УДК 004.942 + 519.246.8
DOI: 10.26102/2310-6018/2023.43.4.030

Быстрый поиск аномалий в числовых рядах при помощи модифицированного метода Хампеля

Т.М. Гильмуллин[1]✉, М.Ф. Гильмуллин[2]

[1]Автоматизатор, независимый ИТ-эксперт, Москва, Российская Федерация
[2]Дата-сайентист, независимый ИТ-эксперт, Казань, Российская Федерация

Резюме. В статье рассмотрены и формально введены понятия аномалии числового ряда и функции-фильтра аномалий. Актуальность работы обусловлена отсутствием разработки единого подхода к пониманию понятия аномалии. В то же время они играют ключевую роль в решении многих проблем практики. В работе применяется метод измерения устойчивости выбранного способа статистической оценки на выбросы с использованием точек разрыва и скользящих окон. В основе метода фильтрации числового ряда на выбросы лежит комбинация медианы и среднего абсолютного отклонения. Применительно к решению широкого круга задач в ИТ-автоматизации предложена модификация метода Хампеля для определения выбросов в выборке. На языке Python разработаны функции фильтрации числового ряда на аномалии и определения индекса первого аномального элемента в ряду. В качестве примера на платформе Jupyter Notebook разработан сценарий для решения задачи быстрого поиска аномалий в биржевых ценах модифицированным методом Хампеля. Для получения выборки с выбросами используется авторская библиотека для генерации тестовых биржевых данных. Результаты эксперимента подтверждают, что предложенные алгоритмы позволяет четко фильтровать аномалии при различных значениях настраиваемых параметров. Отмечены достоинства и недостатки такого метода. Фильтр Хампеля легко поддается оптимизации и распараллеливанию. Материалы статьи представляют практическую ценность для решения задачи автоматизации выделения аномалий в числовых рядах.

Ключевые слова: числовые ряды, аномалии, выбросы, фильтрация, Хампель.

Для цитирования: Гильмуллин Т.М., Гильмуллин М.Ф. Быстрый поиск аномалий в числовых рядах при помощи модифицированного метода Хампеля. *Моделирование, оптимизация и информационные технологии.* 2023;11(4). URL: https://moitvivt.ru/ru/journal/pdf?id=1482 DOI: 10.26102/2310-6018/2023.43.4.030

Quick search for anomalies in number series using the modified Hampel method

T.M. Gilmullin[1]✉, M.F. Gilmullin[2]

[1]IT Automation Expert, Moscow, the Russian Federation
[2]Data Scientist, Kazan, the Russian Federation

Abstract. The article discusses and formally introduces the concepts of a number series anomaly and an anomaly filter function. The relevance of the research is due to the absence of a unified approach to understanding the concept of anomaly. At the same time, they play a key role in solving many practical problems. The study uses a method for measuring the stability of the selected method of statistical assessment for outliers using breakdown points and sliding windows. The method of filtering a number series for outliers is based on a combination of the median and the median absolute deviation. In relation to solving a wide range of issues in IT automation a modification of the Hampel method is proposed for determining outliers in a sample. Functions for filtering a number series for anomalies and determining the index of the first anomalous element are developed in Python. As an example, a script was developed using the Jupyter Notebook platform to solve the problem of quick search for anomalies in stock prices

by means of the modified Hampel method. To obtain a sample with outliers, the author's library is used to generate test stock data. The experimental results confirm that the proposed algorithms can clearly filter anomalies for different values of adjustable parameters. The advantages and disadvantages of this method are noted. The Hampel filter is easy to optimize and parallelize. The article has practical application for solving the problem of automation and identifying anomalies in number series.

Keywords: number series, anomalies, outliers, filtering, Hampel.

For citation: Gilmullin T.M., Gilmullin M.F. Quick search for anomalies in number series using the modified Hampel method. *Modeling, Optimization and Information Technology*. 2023;11(4). URL: https://moitvivt.ru/ru/journal/pdf?id=1482 DOI: 10.26102/2310-6018/2023.43.4.030 (In Russ.).

Введение

На практике встречаются задачи, для решения которых требуется найти аномалии в числовых рядах. Для простоты понимания можно считать, что это значения, которые отличаются от большинства чисел в ряде по некоторым признакам (выброс, нестандартное значение или отклонение от нормы). Такие задачи встречаются в различных областях:

– очистка зашумленных данных в дата-сайенс;

– фильтрация выбросов в обучающей выборке для нейросетей в машинном обучении;

– поиск аномальной сетевой хакерской активности при мониторинге трафика и событий в кибербезопасности;

– выявление выбросов или хвостов в потоке биржевых данных в алгоритмической торговле;

– а также в любых задачах на поиск аномалий, где данные могут быть представлены в виде числового ряда.

