Hochschule Esslingen
University of Applied Sciences

Forschungsprojekt

Ausreißer-Erkennung in Zeitreihen mittels Graphen-basierter Algorithmen

im Studiengang Angewandte Informatik
der Fakultät Informationstechnik
Wintersemester 2020/2021

Bahar Uzun
764647
Jeremy Kielman
764097
Marcus Erz
762294

Abgabedatum: 28. Februar 2021
Prüferin: Prof. Dr. rer. nat. Gabriele Gühring

Kurzfassung

todo: Kurzfassung erstellen

Schlagwörter: Anomalie-Erkennung, Ausreißer-Erkennung, Netsimile, MIDAS, Random Walk, Graphen-basierte Algorithmen, Zeitreihen

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

1 Einleitung

Im Rahmen der Forschungsprojekt werden verschiedene Algorithmen zur Ausreißer-Erkennung in Graphen erforscht und getestet. Nachfolgend soll die Motivation hinter dieser Thematik erläutert werden.

1.1 Hintergrund

todo: formulieren

1.2 Problemstellung

todo: Ziele definieren Das Ziel dieser Forschungsprojekt ist es verschiedene Algorithmen anzuwenden und erste Erkenntnisse aus ihnen zu gewinnen. Dieses Hauptziel, im Zuge des ersten Semesters des Forschungsprojekts, kann wie folgt in drei Teilziele unterteilt werden:

1. Verschaffen eines Überblicks über die existierenden Algorithmen zur Erkennung von Ausreißern in Graphen
2. Die Entwicklung eines Ausreißer-Scores für die zugrundeliegenden Algorithmen
3. Erste Anwendung der verwendeten Graphen-basierten Algorithmen auf Zeitreihen

1.3 Verwandte Arbeiten

todo: related work einfügen

2 Transformation Zeitreihe zu Netzwerk

Ziel unseres Forschungsprojektes ist es unter anderem verschiedene Algorithmen, zur Ausreißer Erkennung in Netzwerken, auf Zeitreihendaten anzuwenden. Als erstes müssen hierzu die Zeitreihen in ein Netzwerk umgewandelt werden, dieser Schritt wird in diesem Kapitel erläutert. Je nach Ausreißer-Erkennung Algorithmus, muss die Transformation leicht unterschiedlich durchgeführt werden. Aus diesem Grund wird in Kap. 2.1, Kap. 2.2 und Kap. 2.3 erläutert wie die Umwandlung für die jeweiligen Algorithmen funktioniert.

2.1 Netsimile

Der erste Schritt der Transformation ist, die Zeitreihe in kleinere Intervalle aufzusplitten. Anschließend kann für jedes der Intervalle ein Netzwerk berechnet werden. Die Länge des Intervalls kann als Hyperparameter an den Algorithmus übergeben werden. Je nach Zeitreihe funktionieren unterschiedliche Intervallgrößen besser oder schlechter. Insofern die Zeitreihe eine Saisonalität aufweist, kann diese bestimmt und als Intervallgröße genutzt werden. todo: Überprüfen ob Saisonalität der richtige Begriff ist.
Um die einzelne Zeitintervalle in ein Netzwerk umzuwandeln, wird zunächst die Distanz zwischen den einzelnen Elementen des Zeitintervalls berechnet. Hierzu wird auf das in (vgl. Amil et al. 2019, S. 2-3) vorgestellte Distanzmaß zurückgegriffen. Insofern für p = 2 eingesetzt wird, handelt es sich um die euklidische Distanz. Die Abstände bilden die Kantengewichte zwischen den jeweiligen Elementen im Netzwerk. Die Elemente der Zeitreihe bilden die Knoten des Netzwerks. Die Netzwerke werden intern als Adjazenzmatrizen gespeichert.

$$D_{ij} = \left(\sum_k \left| v_k^i - v_k^j \right|^p \right)^{1/p}$$

Im nächsten Schritt müssen die Netzwerke in CSV-Dateien gespeichert werden, sodass der Netsimile Algorithmus die Daten einlesen kann. Dazu wird für jede Kante des Netzwerks eine Zeile im der Datei, mit folgendem Format generiert: Ursprungsknoten, Zielknoten, Gewichtung. Für jedes Zeitintervall muss eine einzelne CSV-Datei angelegt werden. Der Netsimile Algorithmus vergleicht...

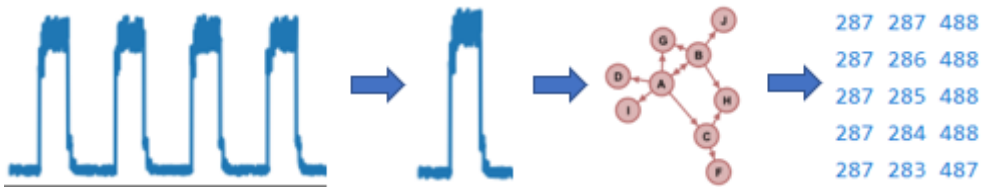


Abb. 2.1: Umwandlung einer Zeitreihe in Netzwerk

todo: Das Netzwerk aus der Grafik noch abändern

2.2 MIDAS

Die Transformation der Zeitreihe in mehrere Netzwerke funktioniert für den MIDAS Algorithmus gleich wie in Kap. 2.1. Allerdings kann der Algorithmus teilweise bessere Ergebnisse erzielen, wenn für p eine Zahl größer als zwei eingesetzt wird. Dadurch werden größere Abstände zwischen Elementen stärker gewichtet.
Außerdem erwartet der MIDAS Algorithmus für die Netzwerkdaten ein anderes Übergabeformat. Hierbei können alle Daten der jeweiligen Zeitabschnitte in einen CSV-File geschrieben werden. Die CSV-Datei muss dabei folgendermaßen strukturiert sein: Ursprungsknoten, Zielknoten, Zeitintervall. Es ist nicht möglich die Kantengewichtung direkt an den Algorithmus zu übergeben. Um die Kantengewichtung trotzdem übergeben zu können, wird die gleiche Kante mehrmals in Abhängigkeit der Gewichtung an den Algorithmus übergeben. Wie funkt Midas ganz kurz..

```
284 174 13
284 174 13
284 174 13
284 175 13
284 175 13
284 175 13
284 175 13
284 175 13
284 175 13
284 175 13
284 175 13
284 176 13
284 176 13
```

Abb. 2.2: Datensatz Midas

todo: Vielleicht kleineren Auszug aus Datensatz verwenden

2.3 MIDAS-R

todo: Hab hier das mit der Hauptkomponentenzerlegung gemacht. Wenn es Ergebnisse hierfür gibt. Kann ich das hier noch erklären

3 Netsimile

todo: In diesem Kapitel werden grundlegende Themen behandelt, die im Rahmen des Forschungsprojekts zum Verständnis der Ausreißer-Erkennung in Graphen gedient haben.

3.1 Grundlagen

todo: Einführung in den Algorithmus

3.1.1 Canberra Distance

Einführung

todo: Stichworte sammeln

3.2 Ergebnisse Zeitreihe

Ausreißer Typ	**Datei Name**	**1D**	**2D**
Einzelne Peaks	anomaly-art-daily-peaks	*	*
Zunahme an Rauschen	anomaly-art-daily-increase-noise	****	*
Signal Drift	anomaly-art-daily-drift	***	*
Kontinuierliche Zunahme der Amplitude	art-daily-amp-rise	***	*
Zyklus mit höherer Amplitude	art-daily-jumpsup	****	*
Zyklus mit geringerer Amplitude	art-daily-jumpsdown	****	*
Zyklus-Aussetzer	art-daily-flatmiddle	****	*
Signal-Aussetzer	art-daily-nojump	-	*
Frequenzänderung	anomaly-art-daily-sequence-change	-	*

4 MIDAS

todo: In diesem Kapitel werden grundlegende Themen behandelt, die im Rahmen des Forschungsprojekts zum Verständnis der Ausreißer-Erkennung in Graphen gedient haben.

