API

OPEN ACCESS

Publiziert im Juni 2021
DOI: 10.15460/apimagazin.2021.2.2.77

Zeitschriftenartikel

Begutachtet

Begutachtet:

Dr. Steffen Rudolph
HAW Hamburg
Deutschland

Erhalten: *31. Mai 2021*

Akzeptiert: *2. Juni 2021*

Publiziert: *30. Juni 2021*

Empfohlene Zitierung:

BÜSCHENFELDT, Maika, 2021: Einführung in die Datenvisualisierung mit Python. In: *API Magazin* **2**(2) [Online] Verfügbar unter: DOI 10.15460/apimagazin.2021.2.2.77

Einführung in die Datenvisualisierung mit Python

Dr. Maika Büschenfeldt[1]*

[1] Hochschule für Angewandte Wissenschaften, Hamburg, Deutschland
Wissenschaftliche Mitarbeiterin am Department Information

* Korrespondenz: redaktion-api@haw-hamburg.de

Zusammenfassung

Durch die Aufbereitung und grafische Darstellung von Daten (Datenvisualisierung) lassen sich Zusammenhänge und Trends besser erkennen und verstehen. Das kompakte Tutorial bietet einen Einstieg in die Datenvisualisierung mit der Programmiersprache Python und dem Tool JupyterLab.

Schlagwörter: Python, Datenvisualisierung, Anaconda, Jupyter Notebook, Jupyter Lab

Introduction to data visualization with Python

Abstract

By preparing and graphically displaying data (data visualization), relationships and trends can be better recognized and understood. The compact tutorial provides an introduction to data visualization using the Python programming language and the JupyterLab tool.

Keywords: Python, Data Visualization, Ananconda, Jupyter Notebook, Jupyter Lab

DOI: 10.15460/apimagazin.2021.2.2.77

1 Einführung

Durch die Aufbereitung und grafische Darstellung von Daten (Datenvisualisierung) lassen sich Zusammenhänge und Trends besser erkennen und verstehen. In diesem kompakten Tutorial nähern wir uns der Visualisierung von Daten mit der Programmiersprache Python und dem Entwicklungstool Jupyter Lab an. Es wendet sich an Programmiereinsteiger*innen mit Interesse an Data-Science-Anwendungen. Das Tutorial ersetzt keinen richtigen Programmierkurs und ist auch keine umfassende Einführung in die Analyse und Visualisierung von Daten. Es handelt sich vielmehr um einen Einstieg in den Einstieg, um einen ersten Schritt auf dem Weg in die spannende Welt der Data-Science-Anwendungen. Wer im Anschluss an dieses Tutorial tiefer einsteigen möchte, findet am Ende des Artikels weiterführende Literatur sowie gute, frei im Internet verfügbare Tutorials und Referenzen.

2 Entwicklungsumgebung

Wir beginnen mit der Einrichtung der Arbeitsumgebung: Dazu installieren wir zunächst die Open-Source-Distribution ANACONDA für die Programmiersprachen Python und R. Die Distribution liefert alles, was zur Verarbeitung großer Datenmengen und der wissenschaftlichen Analyse benötigt wird. Sie bietet damit eine ideale Plattform für den Einstieg in die Datenwissenschaften. Die Installation von ANACONDA erfolgt für die Betriebssysteme Windows, MacOS und Linux durch einen grafischen Installer (Download unter: https://www.anaconda.com/products/individual). Die Installation ist einfach. Der Installer führt komfortabel durch jeden Installationsschritt.

Um nach der Installation mit der Arbeit zu beginnen, muss der ANACONDA Navigator gestartet werden. Im Navigatorfenster wird eine Übersicht aller unterstützten Tools angezeigt. Dort befindet sich auch der Button für den Start von Jupyter Lab. Jupyter Lab ist ein interaktives Webtool mit integriertem Webserver. Nach dem Start öffnet sich die Anwendung im Browser unter der Adresse http://localhost:8888/ (Abb. 1).

