“Feinstaub in Graz”

Hausübung 2

7. November 2018

Originaldaten: https://luftdaten.info/

1 Tasks (20pt)

Laden Sie die Angabedaten[1]auf Ihr System. Das ZIP Archiv besteht aus zwei Dateien. Erstens dem Datensatz (`sensors_graz.csv`) und zweitens der Datei `ass_2_plot.py` die Sie zum Testen Ihres Codes verwenden können. Achten Sie darauf, dass Sie diese beiden Dateien bei Ihrer Abgabe nicht mit hochladen!

1.1 Command Line Parameters (3p)

Ihr Programm soll mit einer Reihe von Kommandozeilenparametern umgehen können. Verwenden Sie das `argparse` package[2] package im diese Funktionialität zu implementieren. Implementieren Sie die folgenden Parameter:

`-i --input:` Pfad zur Inputdatei (dem Datensatz)

- Datentyp: String
- Optional: Nein

`-o --output:` Pfad zur Outputdatei (dem bereinigten Datensatz)

- Datentyp: String
- Optional: Nein

`-s --sensors:` Namen der gewünschten Sensoren

- Datentyp: Liste aus Integern
- Optional: Ja
- Defaultwert: `[ ]` (leere Liste)

`-r --resampling_delta:` Größe der Zeitschritte beim Resampling (in Sekunden)

- Datentyp: Integer
- Optional: Ja
- Defaultwert: `600` (10 Minuten)

`-w --window_smoothing:` Große des Zeitfenstern beim Glätten

- Datentyp: Integer
- Optional: Ja
- Defaultwert: `None`

[1] https://github.com/pkasper/info1-bm/raw/master/2018/assignments/assignment_2_files.zip
[2] https://docs.python.org/3.7/library/argparse.html

1.2 Basic I/O Checks (2p)

Überprüfen Sie die dateibezogenen Parameter auf Ihre Gültigkeit. Implementieren Sie die folgenden Checks:

(i) Verifizieren, dass die Inputdatei existiert und auch tatsächlich eine Datei ist.
(ii) Verfizieren Sie, dass der gewünschte Output Pfad existiert.
(iii) Verifizieren Sie, dass die Outputdatei die Endung `.csv` hat.

Sollte einer der Tests fehlschlagen geben Sie eine Fehlermeldung aus und beenden Sie das Programm mit `exit()`. Alternativ können Sie auch eine Exception[3] raisen.

1.3 Datensatz Lesen & Vorbereiten(6p)

1.3.1 Lesen

Der Datensatz besteht aus einer Vorbereinigten CSV Datei. Dies bedeutet, dass eventuelle Fehler bereits für Sie entfernt wurden. Das Trennzeichen der Datei ist der internationale Standard, das einfache Komma (`,`). Bei CSV Dateien handelt es sich um simpel kodierte Tabellen (wie Sie es vielleicht aus Excel oder ähnlichen Programmen kennen). In der ersten Zeile finden Sie die Labels der jeweiligen Spalten. Das bedeutet, Sie müssen diese getrennt vom Rest lesen.

Im Datensatz finden Sie die folgenden Head (Spalten):

(i) **`sensor_id` (integer)** Identifier des Sensors.
(ii) **`sensor_type` (string)** Typ des Sensors. Immer SDS011
(iii) **`location` (integer)** Location Identifier des Sensors.
(iv) **`lat` (float)** Breitengrad Koordinate
(v) **`lon` (float)** Längengrad Koordinate.
(vi) **`timestamp` (numpy.datetime64)** Timestamp des Eintrags
(vii) **`P1` (float)** Erster Sensorwert (PM_{10})
(viii) **`P2` (float)** Zweiter Sensorwert ($PM_{2.5}$)

Jede Zeile (nach den Headern) enthält die gemessenen Werte eines Sensors zu einem gewissen Zeitpunkt. Arbeiten Sie Zeile für Zeile ab und speichern Sie die Werte. Der Kommandozeilenparameter `--sensors` dient hier als Filter. Überspringen Sie Zeilen die Sensoren (IDs) beschreiben die nicht explizit gewünscht sind. Sollte der Parameter nicht angegenen sein (also den Defaultwert `[ ]` haben, dann speichern Sie alle Sensoren.