Erst erklären wie der MIDAS funktioniert. Und zum Laufen gebracht mit Graphen über die Zeit ENRON & DARPA. Im Anschluss auf Zeitreihendaten angewendet.

4.1 Grundlagen

todo: Einführung in den Algorithmus

4.1.1 Count-min Sketch

Einführung

todo: Stichworte sammeln

4.2 Ergebnisse Ausreißererkennung in Zeitreihen

uiebwiebdciqbewd

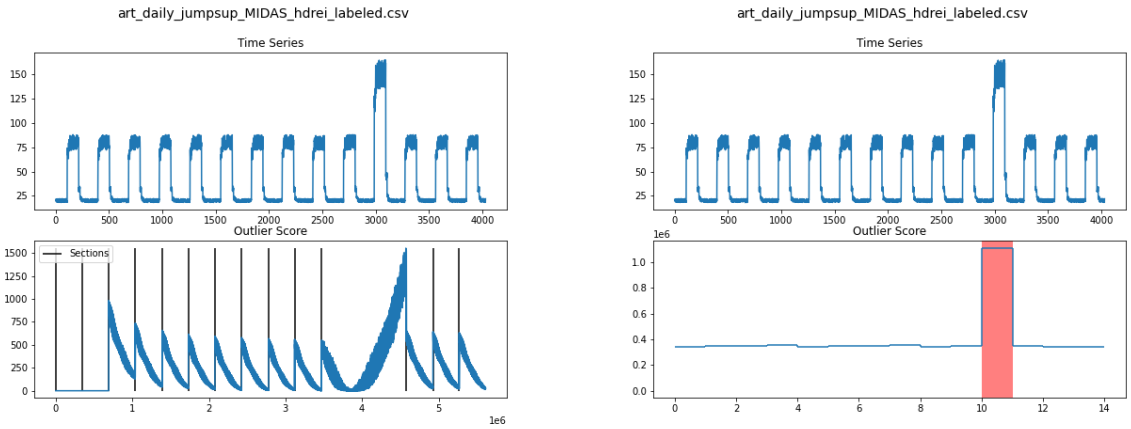


Abb. 4.1: Vergelich Perculation Algorithmus mit Sliding Window Verfahren und ohne Sliding Window Verfahren

5 Statische Algorithmen

todo: Labels für die einzelnen Texte umbenennen Es ist mit dieser Art von Algorithmen möglich Ausreißer in einer vollständigen und abgeschlossenen Zeitreihe zu identifizieren. Es werden zwei Algorithmen vorgestellt, ein auf Percolation basierender Algorithmus und ein auf IsoMap basierender Algorithmus. Beide Algorithmen wurden dazu entwickelt Ausreißer in unterschiedlichen Typen von Datensätzen zu erkennen (z.B. Videos, Bilder, Netzwerke). Voraussetzung hierfür ist lediglich, dass eine Distanz zwischen unterschiedlichen Elementen des Datensatzes berechnet werden kann (vgl. Amil et al. 2019, S. 2). Im Folgenden werden die Algorithmen, hinsichtlich ihrer Fähigkeit Ausreißer in Zeitreihen zu identifizieren evaluiert.

Für beide Algorithmen gilt, dass die Zeitreihe zunächst in ein Netzwerk umgewandelt werden muss. Hierzu die Formel, welche hierzu verwendet wird:

$$a = b \tag{5.1}$$

$$D_{ij} = \left(\sum_k \left| v_k^i - v_k^j \right|^p \right)^{1/p}$$

Formel 1 gibt an, wie ein Element ij der Distanzmatrix berechnet wird. Insofern für p gleich zwei eingesetzt wird, wird zwischen den Elementen die euklidische Distanz berechnet. Ein Element der Zeitreihe kann aus mehreren Werten bestehen z.B. bei multivariaten Zeitreihen. Die berechnete Distanzmatrix bildet ein Netzwerk, dabei bilden die Zeitreihen Elemente die Knoten und die Distanzen zwischen ihnen, die Gewichte (vgl. Amil et al. 2019, S. 2-3).
Beide Algorithmen liefern lediglich einen Ausreißer Score zurück. Um zu bestimmen inwiefern ein Element konkret ein Ausreißer ist, wird zunächst den Mittelwert und die Standardabweichung des Outlier Scores berechnet. Falls ein Element in Abhängigkeit von der Standardabweichung sehr stark vom Mittelwert abweicht wird das Element als Ausreißer klassifiziert.

5.1 IsoMap Basierter Algorithmus

Der Grundgedanke hinter diesem Ansatz ist, dass Informationen über Ausreißer bei der Reduzierung der Dimensionalität mit dem IsoMap Algorithmus verloren gehen. Insofern versucht wird, die Informationen zu rekonstruieren und mit der ursprünglichen Matrix vergleicht, können große Abweichungen bei Ausreißer Elementen festgestellt werden (vgl. Amil et al. 2019, S. 3).

5.1.1 IsoMap

Beim IsoMap handelt es sich um einen Algorithmus zur nichtlinearen Dimensionsreduktion. Zunächst werden beim IsoMap die Nachbarn eines jeden Punktes über K-Nearest Neighbor be-

stimmt. Anschließend wird jeder Punkt mit den gefundenen Nachbarn verknüpft, wodurch ein neuer Körper entsteht. Daraufhin wird eine neue Distanzmatrix auf dem entstandenen Körper berechnet. Diese Matrix kann auch als geodätische Distanzmatrix bezeichnet werden und wird im weiteren Verlauf des Algorithmus benötigt. Der Zweck des Ablaufs ist es das nichtlineare Zusammenhänge in der anschließenden Dimensionsreduktion erhalten bleiben. Die Dimensionsreduktion erfolgt anschließend über Eigenvektor ? (vgl. Tenenbaum et al. 2000, S. 3). todo: Noch nach Seite für Quelle suchen

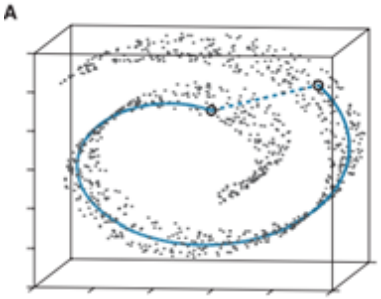


Abb. 5.1: Funktionsweiße IsoMap

5.1.2 IsoMap Basierter Algorithmus

Mithilfe des IsoMap Algorithmus wurden neue Features berechnet. Im nächsten Schritt wird versucht aus diesen Features die ursprüngliche Distanzmatrix zu rekonstruieren. Nun kann die Distanzmatrix aus Kap. 5.1.1 mit dieser Distanzmatrix verglichen werden. Dazu wird die Pearson Korrelation zwischen den jeweiligen Vektoren der Matrizen berechnet. Für Ausreißer wird erwartet, dass die Ähnlichkeit sehr niedrig ist, da die Informationen über sie bei der Reduktion verloren gehen (vgl. Amil et al. 2019, S. 3).

