Verschiedenes anderes zur Virsualisierung

by Woche 13

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

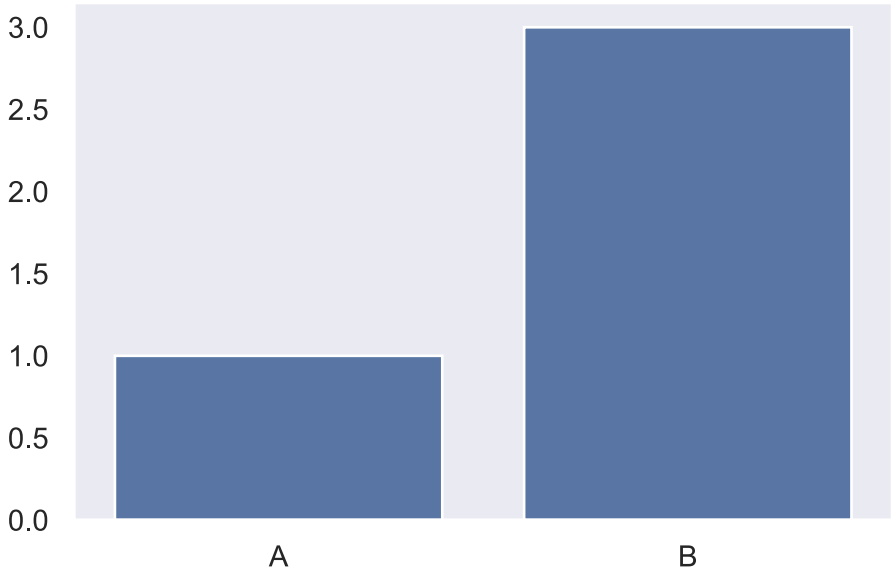
Dieses Kapitel ist das letzte in diesem Kurs, welches sich vorrangig auf die Visualisierung von Daten fokussiert. Als Abschluss werden hier verschiedene, nicht unbedingt zusammenhängende Themen behandelt, Tips gegeben und Ausblicke auf weiterführende Themen gegeben.

Layout/Erscheinungsbild

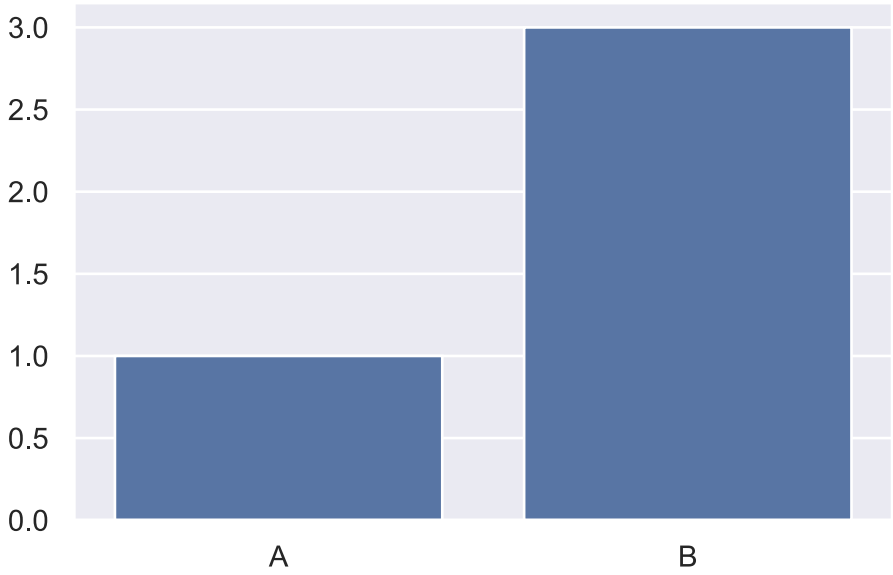
In den bisherigen Kapiteln haben wir uns hauptsächlich darauf konzentriert Diagramme zu erstellen, die die Daten möglichst gut darstellen. Wir haben uns also auf den Teil des Diagramms konzentriert, der die Daten enthält. Man kann aber natürlich auch entscheiden den Rest des Diagramms, also das Layout/Erscheinungsbild auch bekannt als “aesthetics”, anzupassen. Das kann dazu beitragen, dass das Diagramm besser verstanden wird oder einfach schöner bzw. professioneller aussieht.