Interne Datenstruktur. Für diese Übung ist keine fixe interne Datenstruktur vorgegeben. Sie können aber folgendes Template verwenden:

```
sensors = dict()

sensors[sensor_id] = dict()
sensors[sensor_id]['sensor_id'] =  #  ID des Sensors
sensors[sensor_id]['lat'] =  # lat (latitude) Koordinate
sensors[sensor_id]['lon'] =  # lat (longitude) Koordinate
sensors[sensor_id]['sensor_type'] =  #  Typ des Sensors (sollte immer SDS011 sein)
sensors[sensor_id]['data'] = dict()  #  Dictionary für Messwerte
sensors[sensor_id]['data']['timestamp'] = []  #  Timestamps
sensors[sensor_id]['data']['P1'] = []  #  Werte in P1
sensors[sensor_id]['data']['P2'] = []  #  Werte in P2
```

Überprüfen Sie hier stets ob der Sensor Ihrer aktuellen Zeile bereits im Dictionary (hier `sensors`) enthalten ist. Ansonsten müssen Sie den Eintrag frisch anlegen.

[3] https://docs.python.org/3/tutorial/errors.html

1.3.2 Prüfen und Aufbereiten

Wenn Sie den Datensatz erfolgreich eingelesen haben, führen Sie einen kurzen Check durch. Dieser soll überprüfen, dass für jeden Sensor exakt gleich viele Werte für `timestamp`, `P1`, und `P2` gelesen wurden. Sollte das nicht der Fall sein geben Sie eine entsprechende Fehlermeldung aus beenden Sie das Programm.

Als zweiten Vorbereitungsschritt müssen Sie verifizieren, dass die Reihenfolge der Messwerte von jedem Sensor chronologisch korrekt ist. Der `numpy.datetime64` Datentyp kann wie jeder Integer normal sortiert werden. Was in diesem Fall aber benötigt ist, ist die Reihenfolge der Indizes und nicht der eigentlichen Werte (Da man P1 und P2 in der gleichen Reihenfolge sortieren muss). Sie können folgenden Trick verwenden um diese Indizes zu erhalten:

1. Erstellen Sie eine Liste aus aufsteigenden Zahlen (z.B., via `range()`) mit der gleich vielen Einträgen wie Sie `timestamp` Einträge haben.
2. Sortieren Sie nun diese Liste (via `sort()` oder `sorted()`). In der Sortierfunkion geben Sie aber die `__getitem__` Funktion der `timestamp` Liste an. Dann erhalten Sie Ihre Zahlenliste sortiert anhand der Timestamps. Dies wieder entspricht den Indizes.
3. Reordern Sie `timestamp`, `P1`, `P2` anhand der Indizes. Dies lässt sich sehr elegant als List Comprehension lösen.

Beispiel Indizes Erstellen:

```
sort_order = sorted(<indices_list>, key=<timestamp_list>.__getitem__)
```

Hinweis: Der Trick zum Indizes erstellen entspricht der Funktionalität von `numpy.argsort()`[4]. Sie können diese Funktion zum Überprüfen verwenden.

1.4 Bereinigen & Smoothing (7p)

Erstellen Sie hier als erstes eine Kopie Ihrer Datenstruktur. Verwenden Sie hierfür `copy.deepcopy()`[5]. Arbeiten Sie nun mit der Kopie weiter. Dadurch können Sie am Ende einen Vorher/Nachher-Vergleich machen.

1.4.1 Outlier Entfernen

Durchlaufen Sie für jeden Sensor die Liste der Werte für `P1` und `P2`. Vergleichen Sie nun den aktuellen Wert mit seinem Vorgänger und Nachfolger. Wenn der aktuelle Wert höher ist als die Summe aus Vorhänger und Nachfolger, dann betrachten Sie ihn als Outlier. Berechnen Sie den Mittelwert aus Vorgänger und Nachfolger und schreiben Sie diesen an die aktuelle Stelle.