5.1.3 Implementierung

Da für die Implementierung des Algorithmus viele Berechnungen mit Matrizen durchgeführt werden müssen, wurde hierzu auf Python/Numpy zurückgegriffen. Für den IsoMap Algorithmus existierte eine sehr gute Implementierung in SciKitLearn, deshalb wurde auf diese zurückgegriffen. An den Algorithmus können verschieden Parameter übergeben werden, es handelt sich hierbei um dieselben Parameter, welche auch an den IsoMap Algorithmus übergeben werden können.

5.1.4 Ergebnisse

Ausreißer Typ	**1D**
Einzelne Peaks	*
Zunahme an Rauschen	*
Signal Drift	*
Kontinuierliche Zunahme der Amplitude	*
Zyklus mit höherer Amplitude	*
Zyklus mit geringerer Amplitude	*
Zyklus-Aussetzer	*
Signal-Aussetzer	*
Frequenzänderung	*

todo: Die richtigen Ergebnisse rein machen und bisschen was drüber schreiben

5.2 Perculation

Bei diesem Algorithmus werden schrittweiße die Kanten mit den höchsten Gewichten aus der Distanzmatrix entfernt. Ziel dieses Prozesses ist es Ausreißer vom Hauptcluster zu trennen. Dabei kann davon ausgegangen werden, dass Ausreißer höhere Kantengewichte zu ihren Nachbarn aufweisen und deshalb schneller separiert werden. Sobald ein Knoten komplett separiert ist, wird ihm ein Ausreißer Score zugeordnet. Der Wert des Ausreißer Scores wird über die zuletzt entfernte Kante des Knoten definiert (vgl. Amil et al. 2019, S. 3).

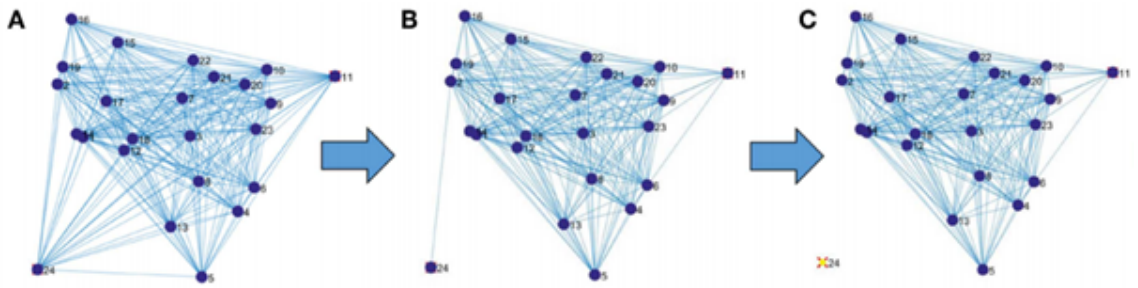


Abb. 5.2: Ablauf Perculation basierter Algorithmus

5.2.1 Implementierung

Aus denselben Gründen wie in Kap. 5.1.3 erläutert, wurde für die Implementierung auf Python/Numpy zurückgegriffen. Da die Distanzmatrix sehr umfangreich werden kann wurden einige Veränderungen an dem Algorithmus vorgenommen, um ihn Performanter zu machen. Eine Modifikation, die vorgenommen wurde, ist das die Kanten nicht einzeln, sondern in Gruppen entfernt werden. Dadurch muss seltener überprüft werden, ob ein Knoten mittlerweile komplett isoliert ist. Außerdem wurde ein Abbruchkriterium implementiert, bei welchem der Algorithmus angehalten wird sobald eine bestimmte Prozentzahl an Kanten entfernt wurde. Dies hat keine Auswirkungen auf die Qualität der Ausreißer Erkennung, da Ausreiser üblicherweise bereits zu beginn des Algorithmus isoliert werden. Der Algorithmus berechnet für jedes Element der Zeitreihe einen Ausreißer Score. Allerdings können die Ausreißer Scores sehr stark schwanken. Deshalb ist es schwierig Ausreißer zu identifizieren, welche sich über mehrere Zeitschritte erstrecken, da kein kontinuierliches Ansteigen des Scores beobachtet werden kann. Eine Möglichkeit, um diese Art der Ausreißer trotzdem zu identifizieren, ist es ein Sliding Window Verfahren einzusetzen. Dabei wird der Ausreißer Score für jedes Element neu berechnet, indem ein Mittelwert über die Zeitpunkte vor einem und nach einem Element gebildet wird. Dadurch werden die Schwankungen im Ausreißer Score abgemildert. Prinzipiell ist der Algorithmus parameterfrei, durch die Veränderungen kann jedoch die Größe des Sliding Window als Parameter übergeben werden.

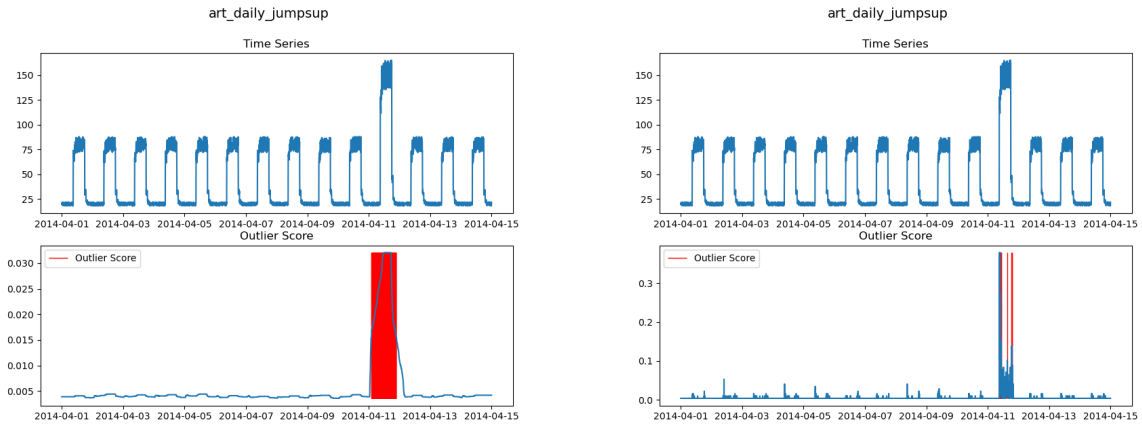


Abb. 5.3: Vergelich Perculation Algorithmus mit Sliding Window Verfahren und ohne Sliding Window Verfahren

5.2.2 Ergebnisse

Ausreißer Typ	**Datei Name**	**1D**
Einzelne Peaks	anomaly-art-daily-peaks	*
Zunahme an Rauschen	anomaly-art-daily-increase-noise	****
Signal Drift	anomaly-art-daily-drift	***
Kontinuierliche Zunahme der Amplitude	art-daily-amp-rise	***
Zyklus mit höherer Amplitude	art-daily-jumpsup	****
Zyklus mit geringerer Amplitude	art-daily-jumpsdown	****
Zyklus-Aussetzer	art-daily-flatmiddle	****
Signal-Aussetzer	art-daily-nojump	-
Frequenzänderung	anomaly-art-daily-sequence-change	-

Die Qualität der Ausreißer Erkennung mit dem Perculation Algorithmus kann großenteils als gut bis sehr gut bezeichnet werden. Lediglich die Ausreißer Typen Einzelne Peaks, Signal Aussetzer und Frequenzänderung können vom Algorithmus nicht erkannt werden. Bei den einzelnen Peaks liegt das an der Verwendung des Sliding Window Verfahren, dadurch werden die Ausschläge im Ausreißer Score weggemittelt und können nur noch sehr schlecht identifiziert werden. Wird jedoch kein Sliding Window Verfahren angewandt können die Ausreißer sehr gut identifiziert werden. Signal Aussetzer und Frequenzänderungen können vom Perculation Algorithmus nicht identifiziert werden, weil die Werte der Zeitreihe hierbei nicht von den Werten der restlichen Zeitreihe abweichen.

todo: Die richtigen Ergebnisse rein machen und bisschen was drüber schreiben

todo: Die Bilder vielleicht noch überarbeiten, sodass sie schöner aussehen. Vielleicht auch noch die Tabelle mit den Sternen rein machen. Vielleicht die beiden Graphiken zu einer Zusammenführen.

todo: Fragestellung: Inwieweit können vielleicht auch andere Datensätze in Graphen umgewandelt werden, sodass z.B. der Netismile darauf angewendet werden kann.

