

Universität Stuttgart

INSTITUT FÜR ANGEWANDTE ANALYSIS UND NUMERISCHE
SIMULATION

Metrik für sEMG-Daten

metrik__speeds.py

Lisa Dollmann, Maximilian Hack, Lilian Cathérine Lepère

Autor	Lisa Dollmann Maximilian Hack Lilian Cathérine Lepère
Betreuer	Aaron KRÄMER
Datum	Stuttgart 23. November 2021

Das ist die Dokumentation der Metrik *metrik_speeds.py*. Die Metrik misst wie häufig eine Geschwindigkeit in einem Datensatz von *data_creator_00x.py* vorkommt. Aufgrund der Struktur des *data_creator_001.py* ist die Metrik für diese Datensätze uninteressant. Für den *data_creator_002.py* veranschaulicht die Metrik, dass geringe Geschwindigkeiten deutlich häufiger auftreten. Die Metrik motiviert also das Erstellen neuer *data_creator_00x.py* mit einem anderen Verhältnis von geringen und hohen Geschwindigkeiten.

In der Dokumentation werden Installation und Nutzung beschrieben, die enthaltenen Features und weitere Ideen aufgelistet, die Ergebnisse kurz interpretiert, relevante Programmausgaben dargestellt und erläutert.

Inhaltsverzeichnis

1	Installation und Nutzung	4
1.1	Module und Elemente	4
1.2	Parameter	4
1.3	Nutzung	4
2	Enthaltene Features und deren Ausgaben	4
2.1	Dateiname aus 'Kennzahlen'	4
2.1.1	@ToDo: <i>str(sys.argv[4])</i>	5
2.1.2	@ToDo: <i>if</i> -Abfrage	5
2.2	Pickle	5
2.3	Dimension der Daten	5
2.4	Liste mit allen Geschwindigkeiten	6
2.5	Das Histogramm	6
2.5.1	@ToDo: Anzahl <i>bins</i>	6
2.5.2	@ToDo: Nicht auftretende Geschwindigkeiten	6
3	Interpretation	7
3.1	<i>data_creator_001</i>	7
3.2	<i>data_creator_002</i>	7
4	Relevante Programmausgaben	8
4.1	<i>data_creator_001</i>	8
4.1.1	Histogramm mit 3 Geschwindigkeiten und 20 Wellenlängen	8
4.1.2	Terminalausgabe bei 3 Geschwindigkeiten und 20 Wellenlängen . .	9
4.2	<i>data_creator_002</i>	10
4.2.1	Histogramm mit 3 Geschwindigkeiten und 20 Wellenlängen	10
4.2.2	Terminalausgabe bei 3 Geschwindigkeiten und 20 Wellenlängen .	11
4.2.3	Histogramm mit 30 Geschwindigkeiten und 30 Wellenlängen . . .	12
4.2.4	Terminalausgabe bei 30 Geschwindigkeiten und 30 Wellenlängen .	13

1 Installation und Nutzung

1.1 Module und Elemente

Das Programm benötigt die bereits an anderen Stellen des Projekts benötigten Module *pickle*, *numpy*, *matplotlib.pyplot* und *sys*.

1.2 Parameter

Es gibt die drei Eingabeparameter x des entsprechenden *data_creator_00x.py*, die *n_speeds* und *n_waveLengths*. Deren Nutzung wird im folgenden Abschnitt näher erklärt.

1.3 Nutzung

Beispielbilder zur Veranschaulichung der Nutzung sind in Kapitel 3 eingebunden.

Um die Metrik *metrik_speeds.py* zu nutzen hat man zwei Optionen. Man muss entweder den Dateinamen eines Datensatzes übergeben oder dessen 'Kennzahlen'. Will man beispielsweise die Datei *DATA/creator_001/RawData_3_velocities____15_waveLengths.pickle* mit der Metrik analysieren, übergibt man dem Terminal entweder diesen Namen, also *python metrik_speeds.py DATA/creator_001/RawData_3_velocities____15_waveLengths.pickle* oder *python metrik_speeds.py 1 3 15*. Letzteres ist eine abkürzende Schreibweise, die im folgenden kurz erläutert wird:

Erzeugt man mit *data_creator_00x.py* einen Datensatz, so ist der Dateiname immer nach dem selben Muster aufgebaut. Die Dateinamen sind vom Typ *DATA/creator_001/RawData_+str(n_speeds)+_velocities____+str(n_waveLengths)+_waveLengths.pickle* (bei *data_creator_002.py* kommt im Dateinamen noch der *grow_factor* hinzu; die Metrik nutzt aber immer den Default Wert). Um das Aufrufen eines Datensatzes aus einer *pickle*-Datei zu Erleichtern kann man also auch die 'Kennzahlen' eingeben. Das sind die Zahl x des entsprechenden *data_creator_00x.py*, die *n_speeds* und *n_waveLengths*. Ein Datensatz mit diesen 'Kennzahlen' muss bereits existieren!