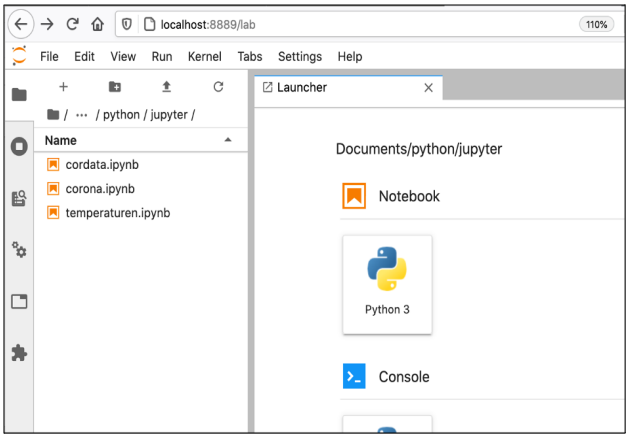

Abb. 1: Jupyter Launcher

 DOI: 10.15460/apimagazin.2021.2.2.77

Im linken Fenster befinden sich die Verzeichnisse und Dateien des eigenen Rechners. Im Dateienverzeichnis wird das gewünschte Arbeitsverzeichnis ausgewählt oder angelegt. Bereits existierende Jupyter-Notebooks werden per Mausklick geöffnet. Im rechten Fenster befindet sich der Launcher. Hier kann ein neues Jupyter-Notebook unter „Notebook/Python3“ angelegt werden. Die Datei des neuen Notebooks erhält zunächst den Namen „Untitled.ipynb“. Über das Kontextmenü (rechte Maustaste) kann die Datei umbenannt werden (Abb. 2).

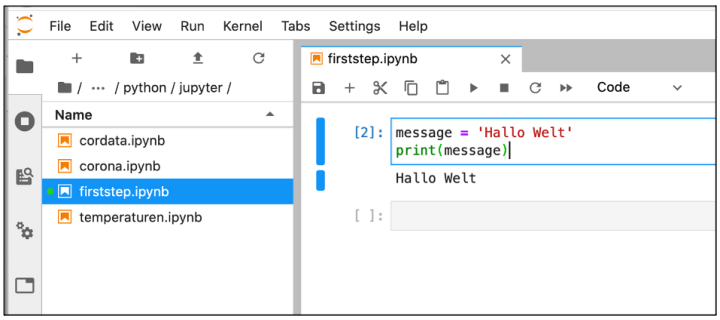

Abb. 2: Jupyter Notebook

Der Programmcode wird durch Betätigen des Run-Buttons ausgeführt.

3 Python – Variablen und Datenstrukturen

Python gilt als intuitive Programmiersprache, in der sich aufwändige Konstrukte mit wenigen Codezeilen umsetzen lassen. Die interaktive Python-Konsole (Python-Shell genannt) macht es möglich, Code auf die „Schnelle“ auszuführen. Sie bietet Zugriff auf alle in Python integrierten Funktionen und installierten Module. Die interaktive Konsole wird selbstverständlich auch in einem Jupyter-Notebook unterstützt. Ein Jupyter-Notebook hat darüber hinaus den Vorteil, dass nicht nur der Programmcode und die Ausgabe von Programmen, sondern auch erläuternde Texte in einem Dokument kombiniert werden können. Für die Visualisierung von Daten ist das grundlegende Verständnis des Python-Variablenkonzepts und der Datenstrukturen sehr hilfreich.

3.1 Variablen und Datentypen

Variablen sind Speicherbereiche für Daten. Sie haben einen Namen (Bezeichner) und einen Wert. Über den Bezeichner (Namen) kann auf den Variablenwert während des Programmlaufs zugegriffen werden. Wird einer Variable ein Startwert zugewiesen, wird diese dadurch initialisiert. Die Zuweisung eines Wertes erfolgt über den Zuweisungsoperator: " = ". Falls einer Variablen eine Zeichenkette (String) zugewiesen wird, muss diese wahlweise in Anführungszeichen oder in Hochkommas gesetzt werden.: `Gruss = "Hallo Welt"`. Bei einem Zahlwert werden keine Hochkommas/Anführungszeichen gesetzt: `Gehalt = 4000`. Einer Variablen kann jederzeit ein neuer Wert zugewiesen werden: `Gruss = "Guten Morgen"`.