6 Kapitelname

6.1 Unterkapitel

6.1.1 Unterunterkapitel

Man kann mit den labels im Text bezug nehmen mit Kap. 6.1.1. Wenn man zitieren möchte, dann verwendet man am besten (vgl. Mihajlovic & Petkovic 2001, S. 1). Ich denke die Logik wird klar, wenn man es hier betrachtet. Anführungszeichen *im Fließtext* erfolgen mittels "Wort" manchmal benötigt man das space um ein Leerzeichen zu generieren.

Fette Überschrift

Für Bilder könnt ihr folgendes copy pasten. Die Bilder, die ihr einfügen möchtet, sollten im Ordner fig eingefügt werden. Hier sollte bei includegraphics der Name eingesetzt werden, caption steht für Bildunterschrift und das label muss individuell sein damit man darauf referenzieren kann.

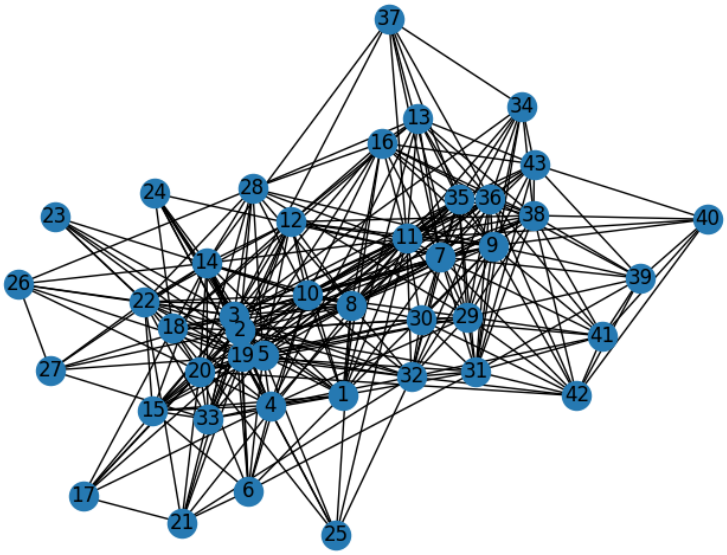


Abb. 6.1: Ego-Netzwerk des Datensatzes

Bei mehreren Bildern nebeneinander könnt ihr folgendes verwenden:

Wenn ihr to dos vermerken wollt, nutzt hierfür: todo: hier ein to do einfügen

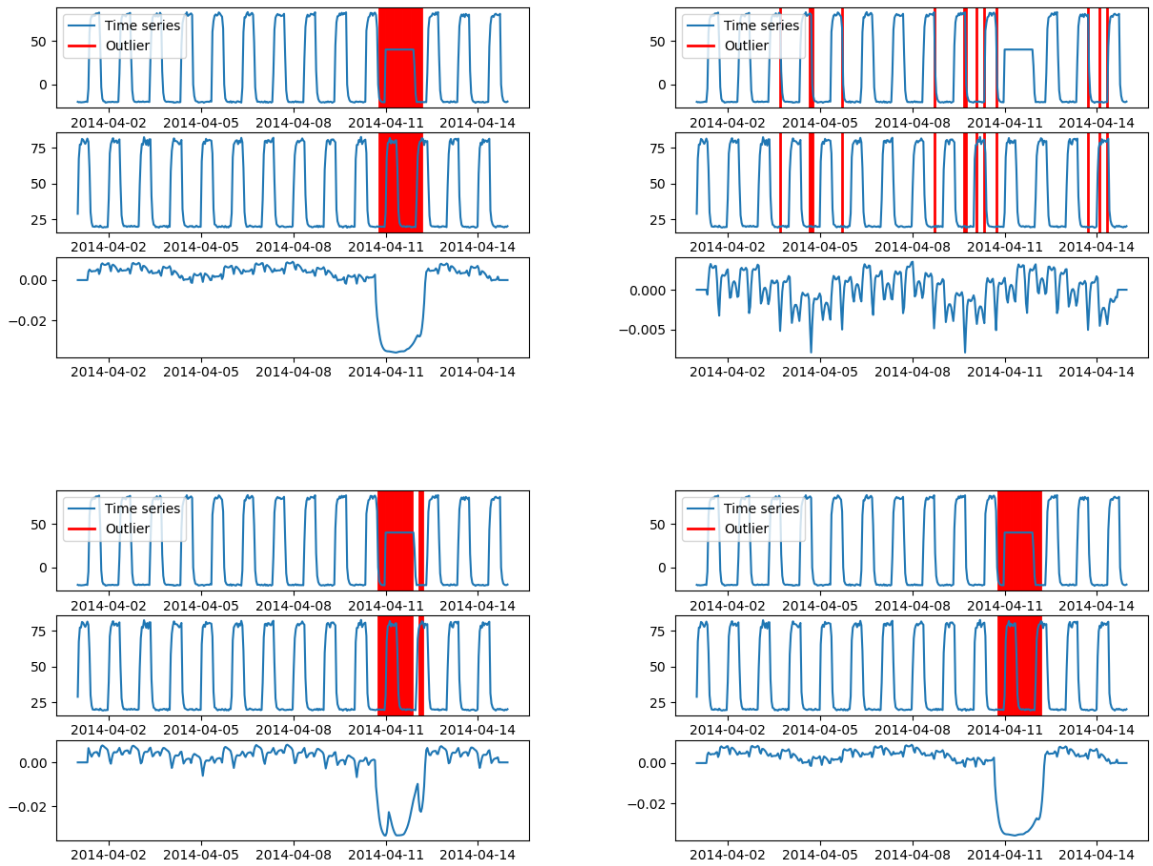


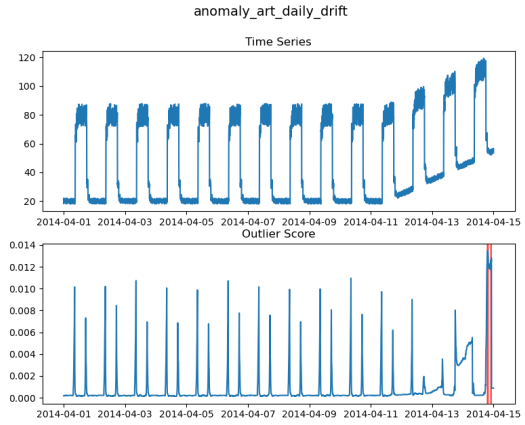
Abb. 6.2: Im Uhrzeigersinn: an erster Variablen ausgerichtet (Variante 1), an zweiter Variablen ausgerichtet (Variante 2), nicht ausgerichtet (Variante 3) und an allen Variablen ausgerichtet (Variante 4).