Понятия числового ряда в математическом анализе и в статистике отличаются. Мы принимаем под числовым рядом его статистическое понимание, то есть конечную последовательность чисел, аналог выборки.

Существуют различные толкования аномалии в числовых рядах. Большей частью они составляют временные ряды [1-4]. Временной ряд понимается как собранный в разные моменты времени статистический материал о значении каких-либо параметров исследуемого процесса. А аномалии в них определяются как образцы данных, которые не соответствуют ожидаемому, четко определенному понятию «нормального» поведения. Понятие «нормальности» и «ненормальности» со временем может меняться.

Для обнаружения аномалий используются различные методы, основанные на прогнозировании, например, статистические или использующие нейронные сети [1].

В конкретных исследованиях определяются различные оценки аномальности наблюдений. Проблемы аномальности иногда решаются за счет построения ансамблей алгоритмов – объединения результатов работы нескольких методов, выдающих различные ошибки, которые взаимно корректируются при их комбинировании [2].

Выявление аномалий часто рассматривается как задача обучения без учителя. Под анализом временных рядов многие понимают процесс применения методов статистики и машинного обучения для выявления закономерностей в их структуре и предсказания будущего поведения описываемых этими рядами систем [3].

Одно из применений аномалий – использование их в качестве диагностических индикаторов [4]. В различных областях применения аномалии называют по-разному: выбросами, противоречивыми наблюдениями, исключениями, аберрациями, неожиданностями, особенностями или примесями (anomalies, outliers, discordant observations, exceptions, aberrations, surprises, peculiarities, contaminants) [5]. Таким

образом, актуальность работы обусловлена отсутствием единого подхода к пониманию понятия аномалии.

Эти подходы частично были обобщены в [6, 7]. Нами было дано собственное определение аномалии, подходящее для практического применения в широком круге задач. Также будет дано определение и реализован фильтр аномалий, максимально абстрагированный от решаемой задачи и имеющий гибкие настройки [8].

Далее в статье будут показаны практические примеры применительно к биржевым данным, как быстро и эффективно находить аномалии в числовых рядах с помощью модифицированного метода Хампеля (F.R. Hampel) [9, 10].

Материалы и методы

Для одномерных данных оценка выбросов широко исследуется в статистической литературе. Традиционно наиболее полезными оценками для характеристики вариативности данных считаются выборочное среднее и дисперсия выборки. Они дают хорошую оценку местоположения и разброса данных в выборке, но только если выбросы ее «не загрязняют». Даже если данные содержат только одно наблюдение, которое значительно отличается от остальных в выборке, то выборочное среднее может также значительно отклоняться от среднего в выборке без этого выброса.

Чтобы измерить устойчивость выбранного метода статистической оценки к выбросам, Хампель ввел понятие точки разрыва (breakdown point). Точка разрыва — это наибольший процент «загрязненных» данных (доля неправильных наблюдений или выбросов), которые выбранный метод может обработать, прежде чем начнет выдавать неверный результат. Это могут быть произвольно большие аберрантные (отклоняющиеся от нормы, ошибочные) значения [6].

Интуитивно можно понять, что точка разрыва не может превышать 50 %, потому что если более половины наблюдений будут «загрязнены», то невозможно отличить основное распределение ряда от загрязняющего распределения. Следовательно, максимальная доля выбросов для точки разрыва равна 0,5. Существуют примеры статистик, которые достигают максимального значения для точки разрыва. Например, она равна 0,5 для медианы, а для выборочного среднего она равна $1/n$, где n – объем выборки.

Обычно считается, что чем выше значение точки разрыва, тем метод статистической оценки надежнее. Статистику с высоким значением точки разрыва иногда называют устойчивой статистикой.

Чтобы более надежно оценивать местоположение и разброс элементов выборки, часто рекомендуют комбинировать медиану и среднее абсолютное отклонение (MAD, Median Absolute Deviation). Именно они лежат в основе метода фильтрации выборки на выбросы по Хампелю.

Значение MAD вычисляется так:

$$\mathrm{MAD}\ (X) = \mathrm{Median}\ (\ |\ x_1 - \mathrm{Median}\ (X)\ |, \ldots, |\ x_n - \mathrm{Median}\ (X)\ |\ ), \qquad (1)$$

где X – выборка из n наблюдений $x_1, \ldots, x_n$.

