Facetten

by Woche 13

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

In diesem Kapitel geht es um die Darstellung von Daten in Facetten. Facetten sind eine Möglichkeit, um Daten in mehreren Teilen darzustellen. Dabei wird im Prinzip das gleiche Diagramm mehrfach erstellt, wobei die Daten in den einzelnen Diagrammen unterschiedlich gefiltert werden. Wie schon im vorangegangenen Kapitel nutzen wir die `palmer_penguins`-Daten.

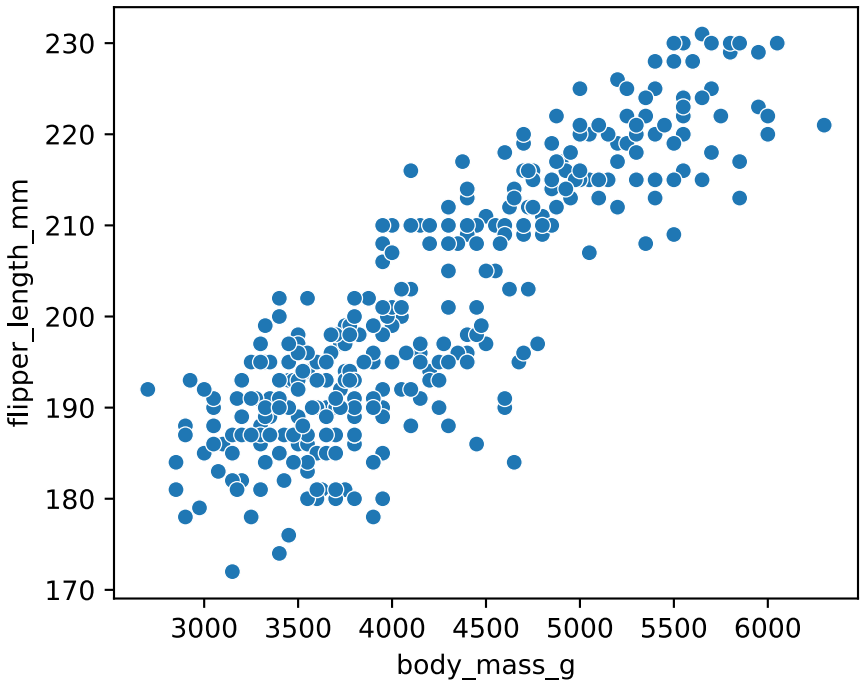
```
csv_url='https://raw.githubusercontent.com/SchmidtPaul/ExampleData/main/
palmer_penguins/palmer_penguins.csv'
df=pd.read_csv(csv_url)

# Konvertiere alle 'object'-Spalten in 'category'
for col in df.select_dtypes(include='object').columns:
    df[col] = df[col].astype('category')
```

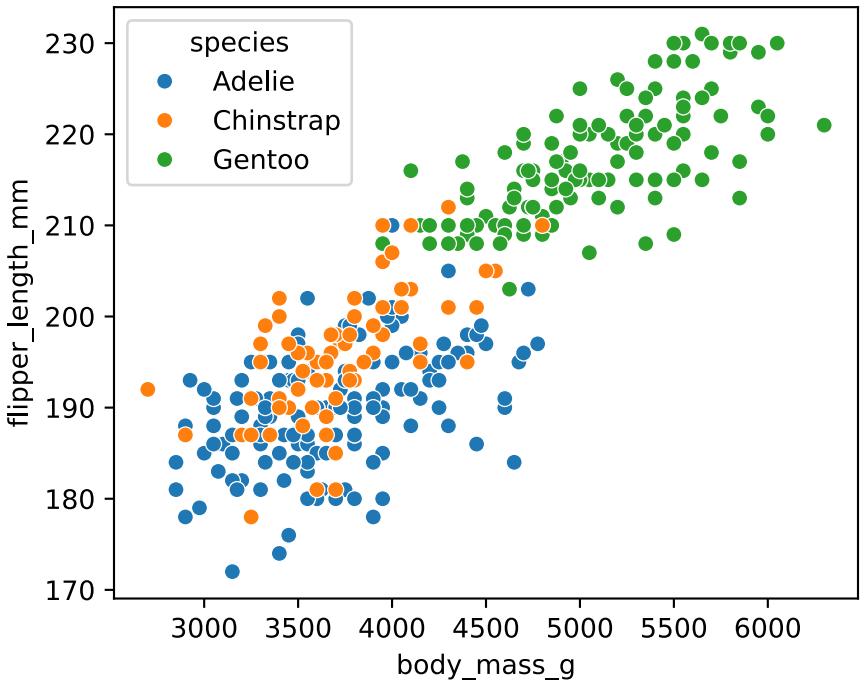
Hier nochmal die Darstellung der Pinguine nach Körpermasse und Flipperlänge als Scatterplot. Daneben haben wir die Pinguine nach Art eingefärbt.

```
plt.figure(figsize=(5, 4))
sns.scatterplot(
  data=df,
  x='body_mass_g',
  y='flipper_length_mm'

)
plt.show()
```




```
plt.figure(figsize=(5, 4))
sns.scatterplot(
  data=df,
  x='body_mass_g',
  y='flipper_length_mm',
  hue='species'
)
plt.show()
```