Wenn ihr etwas aufzählen wollt:

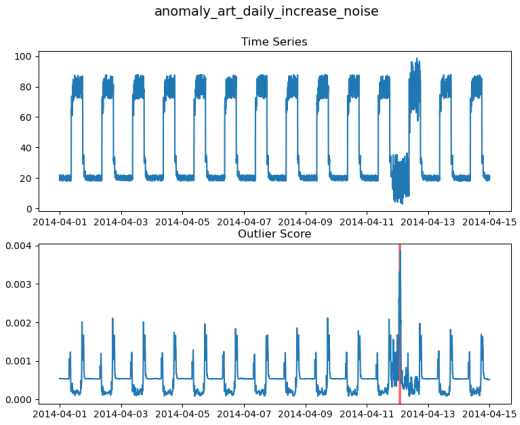
1. Verschaffen eines Überblicks über die existierenden Algorithmen zur Erkennung von Ausreißern in Graphen
2. Die Entwicklung eines Ausreißer-Scores für die zugrundeliegenden Algorithmen
3. Erste Anwendung der verwendeten Graphen-basierten Algorithmen auf Zeitreihen

A Isomap

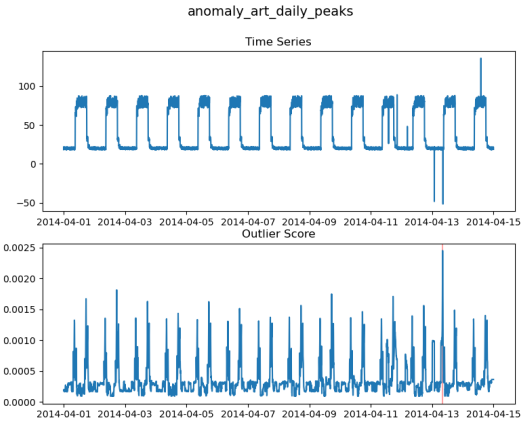
A.1 Eindimensionales Signal



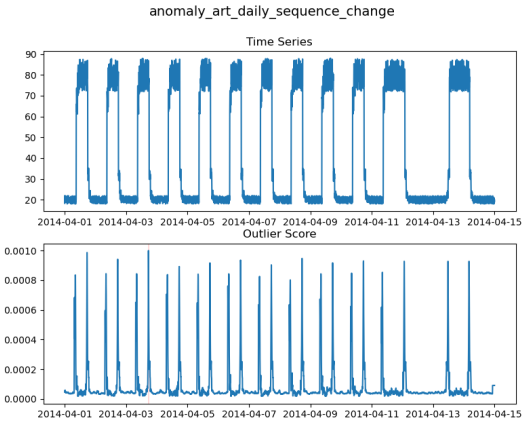
(a) Caption for sub-figure1



(b) Caption for sub-figure1

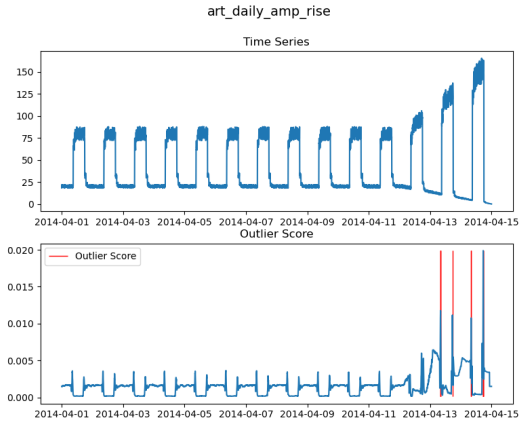


(c) Caption for sub-figure1

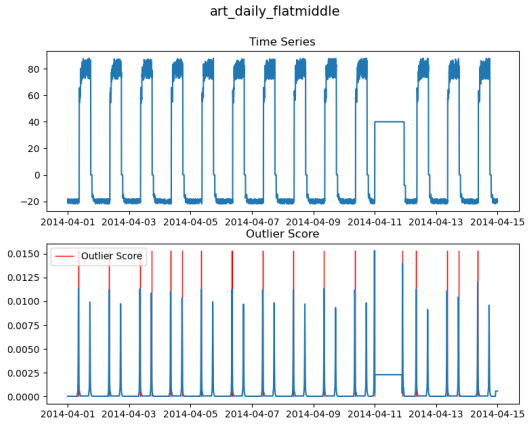


(d) Caption for sub-figure1

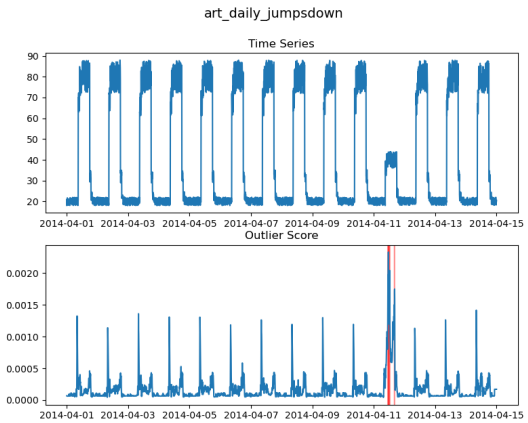
todo: In einigen Bilden fehlt die Legende. Vielleicht noch ein Paar bessere Ergebnisse zu erziehlen