Hat man das Programm ausgeführt zeigt es ein Histogramm, das die Häufigkeit aller im Datensatz vorkommenden Geschwindigkeiten darstellt.

Die Häufigkeit meint dabei in wie vielen Szenarien diese Geschwindigkeit vorkommt und nicht wie viele Wellen es mit dieser Geschwindigkeit im gesamten Datensatz gibt. Eine solche Metrik wäre ebenfalls interessant um die Ergebnisse mit 3.2 zu vergleichen.

2 Enthaltene Features und deren Ausgaben

2.1 Dateiname aus 'Kennzahlen'

Wie in 1.3 bereits angesprochen, werden Dateien entweder über deren Name oder über sogenannte Kennzahlen aufgerufen. Das funktioniert mithilfe des Moduls *sys*. Das Meiste zu diesem Vorgehen ist in 1.3 bereits erklärt. An dieser Stelle soll noch auf zwei Punkte

eingegangen werden. Das Programm soll anhand der eingegebenen *sys*-Argumente herausfinden, ob ein Dateiname oder nur die Kennzahlen der Datei übergeben wurden. Dazu wird mittels einer *if*-Abfrage überprüft wie lange der *string* des ersten *sys*-Arguments ist. Da im Moment mehr als 1000 *data_creator*s sehr nicht absehbar sind, sollte ein *string*, der mehr als 4 Zeichen beinhaltet ein Dateiname sein. Der zweite Punkt bezieht sich auf die Methode, die aus den Kennzahlen den Dateinamen zusammenfügt. Der Dateiname für Daten aus dem *data_creator_002.py* enthält den Zusatz *grow_factor__4*. Dies muss also im Fall $x=2$ geprüft und angepasst werden. Das wurde mithilfe einer *if*-Abfrage umgesetzt.

Daraus ergeben sich zwei @ToDos:

2.1.1 @ToDo: *str(sys.argv[4])*

Man kann ein viertes Argument *str(sys.argv[4])* übergeben um den *grow_factor* ebenfalls variabel zu halten. Eventuell kann man diesem *sys*-Argument schlicht den Default-Wert 4 zuordnen, damit lässt man dem Nutzer die Entscheidung.

2.1.2 @ToDo: *if*-Abfrage

Eventuell kann man die erste *if*-Abfrage, ob ein Dateiname oder nur dessen Kennzahlen übergeben wurden, kompakter fassen.

Man könnte beispielsweise nur den Dateinamen als Ganzes übergeben und die Option mit den Kennzahlen streichen. Wenn es mehr als die bisherigen beiden *data_creator_00x.py* gibt, können die Kennzahlen aufgrund möglicher Zusätze im Dateinamen, wie das beim *grow_factor* in *data_creator_002.py* der Fall ist, kompliziert werden. Man müsste entweder bei all diesen Zusätzen auf einen Default-Wert hoffen, oder 2.1.1 für all diese Zusätze umsetzen. Das ist entweder sehr steif oder ab spätestens 7 *sys*-Argumenten sehr verwirrend und Tippfehler-anfällig.

Bisher erleichtert die Kennzahlen-Schreibweise aber die Eingabe und bleibt vorerst erhalten.

2.2 Pickle

Nachdem der Dateiname aus den *sys*-Argumenten erstellt wurde, wird die Datei mithilfe des *pickle*-Moduls entpickelt. Das läuft standardmäßig ab.

2.3 Dimension der Daten

Je nachdem welcher *data_creator_00x.py* zum Erstellen eines Datensatzes genutzt wurde, sind die Daten aus der geladenen *.pickle*-Datei einfach (für $x=1$) oder mehrfach (für $x=2$) verschachtelt. Da die vorliegende Metrik nur die absolute Häufigkeit einer Geschwindigkeit in einem Datensatz misst, ist der Grad der Verschachtelung irrelevant. Man kann in dieser Metrik also alle Daten in das selbe Format umwandeln, um die Daten aus jedem *data_creator_00x.py* verarbeiten zu können.

Dafür werden die Daten zunächst in *numpy – arrays* umgewandelt. Eine *if*-Abfrage prüft dann über den Befehl *.ndim*, ob die Daten eindimensional sind. Falls nicht, werden sie über den Befehl *.flatten()* in ein eindimensionales *array* umgewandelt.