1.4.2 Resampling

Wenn Sie sich die Werte in `timestamp` anschauen, dann sehen Sie, dass die Zeitintervalle zwischen den Einzelnen Messwerten nicht konstant sind und auch nicht einheitlich über die unterschiedlichen Sensoren. Daher resamplen Sie alle Sensoren auf das gleiche Intervall Definiert durch den Kommandozeilenparameter `--resampling_delta`.

Berechnen Sie für jedes Zeitfenster den Median der Werte aus den Daten (jeweils für `P1` und `P2`). Beginnen Sie das Erstellen der Zeitfenster um 00:00 Uhr an jedem Tag, wo Sie den ersten Dateneintrag für den jeweiligen Sensor haben. Überschreiben Sie die bisherigen Daten mit Ihren neuen Timestamps und Werten. Sollte es in einem Zeitfenster keine Einträge gegeben haben (z.B., Sensor war offline), dann speichern Sie an diese Stelle `None` als Wert.

[4] https://docs.scipy.org/doc/numpy-1.15.4/reference/generated/numpy.argsort.html?highlight=argsort#numpy.argsort
[5] https://docs.python.org/3.7/library/copy.html

Hinweis: Sollte der Median zwischen zwei Werten liegen, dann nimmt man in der Regel den Mittelwert aus größerem und kleinerem Nachbar. Zum Beispiel: Der Median aus $[1, 2, 3, 4]$ ist 2.5.

Tipp: Lesen Sie die Dokumentation zu den numpy datetime und timedelta Datentypen[6]

1.4.3 Rolling Means

Um Trends und längerfristige Verhalten zu erkennen, berechnen Sie nun die Mittelwerte (Means) über ein laufendes Fenster individuell für jeden Sensor. Die Größe dieses Fensters ist durch den Parameter `--window-smoothing` definiert. Betrachten Sie für jeden Eintrag alle Nachbarn, die gleich oder weniger ($\leq$) als $window\_smoothing/2$ entfernt sind. Für jedes dieser Fenster Berechnen sie den Mean (Arithmetisches Mittel).

Beispiel Rolling Mean:

```
example_list = [2, 2, 4, 4, 6, 6, 8, 8]
window_size = 4 # half = 2

#1st window: [2, 2, 4, 4, 6] for index 2
#2nd window: [2, 4, 4, 6, 6] for index 3
#3rd window: [4, 4, 6, 6, 8] for index 4
#4th window: [4, 6, 6, 8, 8] for index 5

rolling_means = [3.6, 4.4, 5.6, 6.4]  #  we calculate the mean for each window
```

Für jene Einträge, wo sie kein volles Fenster bilden können (am Rand), berechnen Sie auch keinen Mittelwert.

Hinweis: Da Sie im letzten Schritt die Sensorwerte auf statische Intervalle umgewandelt haben können Sie hier einfach den Parameter durch das Intervall dividieren und erhalten dadurch die Anzahl der vorherigen und nachfolgenden Werte die Sie betrachten müssen.

1.5 Speichern & Testen (2p)

1.5.1 Speichern

Speichern Sie die bereinigten und geglätteten Daten in ein CSV. Dies soll die gleiche Struktur haben wie Ihre Inputdatei. Dateinamen und Pfad sind definiert durch den dazugehörigen Kommandozeilenparameter (`--output`).

1.5.2 Testen

Importieren Sie das File `ass_2_plot.py`. Achten Sie darauf, dass sich diese Datei auch im gleichen Verzeichnis befindet wie ihr Script. Der Import stellt Ihnen die Funktion `plot_data(title, data_orig=None, data_smooth=None, filename=None)` zur Verfügung. Für `data_orig` und `data_smooth` erwartet diese Funktion jeweils ein Dictionary mit den Keys `P1` und `P2` welche die Sensorwerte enthalten. `data_orig` übergeben Sie die Werte vor allen Säuberungsschritten (deshalb wurden die Werte davor kopiert). `data_orig` enthält die Werte nach allen Schritten. Wenn Sie den Parameter `filename` übergeben, dann speichert das Script ein Bild an dem gewünschten Pfad. Dieses können Sie mit dem Referenzoutput vergleichen.