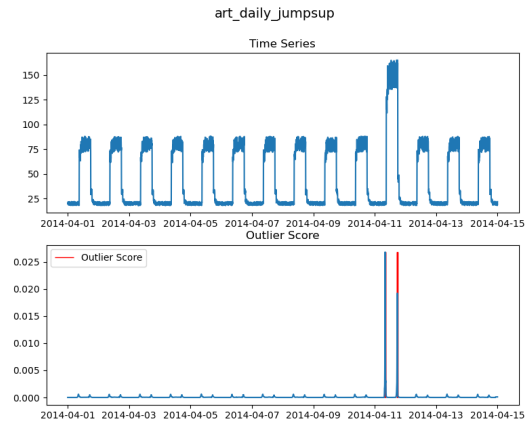
(e) Caption for sub-figure1



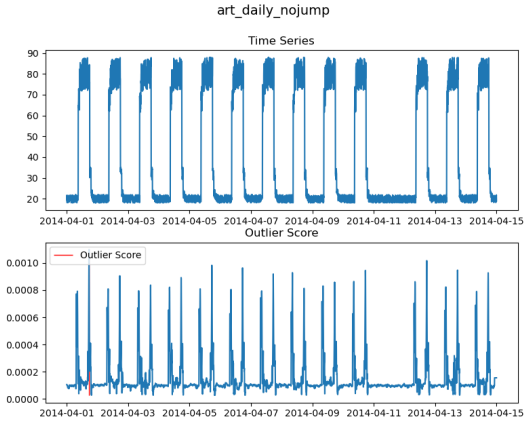
(f) Caption for sub-figure1



(g) Caption for sub-figure1



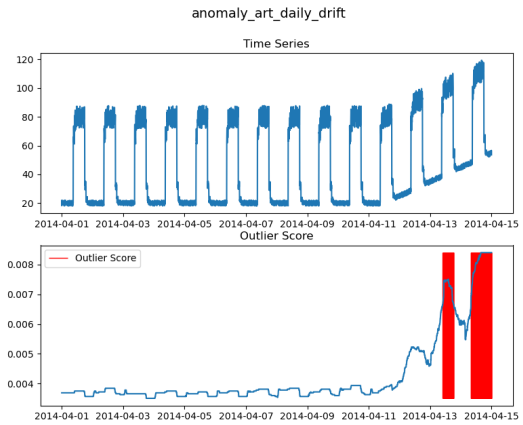
(h) Caption for sub-figure1



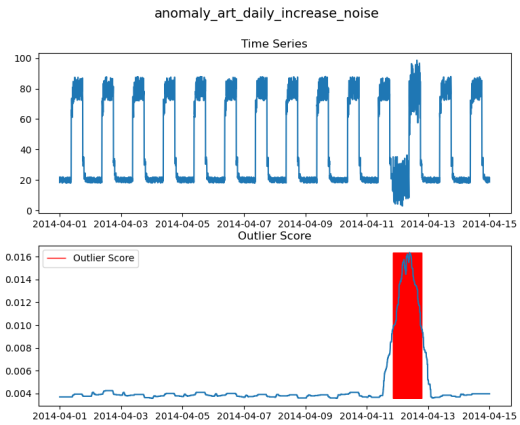
(i) Caption for sub-figure1

B Perculation

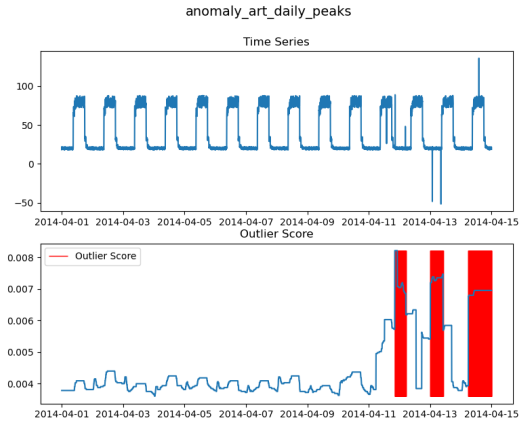
B.1 Sliding Window



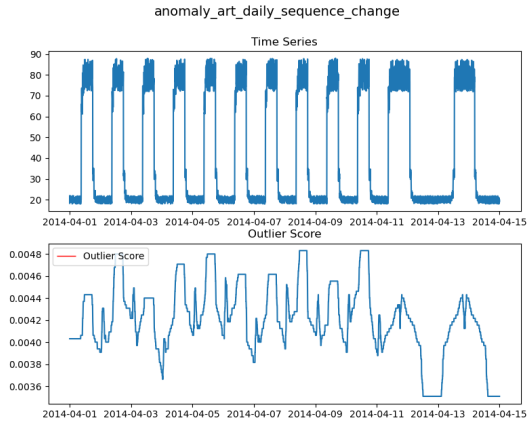
(a) Caption for sub-figure1



(b) Caption for sub-figure1

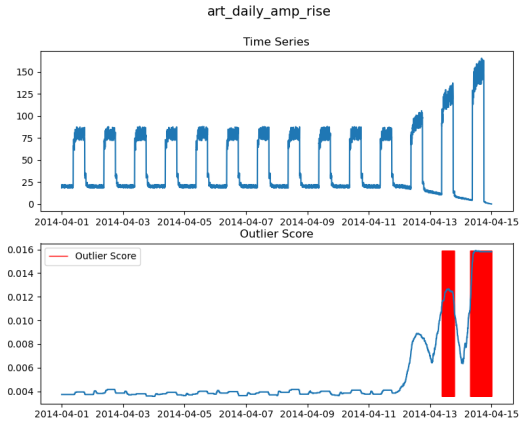


(c) Caption for sub-figure1

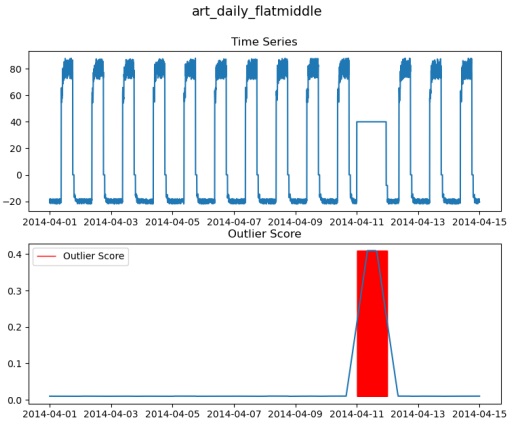


(d) Caption for sub-figure1

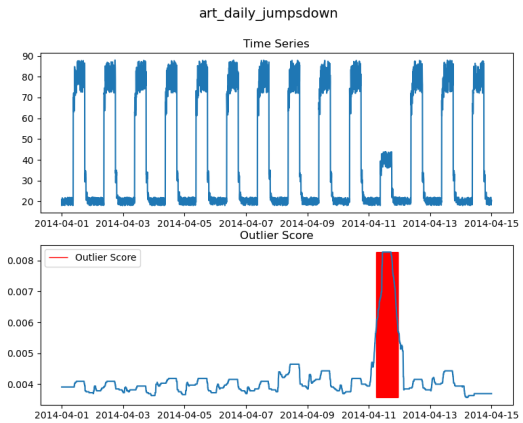
todo: Nim mir nicht sicher ob ich das ohne sliding window auch noch einfÃ¼gen soll. Vielleicht kann ich oben ja einmal einen Vergleich mit sliding window und ohne sliding window rein machen