2.4 Liste mit allen Geschwindigkeiten

Um die Geschwindigkeiten später in einem Histogramm darzustellen, werden sie zunächst in einer Liste namens *speeds* gesammelt. Dafür werden sie in einer *for*-Schleife aus den Metadaten extrahiert. Über *meta_.get_vecs()[2][0]* greift man für jedese *scenario* auf die Geschwindigkeit(en) zu und fügt diese als Zahl vom Datentyp *float* der Liste *speeds* hinzu.

2.5 Das Histogramm

Das Histogramm bildet dann die Häufigkeit aller im Datensatz vorkommenden Geschwindigkeiten ab. Da die genaue Häufigkeit auf dem Bild schwer abzulesen ist, wird die Häufigkeit der jeweiligen Geschwindigkeit zusätzlich in Zahlenform in der Konsole ausgegeben.

Das Histogramm und die dazugehörige Ausgabe im Terminal lassen Verbesserungsmöglichkeiten offen.

2.5.1 @ToDo: Anzahl *bins*

Erzeugt man ein Histogramm, muss man die Anzahl und die Schrittweite zwischen den *bins* im Voraus festlegen. *Bins* sind die Balken im Histogramm. Legt man beispielsweise fest, dass es 5 *bins* mit der Schrittweite 2.5 geben soll, so erhält man die *bins* 0, 2.5, 5, 7.5 und 10. Tritt dann die Geschwindigkeit 3 auf, wird diese schlichtweg der 2.5 zugeordnet. Um ein genaues Bild von der Häufigkeit der Geschwindigkeiten zu bekommen, müsste man also die Anzahl und die Schrittweite an *bins* immer passend zum Datensatz wählen. Dafür nutzt man die Tatsache, dass die Geschwindigkeit maximal 10 wird. Außerdem legt man in jedem Datensatz *n_speeds*, also die Anzahl an verschiedenen Geschwindigkeiten in diesem Datensatz, fest. Zusammen ergibt sich dann $\frac{1}{n_speeds-1}$ als optimale Schrittweite und $(n_speeds - 1) \cdot 10 + 2$ als optimale Anzahl an *bins*. Dafür muss man *n_speeds* aus dem Dateinamen des Datensatzes auslesen. *n_speeds* steht an der Stelle 25 und, falls *n_speeds* echt größer ist als 9, an den Stellen 25 und 26 im Dateiname.

2.5.2 @ToDo: Nicht auftretende Geschwindigkeiten

Betrachtet man einen Datensatz mit vielen Gechwindigkeiten, werden sehr viele *bins* erzeugt. Da im Histogramm nur *bins* abgebildet werden, die häufiger als 0 mal vorkommen, ist das kein Problem. Allerdings wird die Ausgabe im Terminal sehr unübersichtlich. Die Idee ist, nur Geschwindigkeiten mit deren Häufigkeiten im Terminal auszugeben, wenn die Geschwindigkeiten häufiger als 0 mal vorkommen. Das könnte man mit einer *if*-

Abfrage machen.
ERLEDIGT.

3 Interpretation

3.1 *data_creator_001*

Die Metrik ist eher langweilig für die Daten, die mit *creator_01* erzeugt wurden. Das liegt daran, dass man die Anzahl an auftretenden Geschwindigkeiten übergibt und dass man mit der Formel $1 + \frac{10-i}{n_speeds-1} \cdot i$, für $i = 0, \dots, n_speeds-1$ berechnen kann, welche genau auftreten. Wie oft die Geschwindigkeiten auftreten ist gerade $n_waveLengths$.

3.2 *data_creator_002*

Für Daten aus dem *data_creator_02* wird deutlich ersichtlich, dass langsame Geschwindigkeiten deutlich häufiger vorkommen als schnelle (vgl. 3). Das an den ursprünglichen Arrays der einzelnen Geschwindigkeiten. Da langsame Wellen mehr 'Platz' im Array brauchen, da sie länger im Bild sind, müssen die Arrays zu den langsamen Geschwindigkeiten deutlich länger sein. Die schnellen Wellen sind kürzer im Bild und die Arrays entsprechend weniger lang. Der *data_creator_02* sucht sich zufällig Bereiche aus den Arrays und legt diese übereinander, so können Bilder mit mehreren Geschwindigkeiten entstehen. Da die Arrays der langsamen Geschwindigkeiten deutlich länger sind, kopiert der *data_creator_02* häufiger Bereiche davon heraus und nutzt diese für die neuen Daten. Um eine größere Datenvarianz zu erhalten könnte man also weitere *data_creator_0x* erstellen, die ein anderes Verhältnis zwischen schnellen und langsamen Wellen erzeugen.

4 Relevante Programmausgaben

In diesem Abschnitt werden im Text referenzierte Programmausgaben veranschaulicht.