Eine Möglichkeit dies mit möglichst wenig Aufwand zu erreichen, ist die Verwendung von vorgefertigten Themes. Seaborn bietet hierfür die `set_theme`-Funktion an. Was auch immer mit `set_theme` festegelegt wird, gilt für alle folgenden Diagramme. Die Funktion hat verschiedene Argumente, die das Aussehen des Diagramms beeinflussen. Beim Argument `style` hat man folgende Optionen: `dark`, `darkgrid`, `ticks`, `white` and `whitegrid`.

```
sns.set_theme(style="dark")
plt.figure()
sns.barplot(x=["A", "B"], y=[1, 3])
plt.show()
```

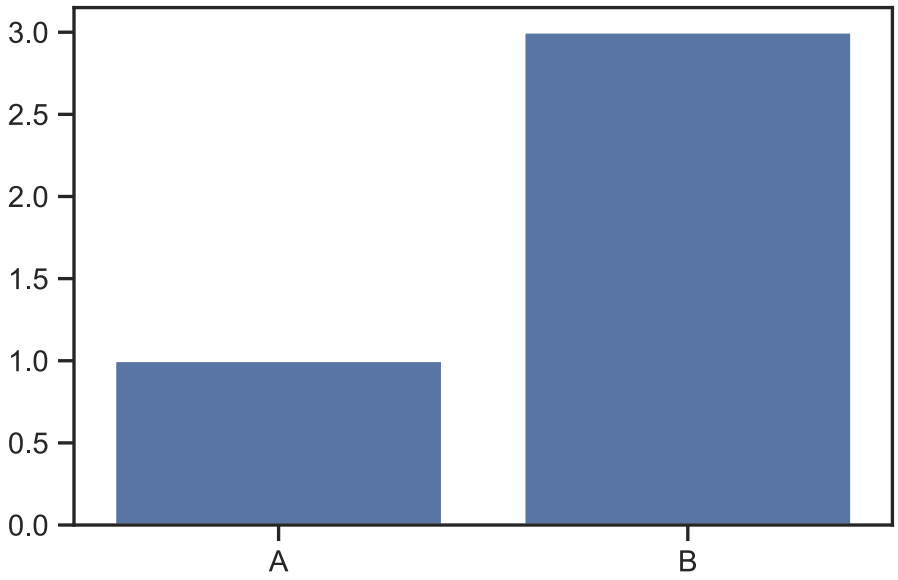



```
sns.set_theme(style="darkgrid")
plt.figure()
sns.barplot(x=["A", "B"], y=[1, 3])
plt.show()
```