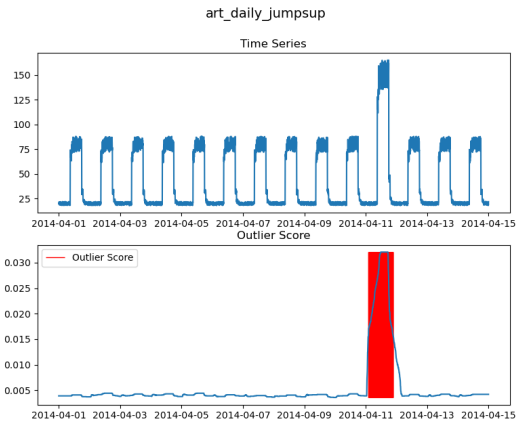
(e) Caption for sub-figure1



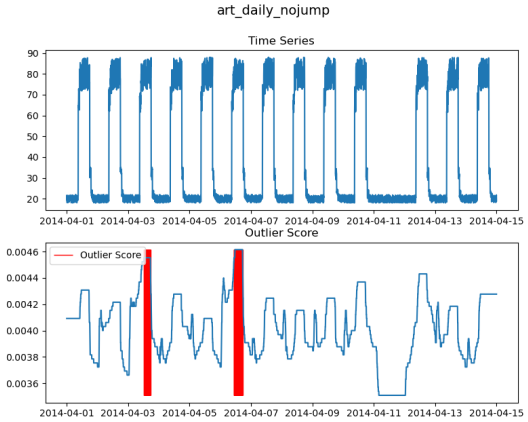
(f) Caption for sub-figure1



(g) Caption for sub-figure1



(h) Caption for sub-figure1

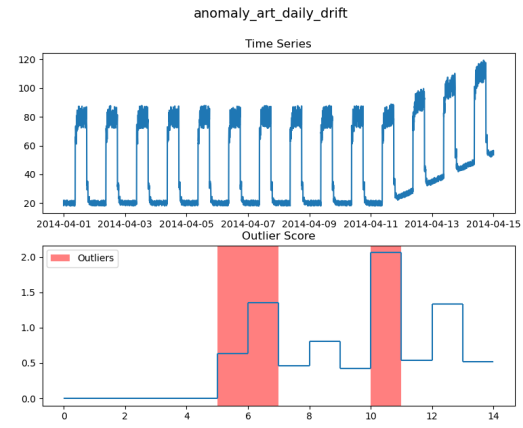


(i) Caption for sub-figure1

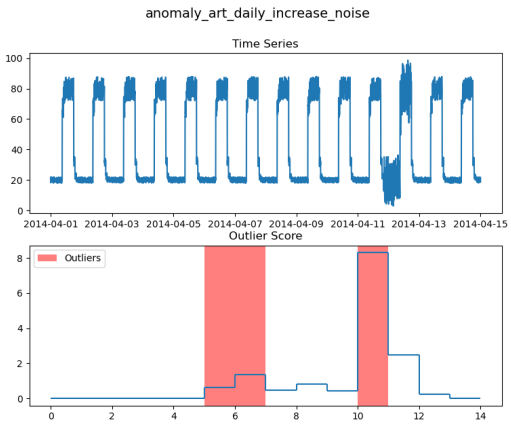
C Netsimile

C.1 Eindimensionales Signal

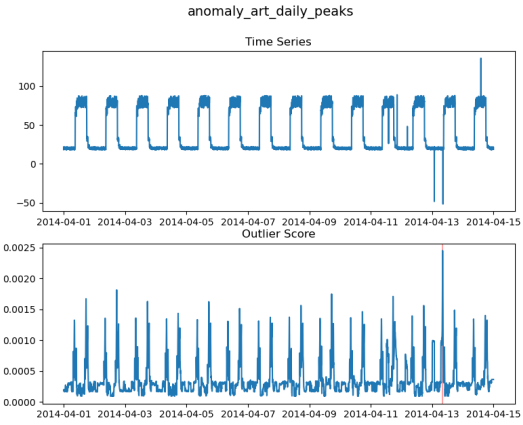
todo: Wrong picture for daily peaks. Change that the sixed element is not always an outlier



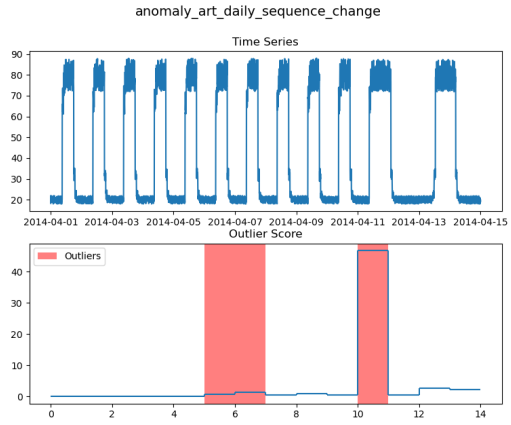
(a) Caption for sub-figure1



(b) Caption for sub-figure1



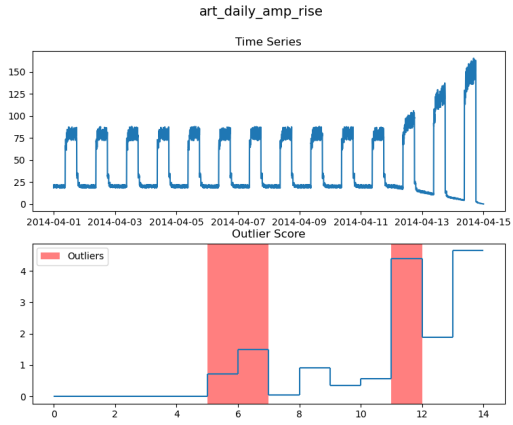
(c) Caption for sub-figure1



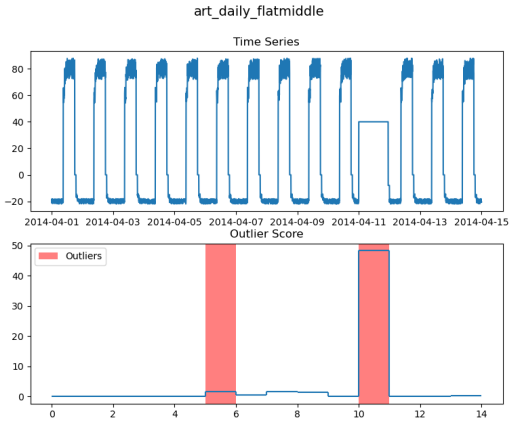
(d) Caption for sub-figure1

C.2 Zweidimensionales Signal

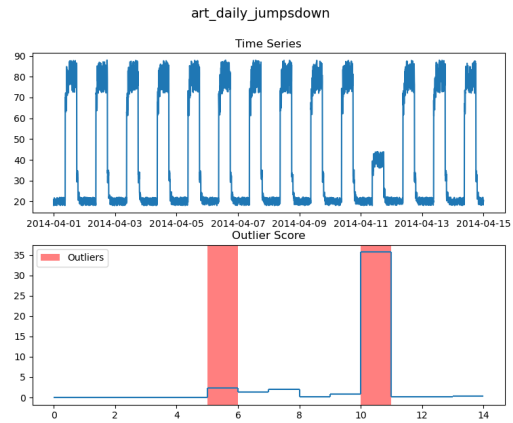
todo: Wrong picture for daily peaks. Change that the sixed element is not always an outlier