4.1 *data_creator_001*

Die folgenden Bilder zeigen das Histogramm und die Ausgabe im Terminal bei Daten, die von *data_creator_001* erzeugt wurden. Dabei wird nur ein Beispiel in Augenschein genommen, da diese Bilder eher uninteressant sind (vgl. 3.1).

4.1.1 Histogramm mit 3 Geschwindigkeiten und 20 Wellenlängen

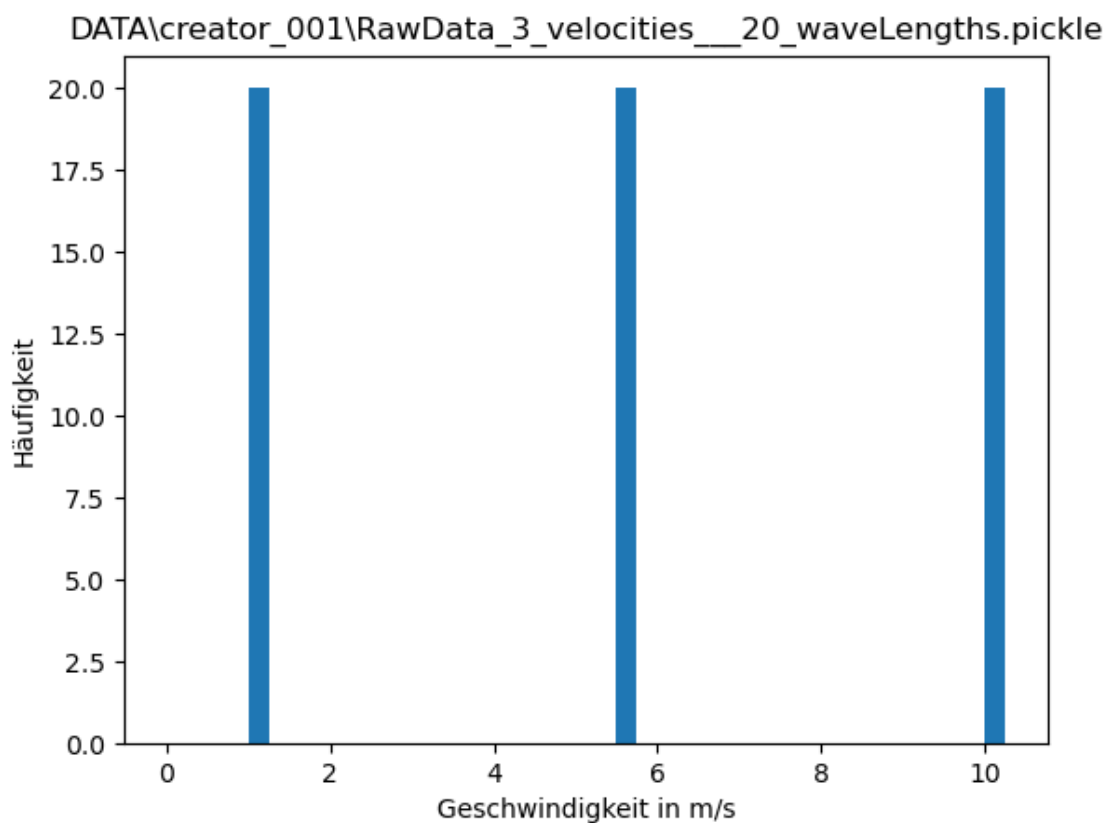
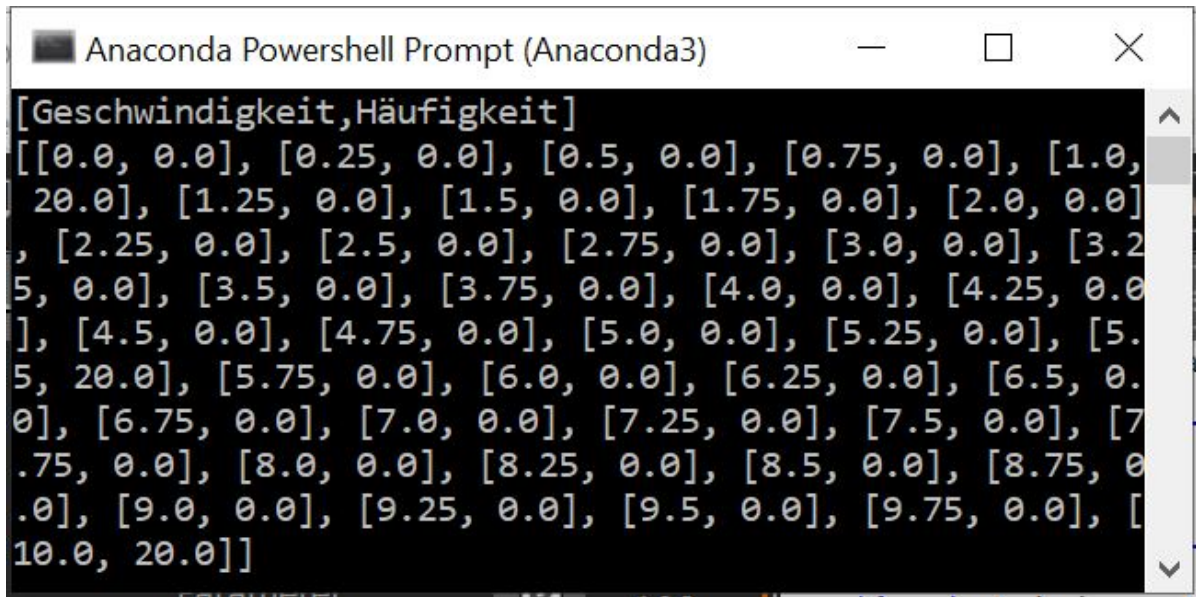