 DOI: 10.15460/apimagazin.2021.2.2.77

Jeder Datentyp wird in Python mit einem Schlüsselwort (Keyword) definiert. Der Datentyp einer Variablen muss in Python nicht explizit festgelegt werden und kann sich während des Programmlaufs ändern. Es lassen sich die folgenden Datentypen unterscheiden:

Tab. 1 Python Datentypen

Datentyp	**Keyword**	**Beschreibung**
Integer	int	Ganzzahl
Float	float	Fließkommazahl
Complex	complex	Komplexe Zahl
String	string	Zeichenketten
Boolean	bool	Boolean

3.2 Datenstrukturen

In einer einfachen Variablen kann immer nur ein Wert gespeichert werden. Python stellt darüber hinaus mit den sogenannten Sequenzen eine Reihe von Objekten für komplexere Datenstrukturen bereit. In einer Sequenz können unter einem Bezeichner mehrere Elemente gespeichert werden. Jedes Element ist über einen Index oder ein Schlüsselwort ansprechbar. In Python unterscheiden wir Listen, Tupeln und Dictionaries.

Eine Liste kann aus Elementen eines Typs (homogene Liste) oder aus Elementen unterschiedlichen Typs (heterogene Listen) bestehen. Ineinander verschachtelte Listen können sehr komplexe Formen annehmen. Listen werden durch eckige Klammern dargestellt. Die einzelnen Elemente werden durch ein Komma voneinander getrennt. Die Zählung des Indexes beginnt immer bei 0.

```
leereListe = []
```

Homogene Liste aus Integer (Ganzzahl) Werten:

```
integerListe = [1,2,3,4,5,6,7]
```

Index	0	1	2	3	4	5	6
Wert	1	2	3	4	5	6	7

Homogene Liste aus Zeichenketten (Strings):

```
stringListe = ['Agata','Balduin','Camilla']
```

Index	0	1	2
Wert	Agata	Balduin	Camilla

 DOI: 10.15460/apimagazin.2021.2.2.77

Heterogene/Mehrdimensionale Liste bestehend aus Zahlen und Zeichen(ketten):
`mixListe = ["Hallo", [1,2,3], 5, "A`

<table>
<tr><td>Index</td><td>0</td><td colspan="2">1</td><td>2</td><td>3</td></tr>
<tr><td rowspan="3">Wert</td><td rowspan="3">Hallo</td><td>0</td><td>1</td><td rowspan="3">5</td><td rowspan="3">A</td></tr>
<tr><td>1</td><td>2</td></tr>
<tr><td>2</td><td>3</td></tr>
</table>

Ein Tupel ist eine Sequenz, deren Elemente nicht manipuliert werden können. Im Unterschied zur Liste ist es hier nicht möglich, Elemente hinzuzufügen, zu entfernen oder zu bearbeiten: tupel = (10,23,27,29).

In einer Liste oder einem Tupel können zusammenhängende Daten unter einem Bezeichner (Namen) zusammengefasst werden. Ein Dictionary hat den Vorteil, dass Werte unter einem Schlüsselwort gespeichert werden. Dieses Schlüsselwort kann Auskunft über die Beschaffenheit der gespeicherten Werte geben. Ein Dictionary besteht aus Schlüssel – Wert Paaren, die durch ein Komma abgetrennt werden und innerhalb von geschweiften Klammern stehen.

```
Person = {"Vorname":"Uschi","Nachname":"Krawuttke",
"Alter":47}
```

3.3 Auf Datenstrukturen zugreifen

Auf Listen und Tupel greifen wir über einen Index zu. Der Index beginnt immer mit 0! Auf das erste Element wird somit über den Indexwert 0 zugegriffen, auf das zweite Element über den Indexwert 1 usw. Um ein Gefühl für den Zugriff auf Datenstrukturen zu bekommen, ist es sinnvoll, die Ausgabe von Elementen aus Listen und Tupeln in einem Jupyter-Notebook zu üben.

```
# Sequenzen definieren und initialisieren
stringListe = ['Agata','Balduin','Camilla']
mixListe = ["Hallo", [1,2,3], 5, "A"]
meinTupel = ("üben","und immer wieder","üben")
person = {"Vorname" : "Uschi", "Nachname":"Krawuttke",
"Alter":47}
# Ausgabe ausgewählter Elemente
print(stringListe[1])
print(mixListe[0])
print(mixListe[1][2])
print(meinTupel[1])
print(person["Vorname"],person["Nachname"])
```

Durch das Hash-Zeichen (#) werden Kommentarzeilen gekennzeichnet. Kommentare sind Annotationen im Quellcode. Sie werden vom Python Interpreter ignoriert, helfen aber den Quelltext für Menschen leichter verständlich zu machen.