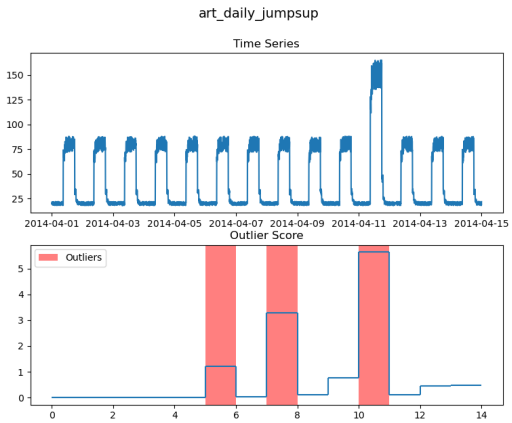
(e) Caption for sub-figure1



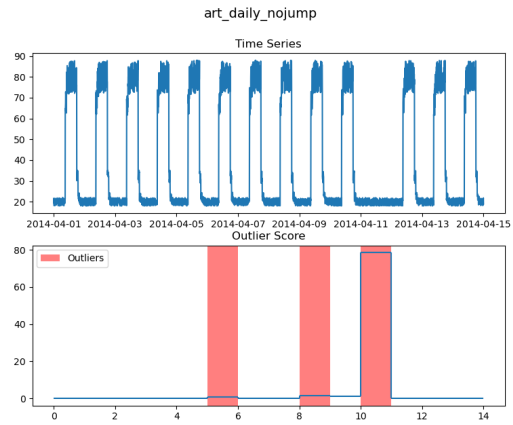
(f) Caption for sub-figure1



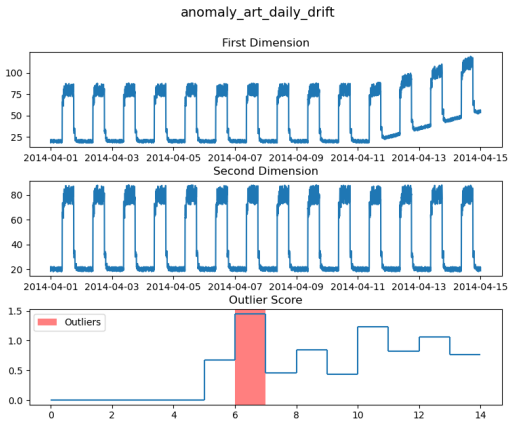
(g) Caption for sub-figure1



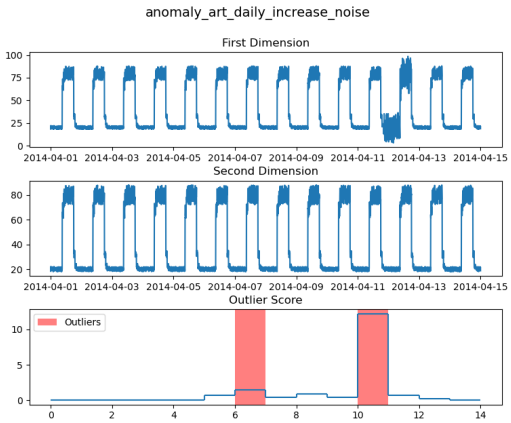
(h) Caption for sub-figure1



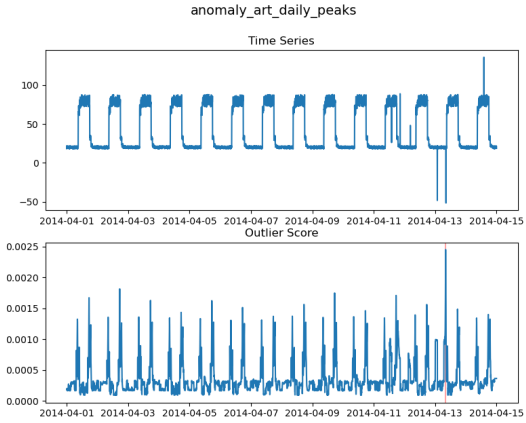
(i) Caption for sub-figure1



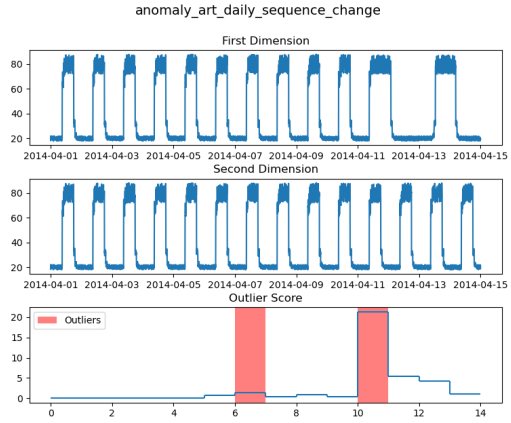
(a) Caption for sub-figure1



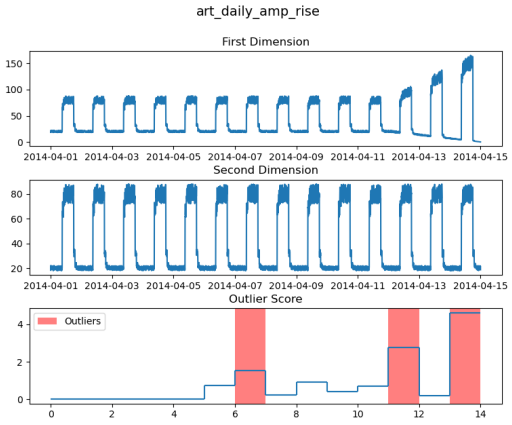
(b) Caption for sub-figure1



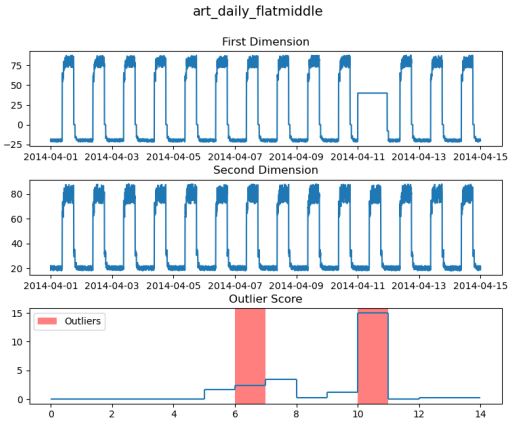
(c) Caption for sub-figure1



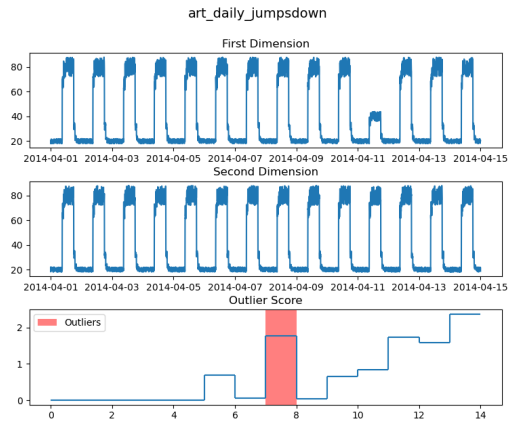
(d) Caption for sub-figure1



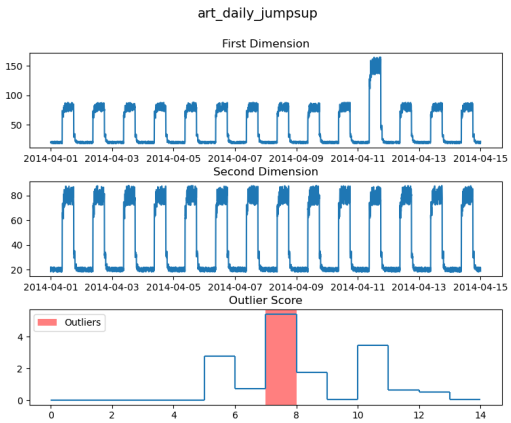
(e) Caption for sub-figure1



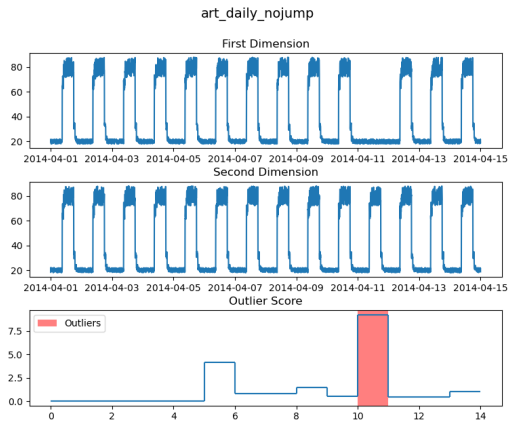
(f) Caption for sub-figure1



(g) Caption for sub-figure1



(h) Caption for sub-figure1



(i) Caption for sub-figure1

Literaturverzeichnis

Amil, P., Almeira, N. & Masoller, C. (2019), 'Outlier mining methods based on graph structure analysis', *Frontiers in Physics* **7**, 194.
URL: *https://www.frontiersin.org/article/10.3389/fphy.2019.00194*

Mihajlovic, V. & Petkovic, M. (2001), *BEISPIELDynamic Bayesian Networks: A State of the Art*, Vol. TR-CTIT-34 of *CTIT Technical Report Series*, University of Twente, Netherlands. Imported from CTIT.

Tenenbaum, J. B., Silva, V. d. & Langford, J. C. (2000), 'A global geometric framework for nonlinear dimensionality reduction', *Science* **290**(5500), 2319–2323.
URL: *https://science.sciencemag.org/content/290/5500/2319*