По Хампелю, под выбросом (outlier) выборки понимается такое число x из выборки X, модуль разности которого и медианы выборки больше, чем MAD выборки, вычисленное по формуле (1) и умноженное на специальный коэффициент k (scale factor), зависящий от распределения ($\approx$1,4826 для нормального) [7].

На практике для построения фильтра выбросов Хампеля это определение модифицируется с использованием метода скользящих окон (sliding windows) и параметра s (sigma), порогового количества стандартных отклонений. Это значит, что

медиана и MAD вычисляются для всех скользящих окон, состоящих из элементов выборки. Далее все модули разности элементов выборки и медианы сравниваются с MAD, умноженный на коэффициент k и на параметр s [8].

Более высокий порог s стандартного отклонения делает фильтр более щадящим, а более низкий порог идентифицирует больше чисел как выбросы.

Теперь определимся с основным понятием: что понимается под аномалией в конечном числовом ряду для практических задач.

Определение 1. Аномалией (anomaly) числового ряда называется такое число a из ряда X, которое по модулю отличается от медианы скользящего окна более чем в s раз от MAD этого окна, умноженного на коэффициент k.

Для того чтобы максимально обобщить фильтр аномалий и абстрагировать его от решаемой практической задачи, предлагаем следующее теоретико-множественное толкование этого определения.

Согласно определению 1, все аномалии ряда образуют некоторое его подмножество A:

$$A = \{ a \in X \mid | a - \text{Median}(W_i) | > s \cdot k \cdot \text{MAD}(W_i) \}, \quad (2)$$

где множество X – исходный числовой ряд, множество W_i – ряд чисел i-го скользящего окна, а всего окон: $(n - w + 1)$, где w – размер окна.

Определение 2. Фильтром аномалий числового ряда называется функция:

$$F: X \to \{\text{True}, \text{False}\},$$

$$F(x_i) = \begin{cases} \text{True}, & \text{если } x_i \in A, \\ \text{False}, & \text{если } x_i \notin A, \end{cases} \quad (3)$$

где X – исходный числовой ряд, состоящий из n элементов x_i, A – множество аномалий (2).

Для фильтрации числового ряда на наличие в нем аномалий предлагается использовать авторскую реализацию функции HampelFilter()[1] на Python. Эта функция позволяет обнаружить аномалию среди значений любого числового ряда, используя модифицированный метод фильтрации по Хампелю. Фильтр обнаруживает все аномалии, определяемые согласно выражению (2).

Настраиваемые параметры в этой функции-фильтре:

– window (переменная w) – размер скользящего окна (по умолчанию 5);

– sigma (переменная s) – пороговое количество стандартных отклонений (по умолчанию 3);

– scaleFactor (переменная k) – специальный коэффициент, зависящий от распределения ряда (по умолчанию1.4826).

Функция фильтрации HampelFilter() возвращает новый ряд F, согласно выражению (3) состоящий из значений True или False, где True означает наличие аномального элемента на соответствующей позиции в исходном ряду.

На практике часто бывает нужно определить только первый аномальный или первый среди наибольших в числовом ряду элемент. Для этого предлагается использовать авторскую реализацию функции HampelAnomalyDetection()[2] на Python. Эта функция возвращает минимальный индекс элемента в списке найденных аномалий или

[1] HampelFilter() — функция для анализа числовых рядов на наличие в них аномалий содержится в открытом доступе в Python-библиотеке TradeRoutines. [Электронный ресурс]. URL: tim55667757.github.io/TKSBrokerAPI/docs/tksbrokerapi/TradeRoutines

[2] Функция HampelAnomalyDetection() содержится в той же библиотеке TradeRoutines.

индекс первого максимального элемента во входном ряду, если его индекс меньше индекса аномального элемента. Если в числовом ряду нет аномалий или все элементы равны (нет максимумов), то возвращается None.

Результаты

Будет проще понять, как работает функция HampelFilter() и что считает аномалией, на конкретных примерах числовых рядов. Приведем их в формате Python-скрипта.

Сначала рассмотрим фильтрацию числовых рядов со значениями параметров по умолчанию.