DOI: 10.15460/apimagazin.2021.2.2.77

4 Diagramme

Für die Visualisierung von Daten benötigen wir die Open-Source-Bibliothek matplotlib. Mit dieser Bibliothek können Daten mit wenigen Code-Zeilen auf unterschiedlichste Weise visualisiert werden: als Liniendiagramm, als Histogramm, Spektrum, Balkendiagramm, Tortendiagramm, Streudiagramm und vieles mehr. Erfahrenen Anwender*innen bietet matplotlib viele Konfigurationsmöglichkeiten für komplexe Darstellungen.

Bevor wir matplotlib verwenden können, müssen wir diese Bibliothek jedoch mit der import Anweisung einbinden: `import matplotlib.pyplot as plt`
Sollte es zur Fehlermeldung „`No module named …`" kommen ist das gewünschte Modul noch nicht installiert. Das lässt sich jedoch mit dem Python Paketinstaller pip schnell beheben. Im Terminal (Anklicken des + Buttons) ist das fehlende Modul mit einer Codezeile nachinstalliert: `pip install matplotlib.`

Über die Taskenkombination **Shift + Enter** werden die Eingaben im Terminal ausgeführt. Wenn alles erledigt ist, kann es losgehen.

4.1 Linien- und Balkendiagramme

Mit Linien- und Balkendiagrammen lassen sich zeitliche Verläufe und Trends darstellen. Sie geben Einblick in die Veränderung von Daten innerhalb eines bestimmten Zeitraums. Im nachfolgenden Beispiel verwenden wir die Daten des harmonisierten Verbraucherpreisindex des Statistischen Bundesamtes für das Jahr 2020 und stellen diese als Liniendiagramm dar (Abb. 3).[1]

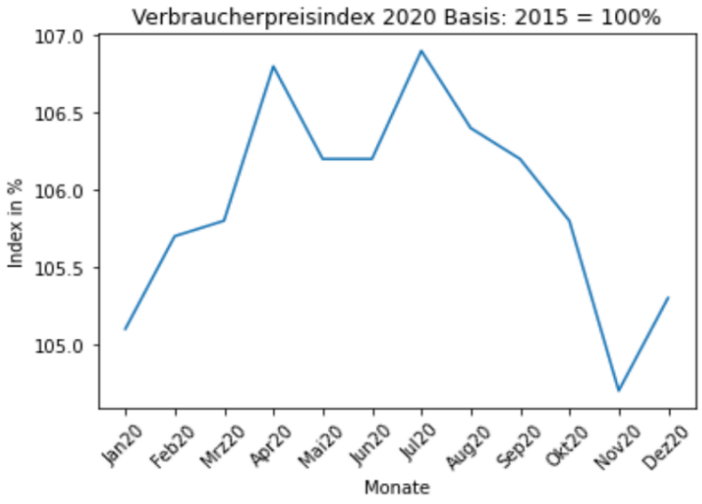

Abb. 3: Liniendiagramm Verbraucherpreisindex

1 Harmonisierter Verbraucherpreisindes DeStatis. Verfügbar unter: https://www.destatis.de/DE/Themen/Wirtschaft/Preise/Verbraucherpreisindex/Tabellen/Harmonisierter-Verbraucherpreisindex.html.

