Datenexport

by Woche 15

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

In diesem Kapitel wollen wir uns vor allem mit dem Export von Daten beschäftigen, aber auch noch etwas mehr zum Import lernen. Tatsächlich wurde in Kapitel 6.2 ja nur erklärt wie wir Abbildungen, aber keine Daten exportieren können. Das wollen wir nun nachholen.

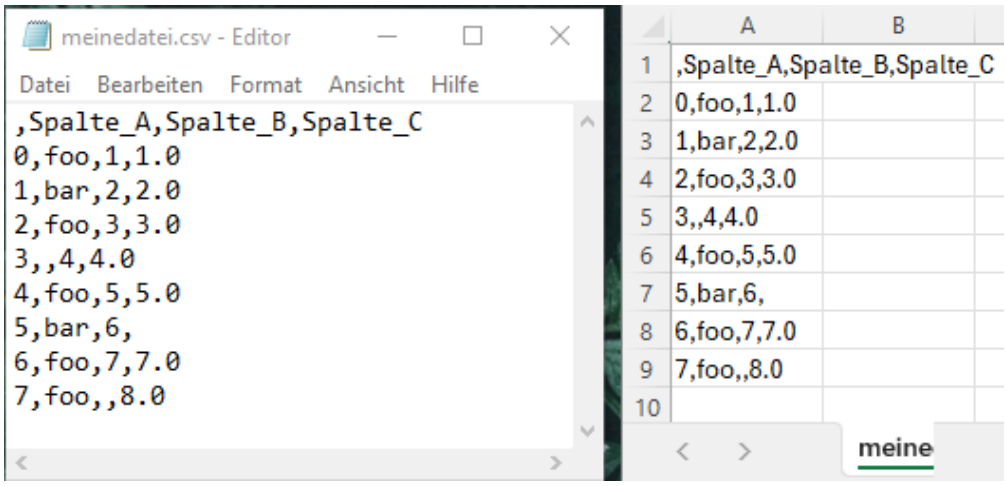
CSV-Dateien

Wie schon in Kapitel 5.3 zum Import von Daten wollen wir mit dem ggf. populärsten Format für den Datenaustausch beginnen: CSV-Dateien. Für den Import nutzen wir die Pandas Funktion `pd.read_csv()`. Für den Export gibt es die Methode `to_csv()` für den entsprechenden DataFrame.

```
df = pd.DataFrame({
    'Spalte_A': ['foo', 'bar', 'foo', pd.NA, 'foo', 'bar', 'foo', 'foo'],
    'Spalte_B': [1, 2, 3, 4, 5, 6, 7, pd.NA],
    'Spalte_C': [1.0, 2.0, 3.0, 4.0, 5.0, pd.NA, 7.0, 8.0]
})

df.to_csv('meinedatei.csv')
```

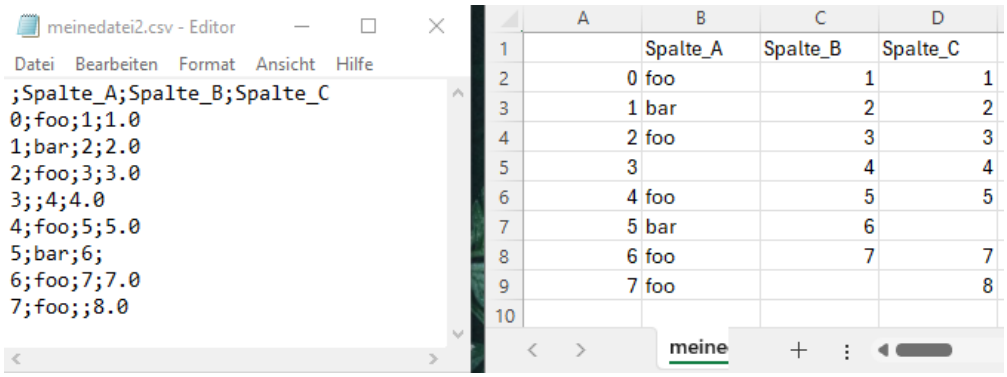
Wenn wir in Jupyter Labs oder Jupyter Notebook arbeiten, wird die Datei namens *meinedatei.csv* im gleichen Verzeichnis gespeichert, in dem auch das Notebook liegt. Sie ist außerdem erwartungsgemäß mit Komma als Trennzeichen gespeichert, was allerdings dazu führt, dass man sie in Excel nicht im gewünschten Format öffnen kann:



Das liegt daran, dass Excel in Deutschland standardmäßig ein Semikolon als Trennzeichen erwartet, was wiederum daran liegt, dass in Deutschland das Komma (und nicht ein Punkt) als Dezimaltrennzeichen verwendet wird. Das können wir aber auch in Pandas einstellen:

```
df.to_csv('meinedatei2.csv', sep=';')
```

Die Datei *meinedatei2.csv* wird dann mit einem Semikolon als Trennzeichen gespeichert und kann gut in Excel geöffnet werden.