HampelFilter(window = 5, sigma = 3, scaleFactor = 1,4826)

```
import pandas as pd
from tksbrokerapi.TradeRoutines import HampelFilter
testData1 = [
    # Все элементы равны, нет ни одного
    # «подозрительного» элемента, значит
    # и аномалий быть не должно:
    pd.Series([10, 10, 10, 10, 10]),
    # Здесь 1-й элемент явно выбивается
    # из общего ряда, поэтому он аномальный:
    pd.Series([1, 10, 10, 10, 10]),
    # В этом ряду больше элементов равных 10,
    # а первые два выбиваются из общей картины,
    # значит они оба аномальные:
    pd.Series([1, 5, 10, 10, 10]),
    # Аналогично, как во втором примере,
    # 2-й элемент аномальный, потому что
    # отличается от большинства в ряду:
    pd.Series([1, 5, 1, 1, 1]),
]
for i, test in enumerate(testData1):
    print("Input {}: {}".format(i + 1, list(test)))
    print("--> {}".format(list(HampelFilter(test))))
```

Для проверки предположений, какие из чисел в рядах являются аномалиями, запускаем код выше и получаем ожидаемый результат:

```
>>> Input 1: [10, 10, 10, 10, 10]
>>> --> [False, False, False, False, False]
>>> Input 2: [1, 10, 10, 10, 10]
>>> --> [True, False, False, False, False]
>>> Input 3: [1, 5, 10, 10, 10]
>>> --> [True, True, False, False, False]
>>> Input 4: [1, 5, 1, 1, 1]
>>> --> [False, True, False, False, False]
```

По умолчанию размер скользящего окна равен 5, то есть совпадает с длиной рядов в примере. Посмотрим, что изменится, если установить размер скользящего окна равным 3, сначала на тех же рядах, а затем добавим новые.

HampelFilter(window = 3, sigma = 3, scaleFactor = 1.4826):

```
testData2 = [
    # Не должно ничего измениться,
    # т. к. здесь нет аномалий:
    pd.Series([10, 10, 10, 10, 10]),
    # Нет изменений, т. к. здесь
    # один аномальный элемент:
```

```
    pd.Series([1, 10, 10, 10, 10]),
    # А здесь первые элементы уже не должны
    # считаться аномалией при sigma = 3:
    pd.Series([1, 5, 10, 10, 10]),
    # Нет изменений, т. к. здесь один
    # явно выраженный аномальный элемент:
    pd.Series([1, 5, 1, 1, 1]),
    # Новые числовые ряды:
    pd.Series([1, 10, 10, 1, 10, 1]),
    pd.Series([1, 10, 10, 10, 10, 1]),
    pd.Series([1, 1, 1, 10, 10, 10]),
]
for i, test in enumerate(testData2):
    print("Input {}: {}".format(i + 1, list(test)))
    print("--> {}".format(list(HampelFilter(test, window=3))))
```

Запустив код, выше получим следующий результат:

```
>>> Input 1: [10, 10, 10, 10, 10]
>>> --> [False, False, False, False, False]
>>> Input 2: [1, 10, 10, 10, 10]
>>> --> [True, False, False, False, False]
>>> Input 3: [1, 5, 10, 10, 10]
>>> --> [False, False, False, False, False]
>>> Input 4: [1, 5, 1, 1, 1]
>>> --> [False, True, False, False, False]
>>> Input 5: [1, 10, 10, 1, 10, 1]
>>> --> [True, False, False, False, False, False]
>>> Input 6: [1, 10, 10, 10, 10, 1]
>>> --> [True, False, False, False, False, True]
>>> Input 7: [1, 1, 1, 10, 10, 10]
>>> --> [False, False, False, False, False, False]
```

В первых двух и четвертом рядах изменений по сравнению с предыдущим примером нет. Обратите внимание на 3-й числовой ряд. В нем первые два элемента, которые раньше считались как аномальные, теперь игнорируются. Это следует из формул (1) и (2) очевидным образом.

Аналогично, и в 5-м числовом ряду получился только один аномальный элемент, хотя на первый взгляд может показаться, что должны быть еще. Кроме того, при слишком малом размере скользящего окна не удалось выявить аномалии в 7-м ряду.

Для понимания принципа работы функции HampelAnomalyDetection() рассмотрим несколько примеров. Важное отличие от предыдущей функции HampelFilter() заключается в том, что здесь учитывается также наличие и местоположение максимальных элементов ряда.

HampelAnomalyDetection(), с параметрами по умолчанию:

```
import pandas as pd
from tksbrokerapi.TradeRoutines import HampelAnomalyDetection
testData3 = [
    pd.Series([1, 1, 1, 1, 111, 1]),
    pd.Series([1, 1, 10, 1, 1, 1]),
    pd.Series([111, 1, 1, 1, 1, 111]),
    pd.Series([1, 11, 1, 111, 1, 1]),
    pd.Series([1, 2]),
    pd.Series([1, 1, 1, 1, 1, 1]),
]
for i, test in enumerate(testData3):
    print("Input {}: {}".format(i + 1, list(test)))
    print("--> {}".format(HampelAnomalyDetection(test)))
```