Abbildung 1: Das Histogramm eines Datensatzes, der mithilfe des *data_creator_001* erzeugt wurde. Der Datensatz enthält drei verschiedene Geschwindigkeiten zu jeweils 20 Wellenlängen.

4.1.2 Terminalausgabe bei 3 Geschwindigkeiten und 20 Wellenlängen



```
Anaconda Powershell Prompt (Anaconda3)
[Geschwindigkeit,Häufigkeit]
[[0.0, 0.0], [0.25, 0.0], [0.5, 0.0], [0.75, 0.0], [1.0, 20.0], [1.25, 0.0], [1.5, 0.0], [1.75, 0.0], [2.0, 0.0], [2.25, 0.0], [2.5, 0.0], [2.75, 0.0], [3.0, 0.0], [3.25, 0.0], [3.5, 0.0], [3.75, 0.0], [4.0, 0.0], [4.25, 0.0], [4.5, 0.0], [4.75, 0.0], [5.0, 0.0], [5.25, 0.0], [5.5, 20.0], [5.75, 0.0], [6.0, 0.0], [6.25, 0.0], [6.5, 0.0], [6.75, 0.0], [7.0, 0.0], [7.25, 0.0], [7.5, 0.0], [7.75, 0.0], [8.0, 0.0], [8.25, 0.0], [8.5, 0.0], [8.75, 0.0], [9.0, 0.0], [9.25, 0.0], [9.5, 0.0], [9.75, 0.0], [10.0, 20.0]]
```

Abbildung 2: Die Ausgabe im Terminal zum Histogramm aus 1. Man kann direkt ablesen, welche Geschwindigkeit wie häufig vorkommt, nämlich die 1.0 m/s im gesamten Datensatz 20 mal, ebenso die 5.5 m/s und die 10.0 m/s.

4.2 *data_creator_002*

Dieser Teil zeigt von der *metrik_speeds.py* erzeugt Bilder aus Daten, die mit dem *data_creator_002* erzeugt wurden. Dabei wird auf einige bereits angesprochene @To-Dos eingegangen.