Erstellen Sie ein solches Bild für jeden gewünschten Sensor (definiert durch den `--sensors` Kommandozeilenparameter). Der Dateiname soll dabei jeweils dem Namen der Outputdatei (`--output`) entsprechen und den Suffix `_<sensor_id>` haben. Als Dateiendung verwenden Sie `.png`

[6] https://docs.scipy.org/doc/numpy-1.15.0/reference/arrays.datetime.html

Output Beispiel:
Aufruf:

```
> assignment_2.py --input=sensors_graz.csv --output=assignment_2.csv --sensors 1503 1693
```

Erstellte Dateien:

- `assignment_2.csv`
- `assignment_2_1503.png`
- `assignment_2_1693.png`

Hinweis: Das Script erwartet, dass das Seaborn package bei Ihnen installiert ist. Dies wird bei anaconda mitgeliefert. Sollte es jedoch auf Ihrem System fehlen, können Sie dies via `pip install seaborn` beheben.

Tipp: Vergleichen Sie Ihre Bilder mit der Referenz im Wiki[7].

1.6 Bonus: Rolling Window Ränder(2p)

An den Rändern der Fenster beim Smoothing finden sich noch die Originalwerte der Sensoren. Überlegen Sie sich eine Methode, wie Sich auch hier die Werte glätten könnten (z.B., durch schrumpfende Fenster in die jeweilige Richtung).

Ändern Sie Ihre Implementierung der bisherigen Rolling Windows dementsprechend ab.

Wichtig: Erwähnen Sie im Kommentarheader, dass Sie diesen Bonustask versucht haben!

2 Beschränkungen

- Fügen Sie keine nutzlosen Komponenten Ihrer Abgabe hinzu
- Sofern möglich, verwenden Sie eingebaute Funktionen
- Importieren Sie nur erlaube Packages
- Verlangen Sie keine extra Kommandozeilenparameter
  - Sie dürfen extra Parameter verwenden. Ihr Programm muss aber auch ohne deren Verwendung funktionieren!

2.1 Erlaubte Packages

- `argparse`
- `copy`
- `os`
- `sys`
- `numpy`

- `ass_2_plot`

[7] https://palme.iicm.tugraz.at/wiki/Info1BM

3 Datei Header

All Ihre Quelldateien in Ihrer Abgabe müssen gleich zu Beginn einen Kommentar mit folgenden Informationen enthalten:

- Author: – Ihr Name
- MatNr: – Ihre Matrikelnummer
- Description: – Generelle Beschreibung der Datei
- Comments: – Kommentare, Erklärungen, usw

Bitte einfach den folgenden Code in Ihre Dateien kopieren und den Inhalt anpassen. Je nach PDF-Reader müssen Sie eventuell die Leerzeichen/Einrückungen per Hand anpassen. Bei jupyter Notebooks fügen Sie den Kommentarheader bitte in die erste Zelle ein.

Beispielheader:

```
#########################################################################
# Author:      [FIRST NAME] [LAST NAME]
# MatNr:       [MATR NR]
# Description: [SHORTDESCRIPTION]
# Comments:    [ANY RELEVANT COMMENTS.
#              CAN BE MULTILINE]
#########################################################################
```

4 Coding Standard

Für diese Lehrveranstaltung orientieren Sie sich am offiziellen PEP 8 Standard[8]. Dieser Beschreibt grundsätzliche Formalitäten im Bezug auf Ihren Code. Folgendes ist besonders zu Beachten:

Sprache. Code schreibt man in Englisch. Im internationalen Zeitalter ist es notwendig, dass auch jemand am anderen Ende der Welt verstehen kann, was Sie programmiert haben. Ihr gesamter Quellcode muss daher auf Englisch geschrieben sein. Dies betrifft sowohl die Kommentare also auch Variablennamen und Ähnliches.