 DOI: 10.15460/apimagazin.2021.2.2.77

Mit der Bibliothek matplotlib sind dazu nur wenige Codezeilen nötig:

Schritt 1: Wir verwenden für die Daten Listen (vgl. Abschnitt 3): Unter dem Bezeichner monate speichern wir die Monate. Diese sollen im Liniendiagramm als Labels auf der X-Achse abgetragen werden. Unter dem Bezeichner vbindex speichern wir die monatlichen Werte des Verbraucherpreisindexes.

```
monate = ['Jan','Feb','Mrz','Apr','Mai','Jun','Jul','Aug
','Sep','Okt','Nov','Dez']
vbindex = [105.1,105.7,105.8,106.8,106.2,106.2,106.9,106
.4,106.2,105.8,104.7,105.3]
```

Schritt 2: Mit der Methode pyplot.subplots() wird das Axis-Objekt erzeugt. Wir können das Axis-Objekt als ein Behältnis begreifen, das die Daten unseres Diagramms sowie dessen X- und eine Y-Achse enthält. Das Axis-Objekt wird in unserem Beispiel über den Bezeichner ax angesprochen: fig, ax = plt.subplots()

Schritt 3: Mit der Funktion plot() schreiben wir die Linie in das Diagramm. Als Parameter übergeben wir die Listen mit unseren Daten: Als erstes die Liste monate für die Labels der X-Achse und als zweites die Liste vbindex für die auf der Y-Achse abgetragenen Indexwerte: `ax.plot(monate, vbindex)`

Schritt 4: Abschließend folgt die Beschriftung des Diagramms sowie die Beschriftungen der X- und Y-Achsen, die sogenannten *Ticklabels*. Da die Monate als Markierungen (*Ticks*) der X-Achse zu viel Platz verbrauchen, werden diese um 45 Grad gedreht. Das geschieht durch die Funktion des pyplot Objektes *xticks().*

```
ax.set_title("Verbraucherpreisindex 2020 Basis: 2015 =
100%")
ax.set_xlabel("Monate")
ax.set_ylabel("Index in %")
plt.xticks(rotation=45)
```

Balkendiagramme werden ähnlich umgesetzt wie Liniendiagramme. Im Unterschied zum Liniendiagramm verwenden wir zur Erzeugung des Balkendiagramms die Methode *bar()*. Das nachfolgende Coding erzeugt zwei nebeneinanderstehende Balkendiagramme, in denen die Fläche und die Bevölkerungszahl der neuen Bundesländer gegenübergestellt werden.

```
labels = ['Brandenburg','Sachsen-Anhalt','Sachsen',
'Thüringen','Mecklenburg-V','Berlin']
flaeche = [29654,20457,18450,16202,23295,891]
einwohner = [2552,2195,4072,2133,1608,3669]
fig, ax = plt.subplots(1,2, figsize=(12, 2))
for tick in ax[0].get_xticklabels():
    tick.set_rotation(90)
for tick in ax[1].get_xticklabels():
    tick.set_rotation(90)
ax[0].bar(labels, flaeche, color="orange")
ax[0].set_title("Fläche der neuen Bundesländer")
ax[0].set_ylabel("Fläche in qm")
```

 DOI: 10.15460/apimagazin.2021.2.2.77

```
ax[1].bar(labels, einwohner)
ax[1].set_title("Einwohnerzahl der neuen Bundesländer")
ax[1].set_ylabel("Einwohnerzahl in 1000")
```

Nebeneinanderstehende Diagramme werden über Parameterangaben der Funktion subplots() ermöglicht. Der erste Parameter gibt die Anzahl der Zeilen und der zweite Parameter die Anzahl der Spalten an. In unserem Fall ergeben eine Zeile und zwei Spalten zwei Diagramme bzw. werden zwei Axis-Objekte erzeugt (Abb. 4). Auf diesem Wege können beliebig viele Diagramme in einem Zeilen-Spalten-Raster erzeugt werden. Über den Parameter figsize wird der Abstand der Diagramme zueinander bestimmt. Die Axis-Objekte werden über einen Index angesprochen. Das erste Diagramm über den Index 0, das zweite Diagramm über den Index 1. Über den Index Labels auf der X-Achse, weil das Axes-Objekt über keine Methode zum Rotieren aller Labels verfügt. Wir müssen die Labels auf der X-Achse mit der Methode *get_xticks-labels()* auslesen und in einer FOR – Schleife über die Methode *set_rotation()* einzeln rotieren. In einer FOR Schleife wird im Schleifenkopf eine Bedingung formuliert. Die Anweisungen innerhalb des Schleifenblocks werden durchlaufen und immer wieder neu ausgeführt, solange die formulierte Bedingung *true* ist. In unserem Beispiel ist das der Fall so lange aus dem Axis-Objekt Beschriftungen der X-Achse (xticklabels) gezogen werden können.