Außerdem könnten folgende Argumente für `to_csv()` interessant sein:

- `index`: Standardmäßig wird der Index des DataFrames mit exportiert, aber in einem Fall wie diesem ist der Index des DataFrames nicht besonders wichtig, sondern eher störend, sodass man ihn mit dem Argument `index=False` auch weglassen kann.
- `header`: Standardmäßig wird auch die Kopfzeile, also die Spaltennamen mit exportiert. Falls dies mal nicht gewünscht sein sollte lässt sich auch das mit `header=False` abstellen.

- `decimal`: Das Dezimaltrennzeichen ist standardmäßig ein Punkt (`decimal='.'`), kann aber auch auf ein Komma geändert werden (`decimal=','`).
- `na_rep`: Standardmäßig werden fehlende Werte so exportiert, dass sie als leere Zellen dargestellt werden. Manchmal kann es aber auch gewünscht sein, dass Fehlwerte als ein bestimmter Wert dargestellt werden, z.B. als String `NA` oder gar als `-9999`. Das lässt sich mit `na_rep='NA'` bzw. `na_rep='-9999'` einstellen.

Alle weiteren Argumente können in der Dokumentation nachgelesen werden.

Excel-Dateien

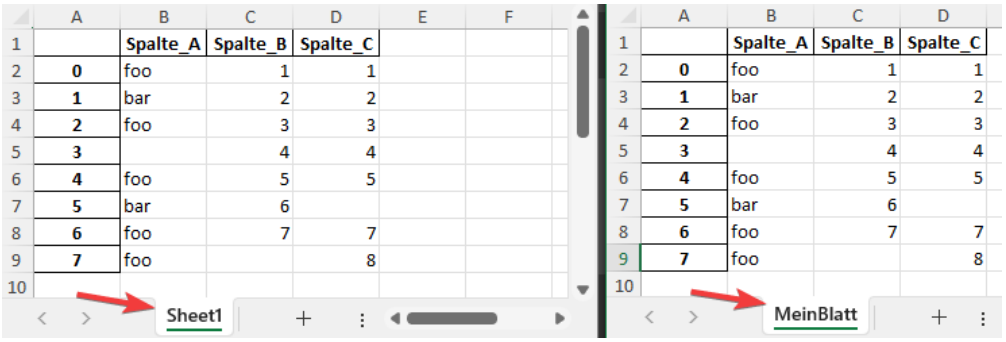
Auch Excel-Dateien lassen sich mit Pandas exportieren mittels der Methode `to_excel()`. Hier gilt es allerdings zu unterscheiden zwischen den zwei unterschiedlich komplexen Fällen ob man nur eine einzelne Tabelle oder mehrere Tabellen in dieselbe Datei exportieren möchte.

Eine Tabelle in eine Datei

Will man - wie auch bei `.to_csv()` - nur eine Tabelle in eine Datei exportieren, so ist das sehr einfach und man braucht nicht mal unbedingt den Tabellenblattnamen anzugeben, man kann es aber tun:

```
df.to_excel('meinedatei1.xlsx')
```

```
df.to_excel('meinedatei2.xlsx', sheet_name='MeinBlatt')
```



		Spalte_A	Spalte_B	Spalte_C
0	0	foo	1	1
1	1	bar	2	2
2	2	foo	3	3
3	3		4	4
4	4	foo	5	5
5	5	bar	6	
6	6	foo	7	7
7	7	foo		8

Auch hier gibt es wieder Argumente wie `index`, `header`, `na_rep`. Außerdem gibt es noch die Argumente `startrow` und `startcol`, mit denen festgelegt werden kann ab welcher Zeile bzw. Spalte die Daten geschrieben werden sollen. Dies ermöglicht es z.B. mehrere Tabellen in ein Tabellenblatt zu schreiben, was zumindest für Präsentationen oder Berichte interessant sein könnte.

Mehrere Tabellen in eine Datei

Noch interessanter ist jedoch die Möglichkeit gleich mehrere Tabellen/Ergebnisse auf mehrere Tabellenblätter derselben Excel-Datei zu schreiben. Dafür muss ein sogenanntes *ExcelWriter* Objekt erstellt werden, welches dann die einzelnen Tabellenblätter schreibt. Zunächst benötigen wir allerdings eine zweite Tabelle, die wir exportieren können:

```
df2 = pd.DataFrame({
    'Spalte_D': ['foo', 'bar', 'foo', pd.NA],
    'Spalte_E': [5, 6, 7, pd.NA],
    'Spalte_F': [5.0, pd.NA, 7.0, 8.0]
})
```

Nun erstellen wir das *ExcelWriter* Objekt und schreiben die beiden Tabellen in die Datei:

```
with pd.ExcelWriter('meinedatei3.xlsx') as writer:
    df.to_excel(writer, sheet_name='MeineTabelle1')
    df2.to_excel(writer, sheet_name='MeineTabelle2')
```

Weitere Ressourcen

- Export Pandas DataFrames to new & existing Excel workbook Ab 2:15 ist es optional, weil über Kursinhalt hinaus