4.2.1 Histogramm mit 3 Geschwindigkeiten und 20 Wellenlängen

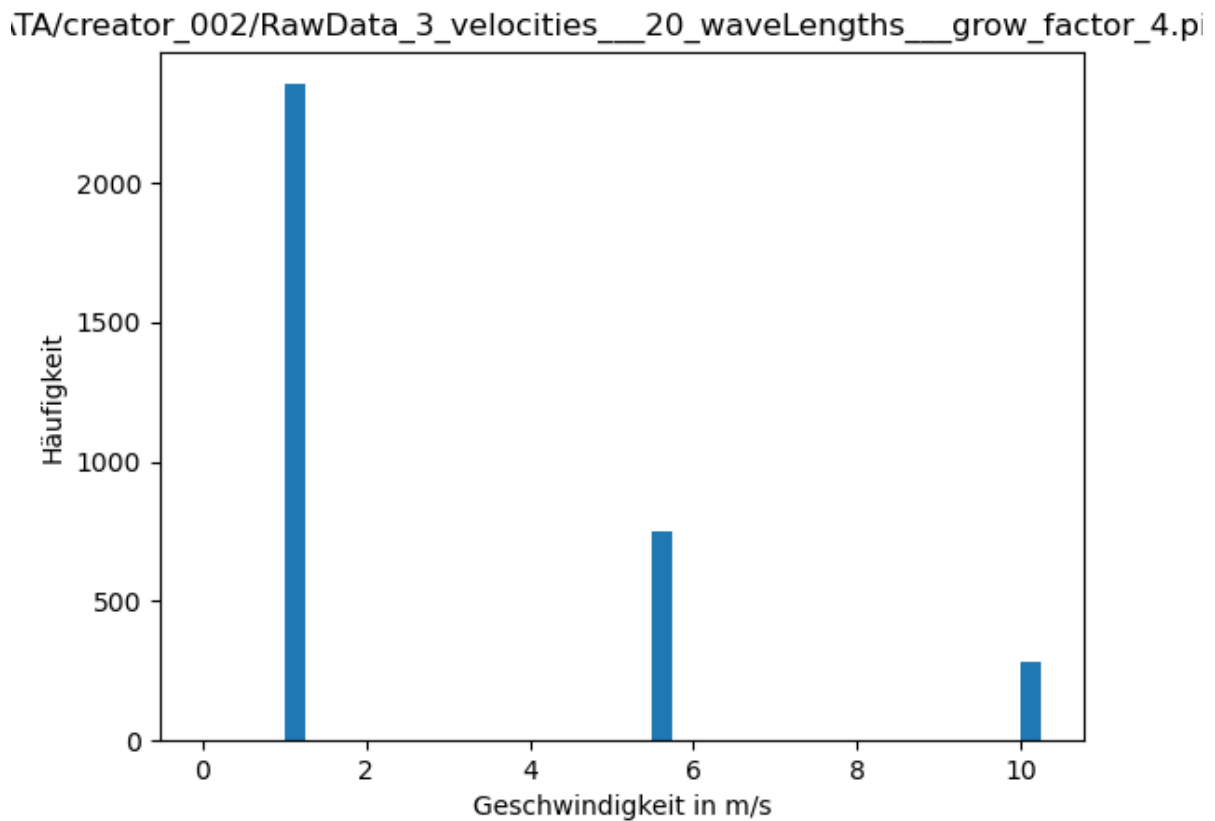
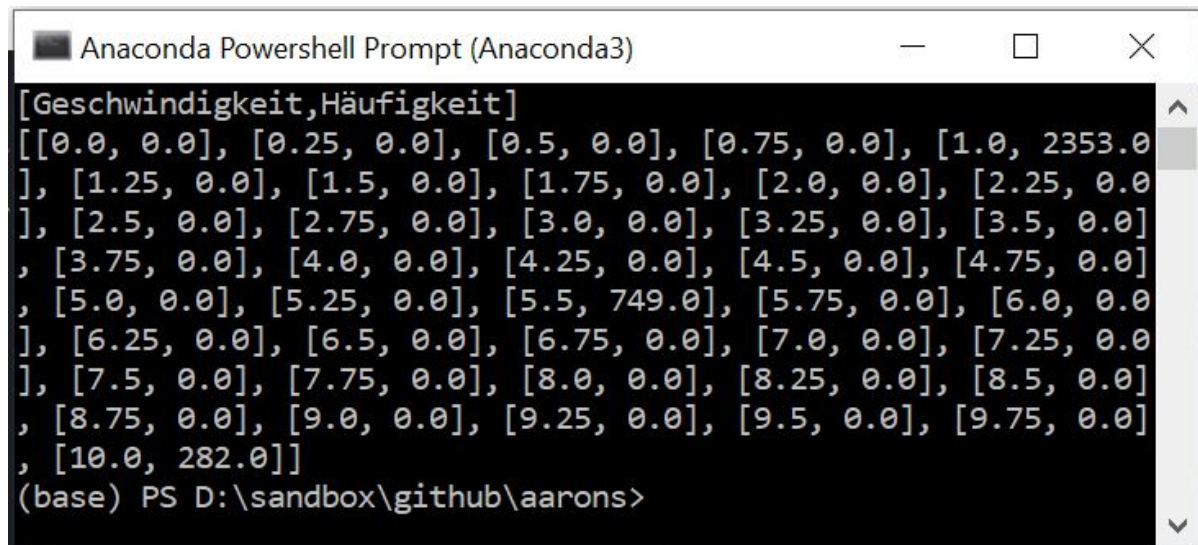


Abbildung 3: Das Histogramm eines Datensatzes, der mithilfe des *data_creator_002* erzeugt wurde. Auch dieser Datensatz enthält drei verschiedene Geschwindigkeiten, allerdings kommen langsame Geschwindigkeiten deutlich häufiger vor, als schnelle (vgl. 3.2).