Leerzeichen statt Tabulatoren. Python basiert auf Einrückungen, anstatt auf geschwungenen Klammern . Theoretisch gibt es die Möglichkeit, Leerzeichen (spaces) oder Tabulatoren (tabs) zu verwenden. PEP8 schreibt aber klar 4 Leerzeichen als Einrückungen vor. Die meisten Python Programmierumgebungen werden automatisch 4 Leerzeichen einfügen, wenn Sie auf die Tabulator-Taste drücken.

Sprechende Namen. Verwenden Sie kurze, aber sprechende Namen für Ihre Variablen, Funktionen, (und Ähnliches). Es muss eindeutig aus dem Namen hervorgehen, was die Aufgabe des Elements ist. Für simple Iterationen kann ein einfaches i ausreichend sein, aber dies kann schnell zu Chaos führen. Sollten Sie Variablen haben, die keine Aufgabe haben und nicht verwendet werden, schreibt PEP 8 vor, einen einfachen Unterstrich (_) zu verwenden. Sollten Sie sich unsicher sein, dann beschreiben Sie ihre Variablen (und Namen) in einem Codekommentar.

120 **Zeichen Zeilenlänge.** PEP8 sieht Zeilenlängen von maximal 79 Zeichen vor. Ein anderes, etwas großzügigeres Limit, welches sich in der Szene etabliert hat, sieht eine Länge von 120 Zeichen vor. In dieser LV verwenden wir daher das erweiterte Limit. Bitte achten Sie darauf, dass keine Zeile in Ihrem Code diese Länge von 120 Zeichen überschreitet (gilt auch für Kommentare). *Hinweis:* Zeilenlänge inkludiert die Einrückungen mittels Leerzeichen!

[8] https://www.python.org/dev/peps/pep-0008/

5 Automatisierte Tests

Ihr Programm wird automatisiert getestet. Achten Sie daher, dass Sie die angabe genau einhalten. Dies gilt im Besonderen für vorgeschriebene Variablen- und Funktionsnamen!
Vergewissern Sie sich, dass Ihre Abgabe allen Beschränkungen, die in dieser Angabe erwähnt sind, konform ist. Zusätzlich wird jede Abgabe auch von einem Mitglied des Tutorenteams begutachtet. Ihre Tutoren führen auch die finale Bewertung (Puntke) durch.

6 Abgabe

6.1 Deadline

4. Dezember 2018 um 23:59:59.

Eine Spätabgabe ist nicht vorgesehen. Ausnahme bilden hier Notfälle. Sollte das Abgabesystem nicht online sein, verlängert sich die Deadline automatisch um 24 Stunden. Sollten Sie, aus diversen Gründen, nicht in der Lage sein, Ihre Abgabe hochzuladen, kontaktieren Sie Ihren Tutor *VOR* der Deadline. (*Hinweis:* Urlaub oder Ähnliches wird nicht als Grund akzeptiert!)

6.2 Hochladen der Abgaben

Assignments werden stets als Archive abgegeben. Erlaubt sind hier die Formate *.zip*, und *.tar.gz*.
Zusätzlich zu ihren Quelldateien, soll Ihre Abgabe auf eine Datei namens *readme.txt* beinhalten. Das Vorhandensein der Datei ist Pflicht, ihr Inhalt aber optional. Sie soll folgenden Inhalt haben: (i) Die Zeit, die Sie benötigt haben, um die Aufgabenstellung zu absolvieren. (ii) Feedback, wo Sie Probleme hatten. Ihr Feedback ermöglicht es, verbreitete Probleme zu erkennen und die Vorlesung und Tutoriumseinheiten entsprechend anzupassen.
Abgaben erfolgen auf der Palme Website. Bitte prüfen Sie vor der Abgabe diese Kriteren:

- Datei- und Ordnerstruktue (siehe unten)
- Kommentarheader in jeder Quelldatei
- Coding Standard

6.3 Struktur der Abgabe

```
assignment_2.zip (or assignment_2.tar.gz)
    assignment_2.py
    readme.txt
```

Die Angabedateien (Datensatz + `ass2_plot.py`) sollen NICHT mit abgeben werden.