata.FBOD <- rdata.FBOD[order(-rdata.FBOD$FBOD),]
head(rdata.FBOD)

rdata.FBOD %>%
  ggplot(aes(x=obs,y=FBOD)) +
  scale_x_continuous(name="Observations") +
  scale_y_continuous(name="Feature Bagging Anomaly Score,
                      k=5, M=10") +
  geom_point(aes(fill=FBOD, colour=FBOD, size=FBOD),
             pch=22) +
  scale_fill_continuous(high = "#0033FF",
                        low = "#CCCCCC") +
  scale_colour_continuous(high = "#0033FF",
                           low = "#CCCCCC") +
  theme_bw() + theme(legend.position = "none")

```

obs	FBOD
102	3.82
62	3.45
55	1.64
67	1.61
33	1.58
74	1.52



27.4.4 Ensemble Methods

In the preceding sections, we have described various anomaly detection algorithms whose relative performance varies with the type of data being considered. As is the case with pretty much of all of data science, it is impossible to come up with an algorithm that outperforms all the others [235].

This is because a particular anomaly detection algorithm may be well-adapted to a dataset and may be successful in detecting abnormal or outlier observations, but it may not work with other datasets whose characteristics do not match the first dataset. 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 [339].

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

Algorithm 6: SequentialEnsemble

```

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

```

Figure 27.28: Sequential ensemble algorithm.

Sequential Ensembles Sequential ensembles require a given algorithm (or a set of algorithms) to be applied to a dataset in a sequential manner, each time on 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.⁴³ The final result is either some weighted combination of all preceding results, or simply the results output by the last step in the sequence.

43: Such as AdaBoost or XGBoost, for instance.

The formal procedure is provided in Figure 27.28.⁴⁴

44: The details are out-of-scope for this chapter, but they are available in Section 21.5.3.

Independent Ensembles In an independent ensemble, we instead apply different algorithms (or different instances of the same algorithm) to the dataset (or a **resampled dataset**). Choices made at the data and algorithm level are **independent** of the results obtained in previous runs.⁴⁵ The results are then combined to obtain more robust outliers.

45: Unlike in a sequential ensemble.

Every **base** anomaly detection algorithm provides an **anomaly score** 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 and *vice versa*. In a dataset with n observations, the rank varies from 1 to n .⁴⁶

46: Ties are allowed.

Algorithm 7: IndependentEnsemble

```

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

```

Figure 27.29:
rithm.

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

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

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

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

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

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

that is, \mathbf{p}_2 is more anomalous than \mathbf{p}_1 , which is more anomalous than \mathbf{p}_3 .⁴⁷

Using the minimal rank method, we obtain

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

from which

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

and so $\mathbf{p}_1 \geq \mathbf{p}_3$ and $\mathbf{p}_2 \geq \mathbf{p}_3$, but $\mathbf{p}_1, \mathbf{p}_2$ have the same anomaly levels.

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

47: We are using the notation introduced in Section 27.1.2.

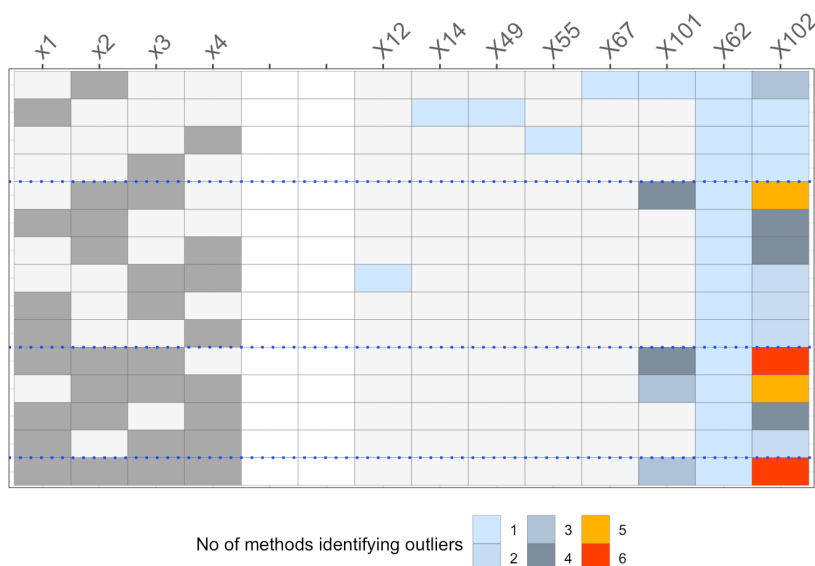
In the context of **HDLSS data**, ensemble methods can sometimes allow the analyst to mitigate some of the effects of the curse of dimensionality by selecting fast base algorithms (which can be run multiple times) and focusing on building robust relative anomaly scores.

Another suggested approach is to use a different sub-collection of the original dataset's features at each step, in in order to **de-correlate** the base detection models (see **feature bagging**, in Section 27.4.3).

Even without combining the results, it may be useful to run multiple algorithms on different subspaces to produce an **Overview of Outliers** (O3), implemented in the R package `OutliersO3`, by [A. Unwin](#).⁴⁸