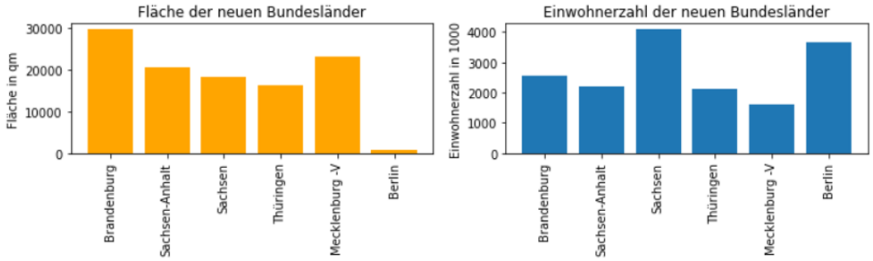

Abb. 4: Balkendiagramme Fläche und Einwohnerzahl der neuen Bundesländer

4.2 Tortendiagramme

Tortendiagramme zeigen die relative Zusammensetzung einer Variablen zu einem fixen Zeitpunkt. Jedes einzelne Tortenstück zeigt auf den ersten Blick, wie groß der jeweilige Anteil ist. Im nachfolgenden Beispiel vergleichen wir die Fläche der neuen Bundesländer miteinander (Abb. 5). Alle neuen Bundesländer ergeben zusammen 100%.[2]

Schritt 1: Wie im Linien- und Balkendiagramm beginnen wir auch hier mit dem Speichern der Daten in Listen. In der Liste *laender* werden die neuen Bundesländer gespeichert und in der Liste *flaeche* die Fläche der jeweiligen Länder in qkm. Wir definieren zusätzlich eine Liste mit dem Bezeichner *stabstand*. In dieser Liste werden

2 Statista. Verfügbar unter https://de.statista.com/statistik/daten/studie/154868/umfrage/flaeche-der-deutschen-bundeslaender/.

 DOI: 10.15460/apimagazin.2021.2.2.77

die Abstände der Tortenstücke festgelegt. Wir nutzen diese Werte, um das Bundesland Brandenburg als größtes Bundesland hervorzuheben. Brandenburg erhält deshalb an der ersten Position den Wert 0.1, während alle anderen Werte 0 betragen.

```
laender=['Brandenburg','Sachsen-Anhalt ','Sachsen',
'Thüringen','Mecklenburg-Vorpommern ','Berlin']
flaeche = [29654,20457,18450,16202,23295,891]
stabstand = [0.1,0,0,0,0,0]
```

Schritt 2: Erzeugen des Axis-Objektes: `fig, ax = plt.subplots()`

Schritt 3: Die Figur der Tortengrafik wird durch die Funktion *pie()* erzeugt. Als erster Parameter wird die Fläche übergeben. Die Flächenangaben in qkm werden automatisch in Prozentwerte umgerechnet. Der zweite Parameter (explode) bestimmt den Abstand der Tortenstücke zueinander. Es folgen als dritter Parameter die Namen der Länder. Sie werden zu den Labels der Tortenstücke. Über den vierten Parameter *autopct* werden Angaben zur Schrift und Schriftgröße gemacht. Der fünfte Parameter *pctdictance* bestimmt den Abstand der Prozentangaben zum Zentrum des Kreises.

```
ax.pie(flaeche,explode=stabstand,labels=laender,autopct='
%1.1f%%',pctdistance=0.5)
```

Schritt 4: Wie in den anderen Diagrammen auch, kann das Diagramm über die Funktion set_title() beschriftet werden: `ax.set_title("Prozentuale`