Запустив код, выше получим следующий результат:

```
>>> Input 1: [1, 1, 1, 1, 111, 1] --> 4
>>> Input 2: [1, 1, 10, 1, 1, 1] --> 2
>>> Input 3: [111, 1, 1, 1, 1, 111] --> 0
>>> Input 4: [1, 11, 1, 111, 1, 1] --> 1
>>> Input 5: [1, 2] --> None
>>> Input 6: [1, 1, 1, 1, 1, 1] --> None
```

Обсуждение

Для интерпретации полученных результатов исследования рассмотрим фильтрацию по Хампелю на примере практической задачи выявления аномалий в числовом ряде биржевых цен. Для получения нужного нам ряда с выбросами воспользуемся авторской библиотекой для генерации тестовых биржевых данных PriceGenerator[3].

Обычно трейдеры и биржевые аналитики используют ценовую модель в виде временного ряда OHLCV-candlesticks (open, high, low, close, volume), так называемых японских свечей. Одна строка таких данных представляет собой набор цен для построения одной свечи: дата открытия, цена открытия, наибольшая цена, наименьшая цена, цена закрытия на данном временном интервале и значение объема торгов за период от открытия до закрытия.

PriceGenerator – это платформа, которую можно использовать как модуль Python или запускать из командной строки и генерировать случайные ценовые данные, максимально похожие на «настоящие» цены, но с заранее заданными статистическими характеристиками. В PriceGenerator можно задать общий тренд движения цен, таймфрейм свечей, максимальное и минимальное значения для диапазона цен, максимальный размер свечей, вероятность направления очередной свечи, вероятность ценовых выбросов, количество генерируемых свечей и множество других параметров.

Для проведения эксперимента на платформе Jupyter Notebook был разработан сценарий HampelFilteringExample[4], с примером использования функции HampelFilter() – модифицированного фильтра Хампеля – для решения задачи быстрого поиска аномалий в биржевых ценах.

В сценарии генерируется ценовой ряд со следующими характеристиками:

– цены на графике состоят только из целых чисел (для простоты);
– интервал свечей и горизонт генерации: 1 день, 75 свечей;
– минимальная и максимальная цены close: от 40 до 140;
– начать ряд с цены close: 50;
– максимальный выброс хвостов свечей: 35;
– максимальный размер тела свечей: 25;
– вероятность того, что очередная свеча будет вверх: 51,5 %;
– вероятность того, что очередная свеча будет иметь выброс: 10 %;
– общий тренд: сначала падающий (40 свечей), затем растущий (35 свечей).

Для получения таких характеристик PriceGenerator запускается с параметрами:

[3] PriceGenerator — программная платформа на Python для генерации временных рядов, похожих на случайные биржевые цены с аномалиями. [Электронный ресурс]. URL: github.com/Tim55667757/PriceGenerator/blob/master/README_RU.md

[4] HampelFilteringExample.ipynb — сценарий на Jupyter Notebook с примером использования фильтра Хампеля для задачи поиска выбросов в биржевых ценах. [Электронный ресурс]. URL: nbviewer.org/github/Tim55667757/TKSBrokerAPI/blob/develop/docs/examples/HampelFilteringExample.ipynb

```
PriceGenerator --debug-level 10 --ticker "TEST_DATA_OF_OHLCV" --precision 0
--timeframe 1440 --horizon 75 --max-close 140 --min-close 40 --init-close 50
--max-outlier 35 --max-body 25 --max-volume 4000000 --up-candles-prob 0.515
--outliers-prob 0.1 --trend-deviation 0.005 --split-trend "down-up" --split-
count 40 35 --generate --render-google index.html
```

После генерации OHLCV-свечи сохраняются в формате Pandas DataFrame и пригодны для дальнейшего анализа в Python.

Чтобы получить представление о полученном с помощью PriceGenerator ряде, строится статический или интерактивный графики и автоматически определяются некоторые статистические характеристики ряда, как показано на Рисунке 1.