```
O3d <- OutliersO3::O3prep(rdata[,1:4],
  method=c("HDo", "PCS", "BAC", "adjOut", "DDC", "MCD"))
O3d1 <- OutliersO3::O3plotM(O3d)
cx <- data.frame(outlier_method=names(O3d1$nOut),
  number_of_outliers=O3d1$nOut)
table(cx, row.names=FALSE)
O3d1$gO3
```

outlier_method	number_of_outliers
HDo	8
PCS	1
BAC	2
adjOut	2
DDC	1
MCD	2



The columns on the left indicate the **subspace variables** (see row colouring). The columns on the right indicate which **observations were identified as outliers** by at least 1 method in at least 1 subspace.⁴⁸

The **colours** depict the number of methods that identify each observation in each subspace as an outlier. For instance, Observation 102 is identified as an outlier by 6 methods in 2 subspaces, 5 methods in 3 subspaces,

48: The available methods are all methods that we have not discussed: `HDoutliers()` from the package `HDoutliers`, `FastPCS()` from the package `FastPCS`, `mvBACON()` from the package `robustX`, `adjOutlyingness()`, `covMcd()` from `robustbase`, `DetectDeviatingCells()` from `cellWise`.

4 methods in 2 subspaces, 3 methods in 1 subspace, 2 methods in 4 subspaces, and 1 method in 3 subspaces – it is clearly the **most anomalous** observation in the dataset. Observations 62 and 101 are also commonly identified as outliers.

Are the results aligned with those we have obtained throughout the chapters?

Ensemble approaches allow analysts to take a big picture view of the anomaly landscape, but it should be recalled 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.

27.5 Exercises

1. Use other metrics and parameter values to find distance-based anomalies, LOF outliers, DBSCAN/HDBSCAN/OPTICS outliers, and Isolation Forest outliers in the artificial dataset.
2. Consider the datasets:
 - [GlobalCitiesPBI.csv](#)
 - [2016collisionsfinal.csv](#)
 - [polls_us_election_2016.csv](#)
 - [HR_2016_Census_simple.xlsx](#)
 - [UniversalBank.csv](#)
 - [algae_blooms.csv](#)
 - a) Find distance-based anomalies in the datasets
 - b) Find density-based anomalies in the datasets.
 - c) Find categorical anomalies in the datasets that have categorical features.
 - d) Find projection-based anomalies in the datasets.
 - e) Find subspace-based anomalies in the datasets
 - f) Find ensemble-based anomalies in the datasets.
3. Conduct an analysis of anomalous observations in the 2011 Gap-minder data (as described in Chapters 20, 21, 22, and 23).
4. Consider the dataset [flights1_2019_1.csv](#).
 - i. Explore and visualize the dataset.
 - ii. Do any observations appear to be anomalous or outlying? Justify your answer.
 - iii. If necessary, reduce the dimension of the dataset prior to analysis.
 - iv. Using at least 4 anomaly detection algorithms, identify anomalous observations in the dataset.
 - v. Can you validate the results?