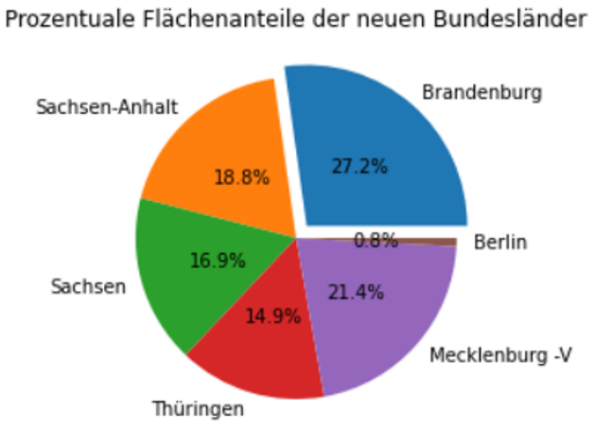

Abb. 5: Tortendiagramm mit der prozentualen Angabe der Fläche der neuen Bundesländer

```
Flächenanteile der neuen Bundesländer")
```

4.3 Diagramme kombinieren

Liniendiagramme können mit anderen Diagrammtypen kombiniert werden. Im folgenden Beispiel kombinieren wir in der Darstellung des harmonisierten Verbraucherpreisindexes ein Balkendiagramm mit einem Liniendiagramm (Abb. 6).

DOI: 10.15460/apimagazin.2021.2.2.77

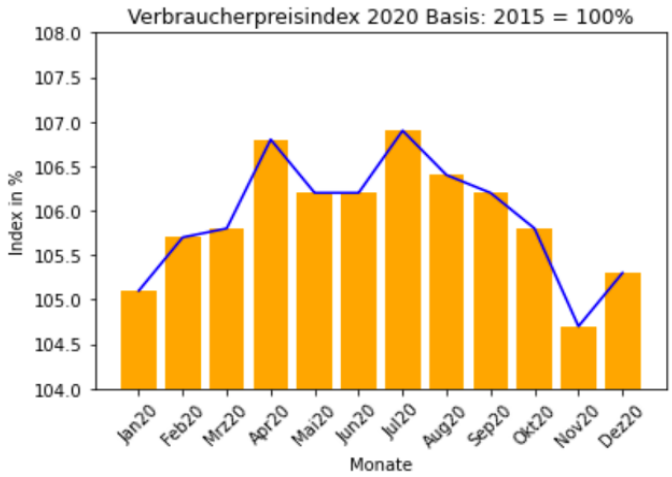

Abb. 6: Balken und Liniendiagramm Verbraucherpreisindex

Dazu werden die Funktionen bar() und plot() einfach nacheinander aufgerufen. Mit dem Parameter color können den Balken und der Linie beliebige Farben zugewiesen werden.

```
monate = ['Jan20','Feb20','Mrz20','Apr20','Mai20',
'Jun20','Jul20','Aug20','Sep20','Okt20','Nov20','Dez20']
vbindex = [105.1,105.7,105.8,106.8,106.2,106.2,106.9,106
.4,106.2,105.8,104.7,105.3,]
fig, ax = plt.subplots()
ax.bar(monate,vbindex,color='orange')
ax.plot(monate,vbindex,color='blue')
ax.set_ylim([104,108])
ax.set_title("Verbraucherpreisindex 2020 Basis: 2015 =
100%")
ax.set_xlabel("Monate")
ax.set_ylabel("Index in %")
plt.xticks(rotation=45)
```

4.4 Tupeln und Dictionaries

Neben Listen können für die Erzeugung von Diagrammen mit matplotlib auch Tupeln verwendet werden. Wenn die Daten nicht automatisiert abgerufen, sondern per Hand in den Quellcode eingefügt werden müssen, ist die Zuordnung von Key – Value-Werten in Dictionaries oft überschaubarer.

```
data = {'Jan20':105.1,
        'Feb20':105.7,
        'Mrz20':105.8,
        'Apr20':106.8,
        'Mai20':106.2,
        'Jun20':106.2,
        'Jul20':106.9,
        'Aug20':106.4,
        'Sep20':106.2,
        'Okt20':105.8,
        'Nov20':104.7,
        'Dez20':105.3
         }
```

DOI: 10.15460/apimagazin.2021.2.2.77

Da die behandelten Zeichen-Funktionen der matplotlib-Bibliothek nur Listen oder Tupeln akzeptieren, müssen die Daten eines Dictionaries jedoch zuvor über die Funktion list() in Listen konvertiert werden.

```
values = list(data.values())
labels = list(data.keys())
```

5 Ausblick

Die in diesem Tutorial vorgestellten Diagramme behandeln nur sehr wenige von sehr vielen Visualisierungsmethoden. Um weitere Diagrammtypen oder andere Visualisierungsmethoden wie Karten (Maps), Infografiken, Tabellen und interaktive Dashboards zu entdecken, lohnt der Besuch auf der Seite der Matplotlib-Community (https://matplotlib.org/).