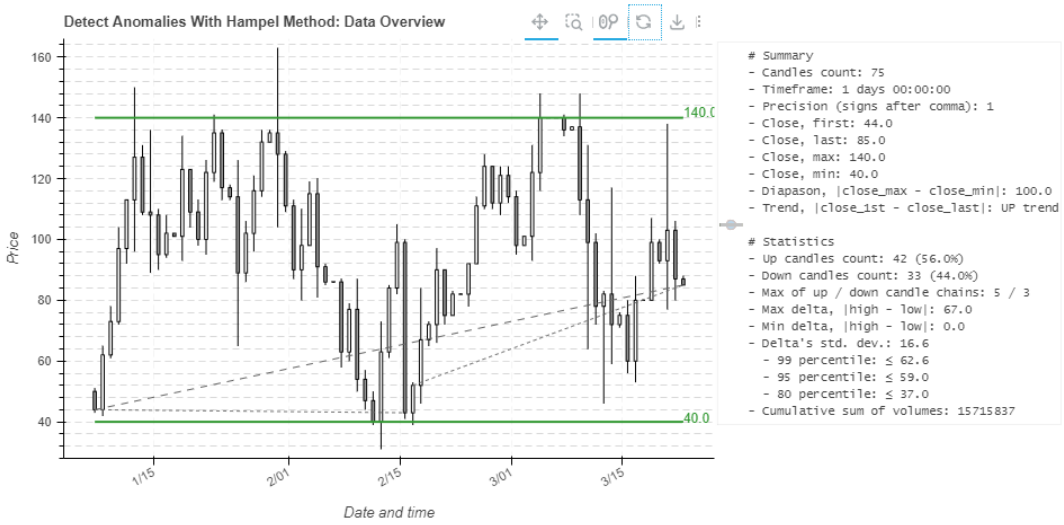

Рисунок 1 – Сгенерированный с помощью PriceGenerator ряд свечей
Figure 1 – Series of candles generated using PriceGenerator

Визуально наблюдается, что в ряду на графике цен явно присутствуют выбросы – сверху и снизу у некоторых свечей. Нас интересуют не все такие выбросы, а только слишком длинные хвосты (тени свечей) или слишком большие размеры тела свечи.

Далее датафрейм с OHLCV-свечами «очищается» от лишних данных стандартными средствами библиотеки Pandas и оставляются только ценовые столбцы. Затем вычисляются зависимые от значений OHLCV параметры:

– размеры тел свечей: $\mathrm{body} = |\mathrm{close} - \mathrm{open}|$;

– верхние тени свечей: $\mathrm{upper} = \mathrm{high} - \max(\mathrm{open}, \mathrm{close})$;

– нижние тени свечей: $\mathrm{lower} = \min(\mathrm{open}, \mathrm{close}) - \mathrm{low}$.

Эти расчеты необходимы для того, чтобы потенциальные аномальные элементы, ранее обнаруженные визуально на графике, были добавлены в датафрейм в виде элементов числовых рядов. А их уже можно отфильтровать функцией HampelFilter().

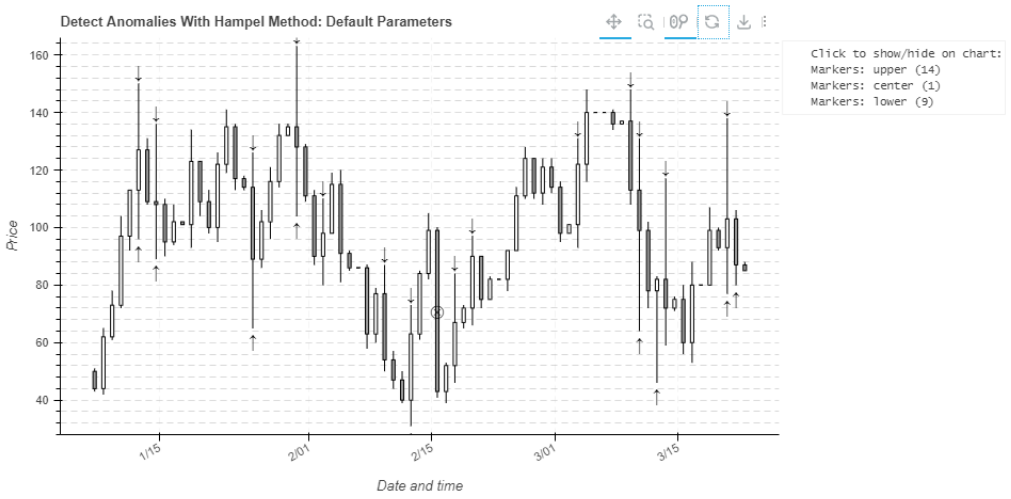

Рисунок 2 – Результаты фильтрации с параметрами по умолчанию
Figure 2 – Filtering results with default parameters

Выясним, как справляется с обнаружением аномальных ценовых выбросов и размеров свечей модифицированный фильтр Хампеля с параметрами по умолчанию (window = 5, sigma = 3, scaleFactor = 1,4826).