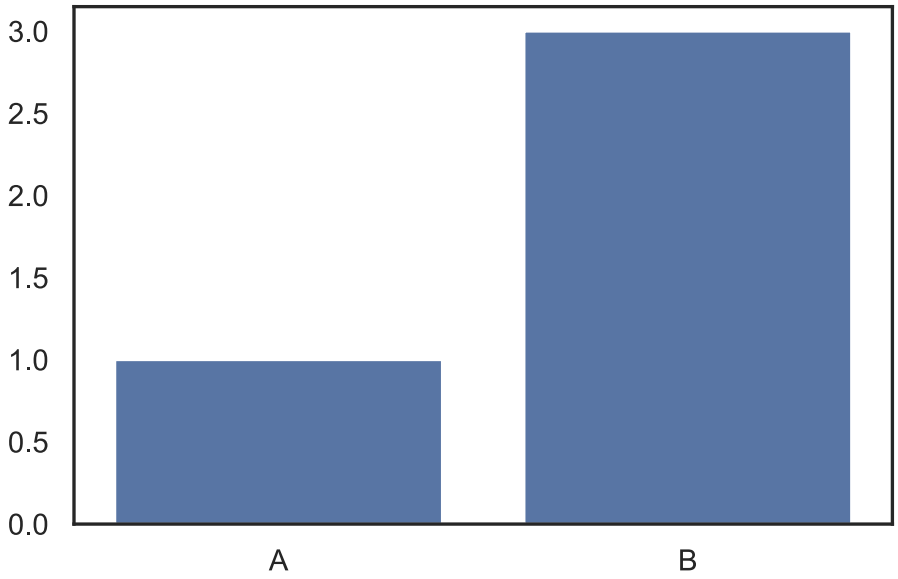


2

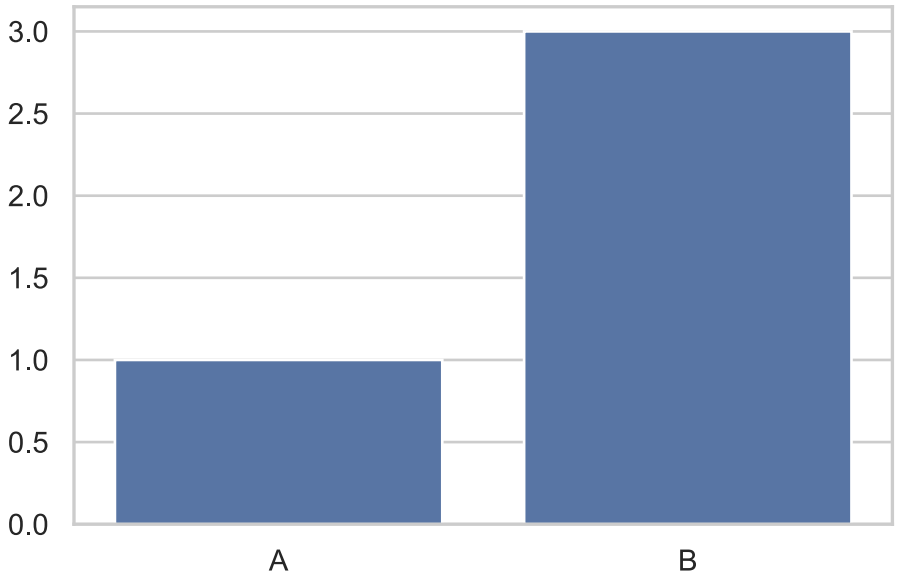
```
sns.set_theme(style="ticks")
plt.figure()
sns.barplot(x=["A", "B"], y=[1, 3])
plt.show()
```




```
sns.set_theme(style="white")
plt.figure()
sns.barplot(x=["A", "B"], y=[1, 3])
plt.show()
```



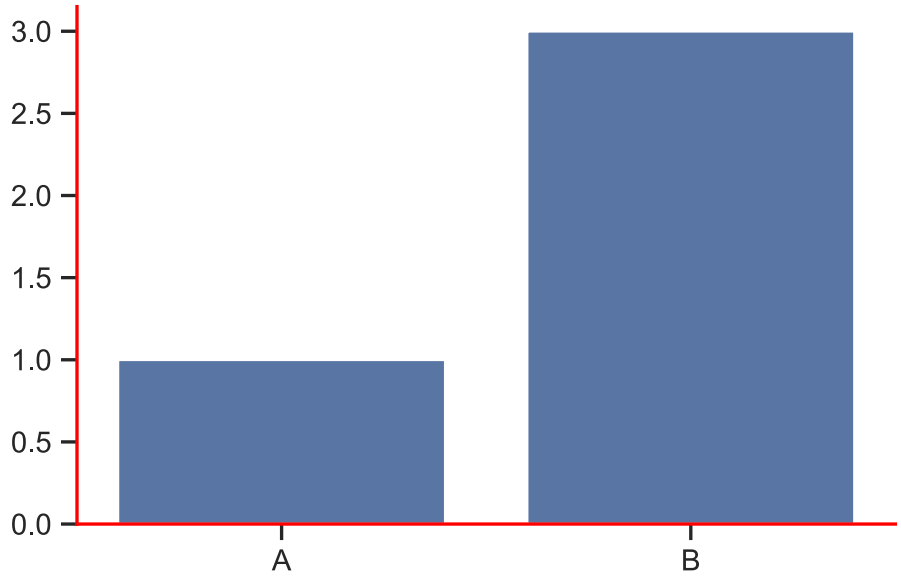

```
sns.set_theme(style="whitegrid")
plt.figure()
sns.barplot(x=["A", "B"], y=[1, 3])
plt.show()
```