Sehr empfehlenswert ist der Einsatz der Software-Bibliothek Pandas (Python Data Analysis Library). Die Bibliothek dient der Bearbeitung und Auswertung tabellarischer Daten und Zeitreihen. Sie stellt dazu besondere Funktionen, Werkzeuge und Datenstrukturen bereit. Dazu zählen unter anderem:

- Die Erweiterung der Python-Standardbibliothek um eindimensionale Listen (Series), zweidimensionale tabellenartige Listen (Dataframes) und dreidimensionale Listen (Panels).
- Funktionen zum Indexieren, Aufteilen, Aggregieren, Transformieren und Zusammenführen von Daten.
- Werkzeuge zur Datenbereinigung, Datenanpassung und Manipulation von Datenstrukturen.

An Pandas ist besonders attraktiv, dass diese Bibliothek das Einlesen von Daten aus verschiedenen Datenformaten und Quellen möglich macht. Mit Pandas können somit Daten über Programmierschnittstellen (APIs), aber auch durch das Einlesen von CSV-, JSON-, XML- oder Excel-Dateien abgerufen und unmittelbar in gut bearbeitbare Datenstrukturen überführt werden. Zusätzliche Bibliotheken wie lxml, html5lib und beautifulsoup4 ermöglichen darüber hinaus das Extrahieren von Daten aus HTML-Seiten (Web-Scraping).

Kurz: In Kombination mit einer reichen Auswahl an Bibliotheken bietet die Programmiersprache Python Tools und Funktionen für alle Phasen der Data-Science: vom Gewinnen über das Aufbereiten bis hin zum Visualisieren von Daten.

DOI: 10.15460/apimagazin.2021.2.2.77

Tutorials, Referenzen und Communities

Python und Anaconda

Python-Anaconda- Tutorial
https://www.sivakids.de/python-anaconda/
Eine sehr schöne interaktive Einführung in die Programmierung mit Python: https://docs.microsoft.com/de-de/learn/modules/intro-to-python/
Python-Kurs
https://www.python-kurs.eu/python3_kurs.php

Python Bibliotheken

Matplotlib
https://matplotlib.org/
Pandas
https://pandas.pydata.org/

Jupyter Lab

Jupyter-Lab-Dokumentation
https://jupyterlab.readthedocs.io/en/stable/

Daten gewinnen

Einführendes Tutorial zum Thema Web-Scraping in Deutsch:
https://www.ionos.de/digitalguide/websites/web-entwicklung/web-scraping-mit-python/
Beautiful Soup (HTML Parser) Dokumentation:
https://www.crummy.com/software/BeautifulSoup/bs4/doc/

Daten aufbereiten

Tutorial-Daten nach Tabellenextraktion aufbereiten:
https://pbpython.com/pandas-html-table.html
Tutorial zur Extraktion von Daten aus Pandas-Objekten:
https://medium.com/dunder-data/selecting-subsets-of-data-in-pandas-6fcd0170be9c

Daten visualisieren

Einstiegs-Tutorials auf der Projektseite von Mathplotlib:
https://matplotlib.org/3.1.1/tutorials/index.html

Hilfsreiche Literatur

KAHL, Timo und ZIMMER, Frank Hrsg., 2020. *Interaktive Datenvisualisierung in Wissenschaft und Unternehmenspraxis*. Wiesbaden: Springer Vieweg. ISBN 978-3-658-29562-2

DOI: 10.15460/apimagazin.2021.2.2.77

LUTZ, Mark, 2014. *Python – kurz und gut.* 5. Aufl. o.O: O'Reilly. O´Reillys Taschenbibliothek. ISBN 978-3-95561-770-7

MCDANIEL, Eileen und MCDANIEL, Stephen, 2012. *The Accidental Analyst: Show Your Data Who's Boss.* Seattle: Freakalytics. ISBN 978-1-47743-226-6