Как было показано в примерах выше, аномальными могут быть признаны как очень маленькие элементы, так и очень большие (по сравнению с остальными элементами в скользящем окне). Отметим аномальные свечи символом ⊗, нижние тени – символом ↑, верхние тени – символом ↓, и посмотрим на результаты на Рисунке 2.

На первый взгляд кажется, что фильтр справился неплохо: аномалии найдены и отмечены. Однако для наблюдателя, который не знает про малый размер скользящего окна, такие результаты будут неочевидны. Кажется, что на графике отмечены лишние элементы. Попробуем расширить скользящее окно на весь числовой ряд (в нашем примере, window = 75) и заново определить аномальные элементы, как на Рисунке 3.

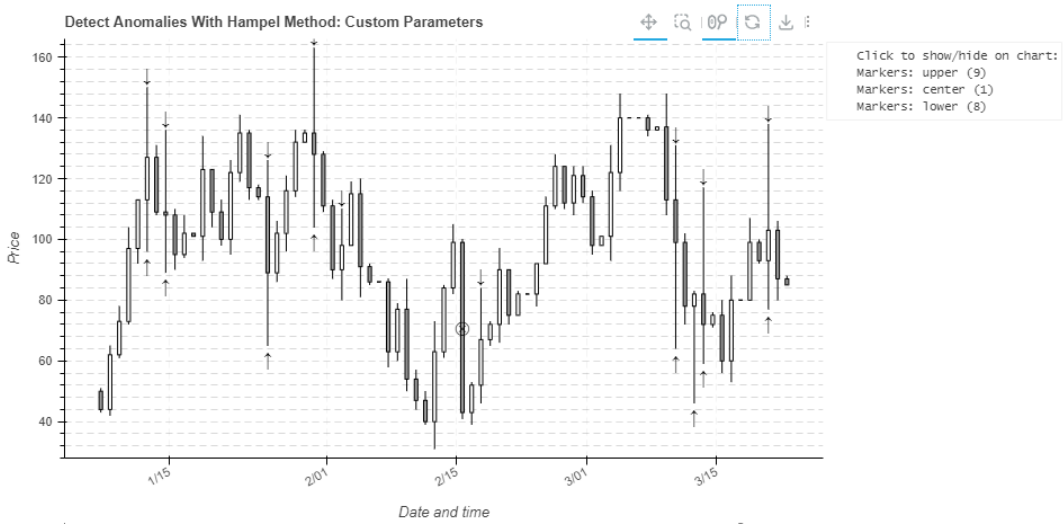

Рисунок 3 – Результаты фильтрации после расширения скользящего окна
Figure 3 – Filtering results after extending the sliding window

Теперь видно, что на графике отмечены только те элементы, про которые большинство наблюдателей, оценивающих картину в целом, скорее всего, сказали бы, что это аномалии (по сравнению с другими элементами в обозримом ряду).

Заключение

Конечно, у метода Хампеля есть и недостатки. Например, при реализации алгоритма можно столкнуться с тем, что если аномалии присутствуют в первом или последнем элементах ряда, то по оригинальному алгоритму они будут проигнорированы. Это связано с использованием скользящего окна. Чтобы обойти проблему, в авторской реализации функции HampelFilter() нам пришлось предварительно расширять числовые ряды спереди и сзади на размер этого окна. Только после этого удалось обнаружить аномалии и в первых, и в последних элементах исходных рядов.

Однако на практике фильтр Хампеля оказывается чрезвычайно эффективным. Благодаря современным библиотекам, таким как Pandas, он справляется с достаточно большими числовыми рядами на десятки тысяч элементов за считанные секунды, а также легко поддается оптимизации и распараллеливанию (например, с помощью CUDA Python и декоратора @cuda.jit от Numba, либо Python Multiprocessing ThreadPool).

Таким образом, задача поиска аномалий в числовом ряду эффективно решается фильтрацией модифицированным методом Хампеля. Он дает быстрые результаты, а также прост в понимании и реализации.