4.2.2 Terminalausgabe bei 3 Geschwindigkeiten und 20 Wellenlängen



```
Anaconda Powershell Prompt (Anaconda3)
[Geschwindigkeit,Häufigkeit]
[[0.0, 0.0], [0.25, 0.0], [0.5, 0.0], [0.75, 0.0], [1.0, 2353.0], [1.25, 0.0], [1.5, 0.0], [1.75, 0.0], [2.0, 0.0], [2.25, 0.0], [2.5, 0.0], [2.75, 0.0], [3.0, 0.0], [3.25, 0.0], [3.5, 0.0], [3.75, 0.0], [4.0, 0.0], [4.25, 0.0], [4.5, 0.0], [4.75, 0.0], [5.0, 0.0], [5.25, 0.0], [5.5, 749.0], [5.75, 0.0], [6.0, 0.0], [6.25, 0.0], [6.5, 0.0], [6.75, 0.0], [7.0, 0.0], [7.25, 0.0], [7.5, 0.0], [7.75, 0.0], [8.0, 0.0], [8.25, 0.0], [8.5, 0.0], [8.75, 0.0], [9.0, 0.0], [9.25, 0.0], [9.5, 0.0], [9.75, 0.0], [10.0, 282.0]]
(base) PS D:\sandbox\github\aarons>
```

Abbildung 4: Die Ausgabe im Terminal zum Histogramm aus 3. Man erkennt, dass Wellen mit der Geschwindigkeit 1.0 m/s im gesamten Datensatz 2353 mal vorkommen, die Wellen mit 5.5 m/s kommen 749 mal vor und die mit 10.0 m/s nur 282 mal. Man beachte jedoch 2.5.1. Hier kann man erkennen, dass die Geschwindigkeiten, die mit Häufigkeit 0 auftreten, bereits bei relativ großen Schrittweiten, wie $\frac{1}{4}$ die Lesbarkeit stören (vgl. 2.5.2).

4.2.3 Histogramm mit 30 Geschwindigkeiten und 30 Wellenlängen

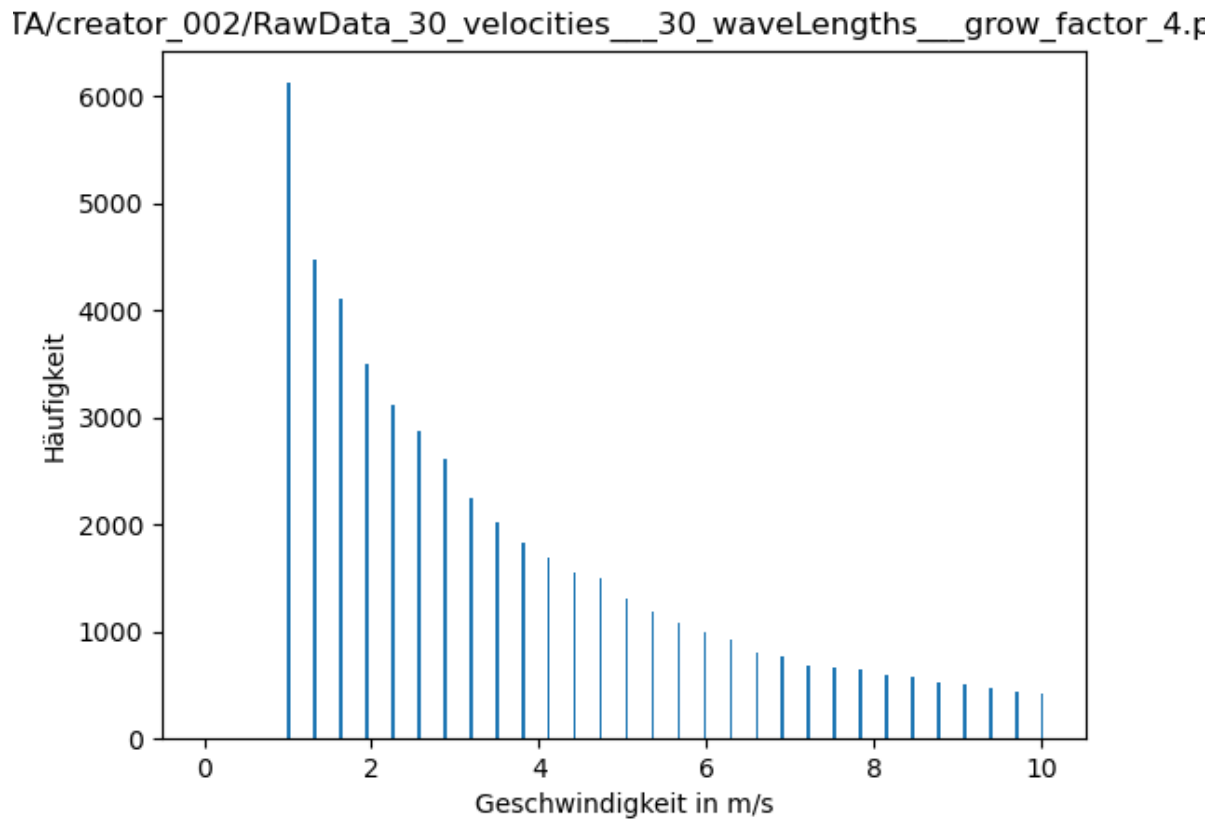
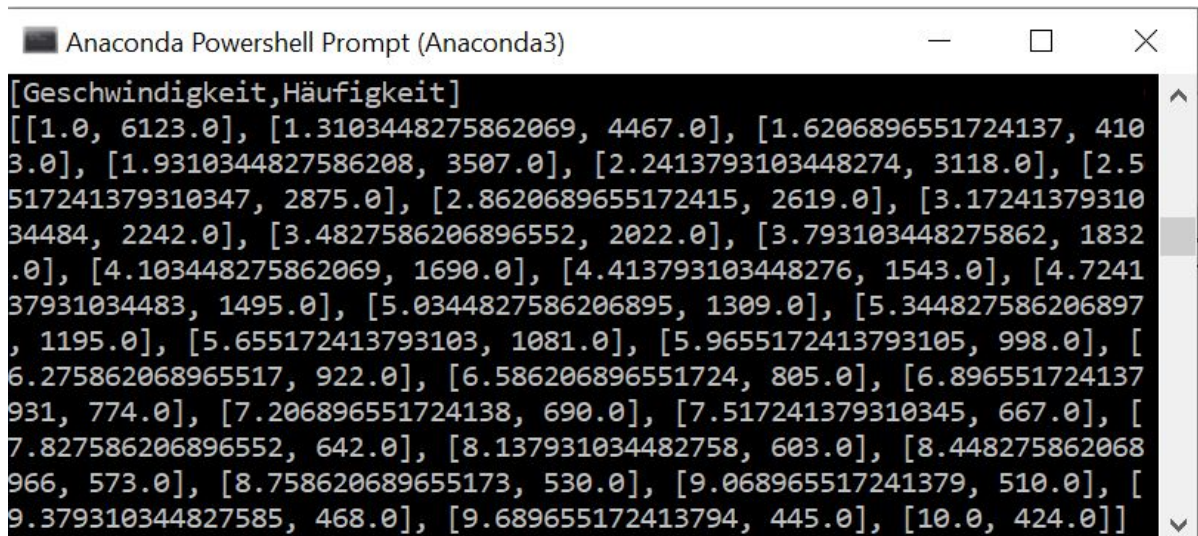


