Pandas & JupyterLab

by Woche 6

Nun befassen wir uns endlich mit dem `pandas` Modul. Dieses Modul ist das Herzstück der Datenanalyse in Python. Wir werden “richtige” Daten in Tabellenstruktur importieren, bearbeiten, analysieren und exportieren. Mit anderen Wort: Alles bisher war Vorbereitung auf das, was jetzt kommt.

Wie auch bei numpy und matplotlib, ist es üblich, pandas unter einem Alias zu importieren. Der Standard-Alias für pandas ist `pd`, welchen man auch in dem Logo wiederfindet.

Showcase Pandas

Ohne, dass wir an dieser Stelle den Code schon verstehen, wollen wir uns hier mit Beispielen einen Pandas-Vorgeschmack geben. Dazu erzeugen wir erstmal eine Tabelle mit einigen Daten. Ja, richtig, wir arbeiten endlich mit Tabellen und nicht nur mit eindiemensionalen Sammlungen von Zahlen etc. Die Beispieldaten haben drei Spalten und sieben Zeilen:

```
import numpy as np
import pandas as pd

df = pd.DataFrame({
    'Kategorie': ['Gemüse', 'Gemüse', 'Backware', 'Backware', 'Getränk',
'Getränk', 'Gemüse', 'Backware'],
    'Preis': [1.20, 2.50, 0.80, 3.40, 2.00, 1.50, 3.00, 2.30], # Preise in
Euro
```

```
    'Kalorien': [50, 70, 250, 300, 120, 110, 40, 500] # Kalorien pro Produkt
})

print(df)
```

```
  Kategorie  Preis  Kalorien
0    Gemüse    1.2        50
1    Gemüse    2.5        70
2  Backware    0.8       250
3  Backware    3.4       300
4   Getränk    2.0       120
5   Getränk    1.5       110
6    Gemüse    3.0        40
7  Backware    2.3       500
```

Nun können wir mit den folgenden drei Zeilen Code die Daten in der Tabelle analysieren. In diesem Fall berechnen wir

- den Mittewlert, die Standardabweichung, das Minimum, den Median und das Maximum
- jeweils für die Spalten `Preis` und `Kalorien` und
- gruppiert pro `Kategorie`,
- wobei am Ende alle Werte auf zwei Nachkommastellen gerundet werden.

```
agg_funcs = ['mean', 'std', 'min', 'median', 'max']
agg_dict = {col: agg_funcs for col in ['Preis', 'Kalorien']}
summary = df.groupby('Kategorie').agg(agg_dict).round(2)

print(summary)
```

```
           Preis                        Kalorien                       
            mean   std  min median  max     mean     std  min median  max
Kategorie                                                               
Backware    2.17  1.31  0.8   2.30  3.4   350.00  132.29  250  300.0  500
Gemüse      2.23  0.93  1.2   2.50  3.0    53.33   15.28   40   50.0   70
Getränk     1.75  0.35  1.5   1.75  2.0   115.00    7.07  110  115.0  120
```

Man sieht also, dass wir mit Pandas sehr mächtige Funktionen zur Verfügung haben, um Daten zu analysieren. Zum Vergleich soll überlegt werden wie wir alle dies mit unserem bisherigen Wissen umgesetzt hätten.

Auch um Daten zu bearbeiten, bietet Pandas viele Funktionen. So können wir die Tabelle mit folgenden zwei Zeilen Code

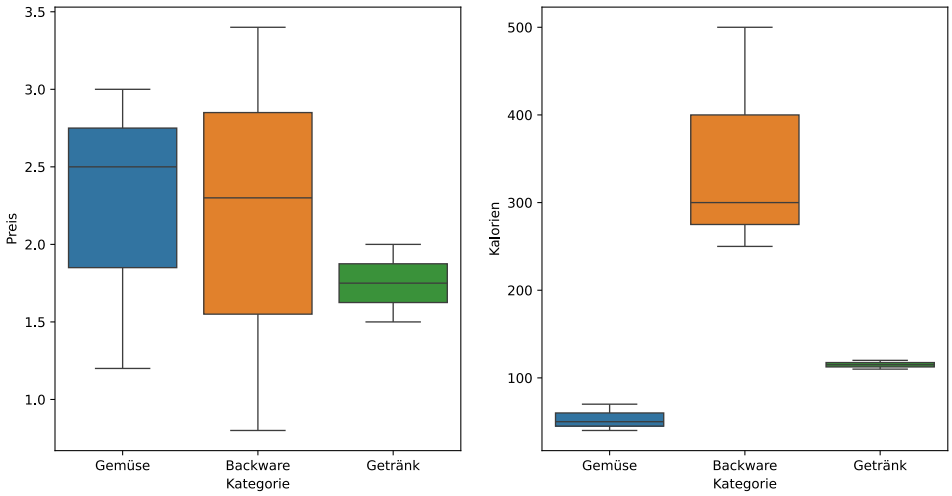
- in eine “lange” Form bringen, in der `Preis` und `Kalorien` in einer Spalte `Merkmal` zusammengefasst sind und
- den Spalten deutsche Namen geben.

```
newnames = {'level_1':'Merkmal', 'mean':'MW', 'std':'StdAbw', 'min':'Min',
'median':'Median', 'max':'Max'}
summary_long = summary.stack(level=0,
future_stack=True).reset_index().rename(columns=newnames)

print(summary_long)
```

```
  Kategorie   Merkmal      MW  StdAbw    Min  Median    Max
0  Backware     Preis    2.17    1.31    0.8    2.30    3.4
1  Backware  Kalorien  350.00  132.29  250.0  300.00  500.0
2    Gemüse     Preis    2.23    0.93    1.2    2.50    3.0
3    Gemüse  Kalorien   53.33   15.28   40.0   50.00   70.0
4   Getränk     Preis    1.75    0.35    1.5    1.75    2.0
5   Getränk  Kalorien  115.00    7.07  110.0  115.00  120.0
```