СПИСОК ИСТОЧНИКОВ

1. Laxman S., Sastry P.S. A survey of temporal data mining. *Sadhana*. 2006;31:173–198. DOI: 10.1007/BF02719780.
2. Чесноков М.Ю. Поиск аномалий во временных рядах на основе ансамблей алгоритмов DBSCAN4 Москва; 2018. URL: http://www.isa.ru/aidt/images/documents/2018-01/99-107.pdf (дата обращения 01.10.2023).
3. Мастицкий С.Э. Анализ временных рядов с помощью R; 2020. URL: https://ranalytics.github.io/tsa-with-r/ch-anomaly-detection.html (дата обращения 01.10.2023).
4. Ardelean V. Outliers in Time Series. Department of Statistics and Econometrics, University of Erlangen-Nuremberg; 2011. URL: https://www.statistik.rw.fau.de/files/2016/03/v01-2011.pdf (дата обращения 01.10.2023).
5. Chandola V., Banerjee A., Kumar V. Anomaly detection: a survey, ACM Computing Surveys; 2009. URL: http://cucis.ece.northwestern.edu/projects/DMS/publications/AnomalyDetection.pdf (дата обращения 01.10.2023).
6. Hampel F.R. The influence curve and its role in robust estimation. *Journal of the American Statistical Association*. 1974;69:383–393. DOI: 10.2307/2285666.
7. Liu H., Shah S., Jiang W. On-line outlier detection and data cleaning. *Computers & Chemical Engineering*. 2004;28(9):1635–1647. URL: https://sites.ualberta.ca/~slshah/files/on_line_outlier_det.pdf (дата обращения: 01.10.2023).
8. Lewinson E. *Python for Finance Cookbook — Second Edition*. Birmingham, Packt; 2022. 740 p.
9. Hampel F.R. A general qualitative definition of robustness. *Ann. Math. Stat.* 1971;42:1887–1896.
10. Hampel F.R., Rousseeuw P.J., Ronchtti E.M., Stahel W.A. *Robust Statistic: The Approach Based on Influence Functions*. New York, Wiley & Sons; 1986. 536 p.

REFERENCES

1. Laxman S., Sastry P.S. A survey of temporal data mining. *Sadhana*. 2006;31:173–198. DOI: 10.1007/BF02719780.
2. Chesnokov M.Ju. Poisk anomalij vo vremennyh rjadah na osnove ansamblej algoritmov DBSCAN4 Moscow; 2018. URL: http://www.isa.ru/aidt/images/documents/2018-01/99-107.pdf (accessed on 01.10.2023).
3. Mastickij S.Je. Analiz vremennyh rjadov s pomoshh'ju R; 2020. URL: https://ranalytics.github.io/tsa-with-r/ch-anomaly-detection.html (accessed on 01.10.2023).
4. Ardelean V. Outliers in Time Series. Department of Statistics and Econometrics, University of Erlangen-Nuremberg; 2011. URL: https://www.statistik.rw.fau.de/files/2016/03/v01-2011.pdf (accessed on 01.10.2023).
5. Chandola V., Banerjee A., Kumar V. Anomaly detection: a survey, ACM Computing Surveys; 2009. URL: http://cucis.ece.northwestern.edu/projects/DMS/publications/AnomalyDetection.pdf (accessed on 01.10.2023).
6. Hampel F.R. The Influence curve and its role in robust estimation. *Journal of the American Statistical Association*. 1974;69:383–393. DOI: 10.2307/2285666.
7. Hancong Liu, Sirish Shah and Wei Jiang. On-line outlier detection and data cleaning. *Computers & Chemical Engineering*. 2004;28(9):1635–1647. URL: https://sites.ualberta.ca/~slshah/files/on_line_outlier_det.pdf (accessed on 01.10.2023).
8. Lewinson E. *Python for Finance Cookbook — Second Edition*. Birmingham, Packt; 2022. 740 p.
9. Hampel F.R. A general qualitative definition of robustness. *Ann. Math. Stat.* 1971;42:1887–1896.
10. Hampel F.R., Rousseeuw P.J., Ronchtti E.M., Stahel W.A. *Robust Statistic: The Approach Based on Influence Functons*. New York, Wiley & Sons; 1986. 536 p.

ИНФОРМАЦИЯ ОБ АВТОРАХ / INFORMATION ABOUT THE AUTHORS

Гильмуллин Тимур Мансурович, кандидат технических наук, автоматизатор, независимый ИТ-эксперт, Москва, Российская Федерация.
e-mail: tim55667757@gmail.com

Timur M. Gilmullin, Candidate of Technical Sciences, IT Automation Expert, Moscow, the Russian Federation.

Гильмуллин Мансур Файзрахманович, кандидат педагогических наук, доцент, дата-сайентист, независимый ИТ-эксперт, Казань, Российская Федерация.
e-mail: gilmullin.mansur@gmail.com
ORCID: 0000-0002-4134-2133

Mansur F. Gilmullin, Candidate of Pedagogical Sciences, Associate Professor, Data Scientist, Kazan, the Russian Federation.

Статья поступила в редакцию 04.12.2023; одобрена после рецензирования 08.12.2023; принята к публикации 20.12.2023.

The article was submitted 04.12.2023; approved after reviewing 08.12.2023; accepted for publication 20.12.2023.