4

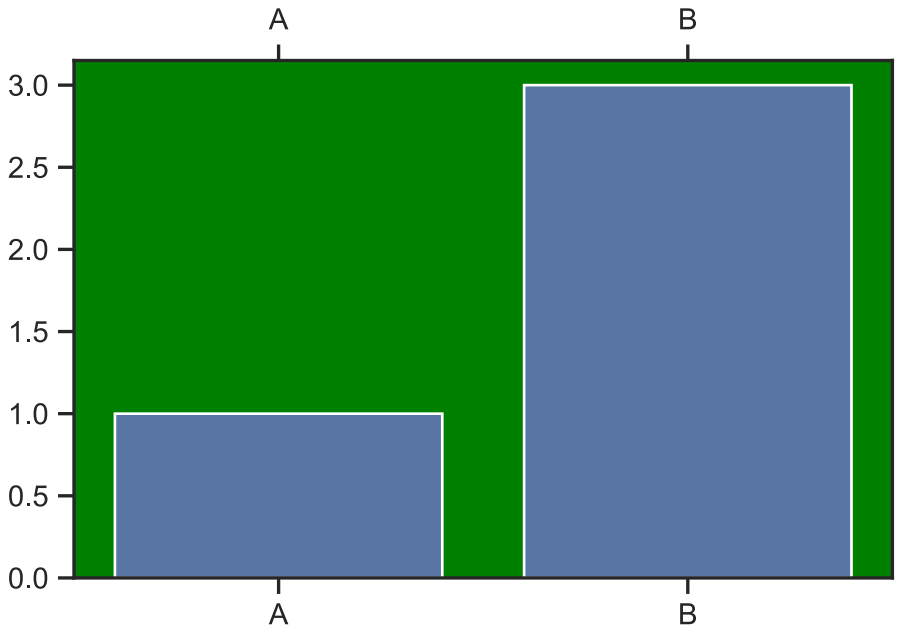
Möchte man das Aussehen des Diagramms noch weiter anpassen, können die anderen Argumente genutzt werden. Insbesondere mit `rc` können viele Einstellungen vorgenommen werden. Hier ein paar Beispiele:

```
meine_rc = {
  "axes.spines.right": False,
  "axes.spines.top": False,
  "axes.edgecolor": "red"
}
sns.set_theme(
  style="ticks",
  rc = meine_rc
)
plt.figure()
sns.barplot(x=["A", "B"], y=[1, 3])
plt.show()
```




```
meine_rc = {
  "axes.facecolor": "green",
  "xtick.top": True,
  "xtick.labeltop": True
}
sns.set_theme(
  style="ticks",
  rc = meine_rc
)
```

```
plt.figure()
sns.barplot(x=["A", "B"], y=[1, 3])
plt.show()
```



Um zu sehen, welche Einstellungen möglich und sind und wie diese standardmäßig eingestellt sind, könnte man auf `plt.rcParams` mit z.B. folgende Schleife zugreifen. Um die Ausgabe nicht unnötig lang zu machen, wird hier nur nach Einstellungen gesucht, die mit “xtick” beginnen - also Einstellungen speziell für die Ticks (Markierungen) an der x-Achse. Ihr könnte aber natürlich einfach das `if key.startswith("xtick"):` löschen um alle Einstellungen zu sehen.

```
for key in plt.rcParams:
    if key.startswith("xtick"):
        print(f"{key}: {plt.rcParams[key]}")
```

```
xtick.alignment: center
xtick.bottom: True
xtick.color: .15
xtick.direction: out
xtick.labelbottom: True
xtick.labelcolor: inherit
xtick.labelsize: 11.0
xtick.labeltop: True
xtick.major.bottom: True
```

```
xtick.major.pad: 3.5
xtick.major.size: 6.0
xtick.major.top: True
xtick.major.width: 1.25
xtick.minor.bottom: True
xtick.minor.ndivs: auto
xtick.minor.pad: 3.4
xtick.minor.size: 4.0
xtick.minor.top: True
xtick.minor.visible: False
xtick.minor.width: 1.0
xtick.top: True
```

Zeit für flexibles Ausprobieren

Darüber hinaus haben wir natürlich noch Unmengen von Aspekten der Datenvisualisierung nicht behandeln können. So haben wir nicht mal alle der gängigsten Diagrammtypen erzeugt. Anstatt dies nun nachzuholen, soll hier einfach etwas mehr Raum und Zeit als sonst sein um sich mit den weiteren Ressourcen unten zu beschäftigen.

Die ersten drei sind noch obligatorisch für alle und speziell die 14 Tipps in *Friends Don't Let Friends Make Bad Graphs* sind sehr empfehlenswert und dürften euch in bestimmten Situationen einen Vorsprung gegenüber anderen verschaffen.

Hat man erstmal ein gewisses Level in der Datenvisualisierung erreicht, lernt man meines Erachtens am meisten durch den Versuch eine bestimmte Visualisierung nachzubauen. Im Idealfall gibt es für die gewünschte Visualisierung auch den zugrundeliegenden Code, ggf. sogar als Tutorial aufbereitet. Genau sowas findet ihr in *Best Python Chart Examples*, sodass ihr - je nachdem wie viel Zeit ihr investieren wollt - euch an verschiedenen Diagrammen versuchen könnt.

Weitere Ressourcen

- Dokumentation: seaborn.set_theme
- Dokumentation: Controlling figure aesthetics
- Friends Don't Let Friends Make Bad Graphs

!Optional weil über Kursinhalt hinaus!

- Matplotlib cheatsheets and handouts
- Seaborn cheat sheet - A guide to most graphs
- Best Python Chart Examples
  - Daraus z.B. dieser hier