Schließlich können wir die Tabelle auch direkt an Matplotlib und Seaborn übergeben, um sie zu visualisieren. Hier ein Beispiel, wie wir Boxplots für `Preis` und `Kalorien` pro `Kategorie` erstellen könnten:

```
import seaborn as sns
import matplotlib.pyplot as plt

fig, axes = plt.subplots(1, 2, figsize=(12, 6))
sns.boxplot(data=df, x='Kategorie', y='Preis',    ax=axes[0], hue='Kategorie')
sns.boxplot(data=df, x='Kategorie', y='Kalorien', ax=axes[1], hue='Kategorie')
plt.show()
```



Diese und weitere Funktionalitäten von Pandas gilt es also in den nächsten Kapiteln kennenzulernen.

Wechsel: Jupyter Notebooks → JupyterLabs

Bisher haben wir in **Jupyter Notebooks** gearbeitet, also sämtlichen Code darüber ausgeführt, mit Markdown-Zellen ergänzt und so mehrere `.ipynb`-Dateien erstellt. Als Einstieg in die Datenanalyse ist das nicht nur in Ordnung, sondern auch förderlich, da das simplere Layout und die einfache Handhabung von Jupyter Notebooks den Fokus auf das Wesentliche legen: Erstmal zu lernen mit Python zu programmieren.

Da wir aber nun in die “richtige” Datenanalyse einsteigen, ist jetzt ein guter Zeitpunkt um eine neue Arbeitsumgebung kennenzulernen. Ein Grund dafür ist z.B., dass wir externe Dateien wie `.csv`-Dateien importieren wollen. Demnach findet unsere Arbeit nicht mehr nur “in einem Notebook” statt, sondern bezieht auch weitere Dateien mit ein. Das ist natürlich auch in der Praxis eines Data Analysten so. Jupyter Notebooks ist dafür nicht unbedingt die beste Umgebung, da man in einem Tab seines Browsers stets nur eine Datei bearbeiten und überblicken kann.

Ganz zu Beginn dieses Kurses wurde bereits erwähnt, dass es auch andere Arbeitsumgebungen gibt. Wir wollen hier bzgl. der Möglichkeiten einen Schritt nach vorne machen, uns gleichzeitig aber auch nicht so weit vom bisherigen entfernen, dass wir uns überfordert fühlen.

Daher bietet sich **JupyterLabs** an. JupyterLabs ist Jupyter Notebooks sehr ähnlich, bietet aber eine Reihe von zusätzlichen Funktionen, die das Arbeiten mit Daten erleichtern. JupyterLabs ist eine Weiterentwicklung von Jupyter Notebooks und beide stammen von Project Jupyter, einer Non-Profit-Organisation, die sich der Entwicklung

von Open-Source-Software für interaktive Datenanalyse und wissenschaftliche Berechnungen verschrieben hat.

Es muss nichts installiert werden, da JupyterLabs bereits in Anaconda enthalten ist. Wir werden auch weiterhin `.ipynb`-Dateien verwenden. Hier sind Screenshots von Jupyter Notebooks und JupyterLabs, um zu verdeutlichen wie ähnlich sie sich sind:

JupyterLabs nutzen

Um JupyterLabs zu starten, öffnen wir den Anaconda Navigator und klicken unter JupyterLab auf `Launch`. Alternativ können wir auch wieder in der Konsole (=Anaconda Prompt) `jupyter lab` eingeben. In diesem Video sind weitere Informationen dazu:

https://youtu.be/SAjwOJgZ1Js?si=hdP4qj1rVzFfAHf5

Ab dem jetzigen Zeitpunkt sollte also JupyterLabs die bevorzugte Arbeitsumgebung sein.

Übungen

Öffne eines deiner alten `.ipynb` Notebooks - also eins, das du in den letzten Wochen mit Jupyter Notebooks erstellt hast - in JupyterLabs.

- (A) Geschafft