Abbildung 5: Das Histogramm eines Datensatzes, der mithilfe des *data_creator_002* erzeugt wurde. Dieser Datensatz enthält dreißig verschiedene Geschwindigkeiten, langsame Geschwindigkeiten kommen auch hier deutlich häufiger vor, als schnelle (vgl. 3.2). Die Anzahl an *bins* wurde manuell eingestellt (vgl. 2.5.1).

4.2.4 Terminalausgabe bei 30 Geschwindigkeiten und 30 Wellenlängen



```
Anaconda Powershell Prompt (Anaconda3)
[Geschwindigkeit,Häufigkeit]
[[1.0, 6123.0], [1.3103448275862069, 4467.0], [1.6206896551724137, 4103.0], [1.9310344827586208, 3507.0], [2.2413793103448274, 3118.0], [2.5517241379310347, 2875.0], [2.8620689655172415, 2619.0], [3.1724137931034484, 2242.0], [3.4827586206896552, 2022.0], [3.793103448275862, 1832.0], [4.103448275862069, 1690.0], [4.413793103448276, 1543.0], [4.724137931034483, 1495.0], [5.0344827586206895, 1309.0], [5.344827586206897, 1195.0], [5.655172413793103, 1081.0], [5.9655172413793105, 998.0], [6.275862068965517, 922.0], [6.586206896551724, 805.0], [6.896551724137931, 774.0], [7.206896551724138, 690.0], [7.517241379310345, 667.0], [7.827586206896552, 642.0], [8.137931034482758, 603.0], [8.448275862068966, 573.0], [8.758620689655173, 530.0], [9.068965517241379, 510.0], [9.379310344827585, 468.0], [9.689655172413794, 445.0], [10.0, 424.0]]
```

Abbildung 6: Die Ausgabe im Terminal zum Histogramm aus 5. Mittlerweile ist 2.5.2 erledigt. Es werden nur noch Geschwindigkeiten im Terminal angezeigt, die auch wirklich vorkommen.