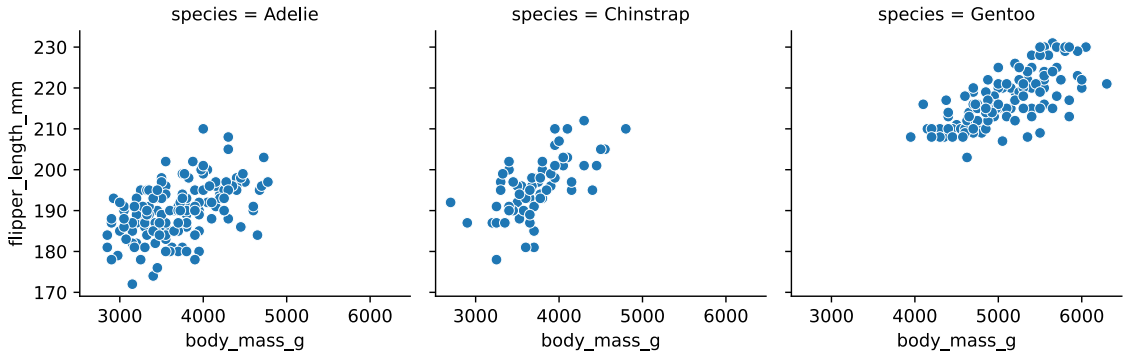


FacetGrid

Wir können die Arten aber aber eben auch durch Facetten anstatt durch Farben separieren. Es wird also ein Scatterplot für jede Art erstellt.

```
plt.figure()
g = sns.FacetGrid(data=df, col='species')
g = g.map(sns.scatterplot, 'body_mass_g', 'flipper_length_mm')
plt.show()
```

```
plt.figure()
g = sns.FacetGrid(data=df, col='species')
g = g.map_dataframe(sns.scatterplot, x='body_mass_g', y='flipper_length_mm',
hue='species')
plt.show()
```

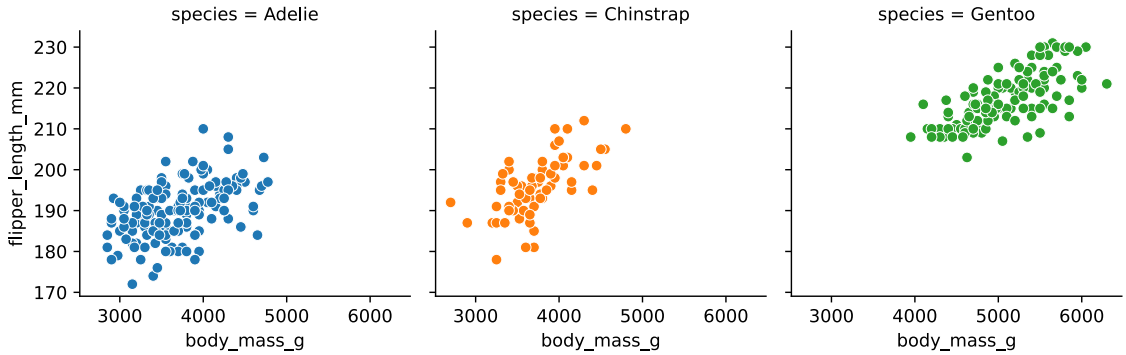


Die so erzeugte Abbildung ist intuitiv zu verstehen und über jeder Facette ist automatisch der Name der Art gelabelt. Der zugrunde liegende Code ist auch recht kurz und relativ einfach zu verstehen:

- `sns.FacetGrid` erzeugt ein leeres Raster, in das die einzelnen Facetten gezeichnet werden. Hier übergeben wir als Argumente den Datensatz `df` und den Namen der darin enthaltenen Spalte `species`, nach der die Facetten getrennt werden sollen. Wir tun dies im Argument `col`, da hier die drei Facetten nebeneinander, also quasi in Spalten (*columns*), angeordnet werden sollen. Wir könnten auch das Argument `row` verwenden, um die Facetten untereinander, also in Zeilen (*rows*), anzuordnen. Dazu gleich mehr. Schließlich speichern wir das Ergebnis, also sozusagen das noch leere Raster, in der Variable `g`.
- Im nächsten Schritt fügen wir die eigentlichen Diagramme in die Facetten ein. Dazu verwenden wir die Methode `map` auf dem Objekt `g`. Als Argumente übergeben wir die Funktion, die das eigentliche Diagramm zeichnet, sowie die Namen der Spalten, die auf der x- und y-Achse dargestellt werden sollen. In unserem Fall ist das die Funktion `sns.scatterplot` und die Spalten `body_mass_g` und `flipper_length_mm`. Man kann also sehen, dass innerhalb von `g.map()` hier dieselben Informationen übergeben werden wie im ersten Scatterplot oben.

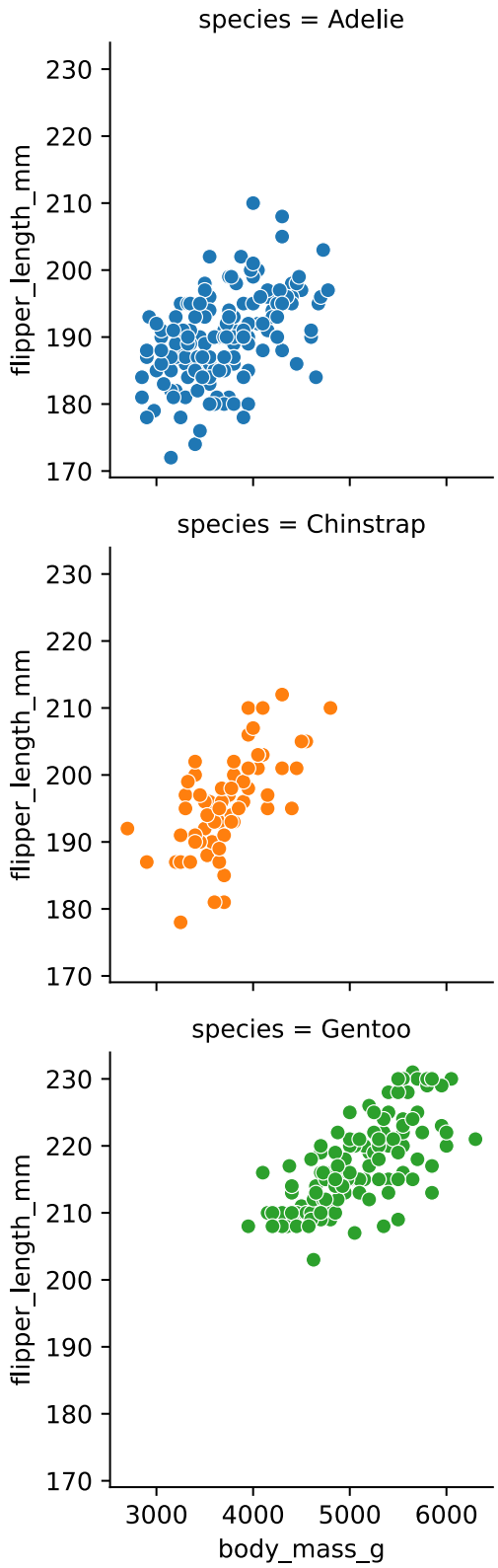
Wir können übrigens auch Facetten und Farben gleichzeitig nutzen, also trotz separater Facetten auch unterschiedliche Farben je Art nutzen. Das ist zwar redundant, sieht aber ggf. trotzdem etwas besser aus.

```
plt.figure()
g = sns.FacetGrid(data=df, col='species', hue='species')
g = g.map(sns.scatterplot, 'body_mass_g', 'flipper_length_mm')
plt.show()
```



Wie schon erwähnt, können wir die Facetten auch in Zeilen anordnen. Dazu ändern wir einfach das Argument `col` in `row`.

```
plt.figure()
g = sns.FacetGrid(data=df, row='species', hue='species')
g = g.map(sns.scatterplot, 'body_mass_g', 'flipper_length_mm')
plt.show()
```

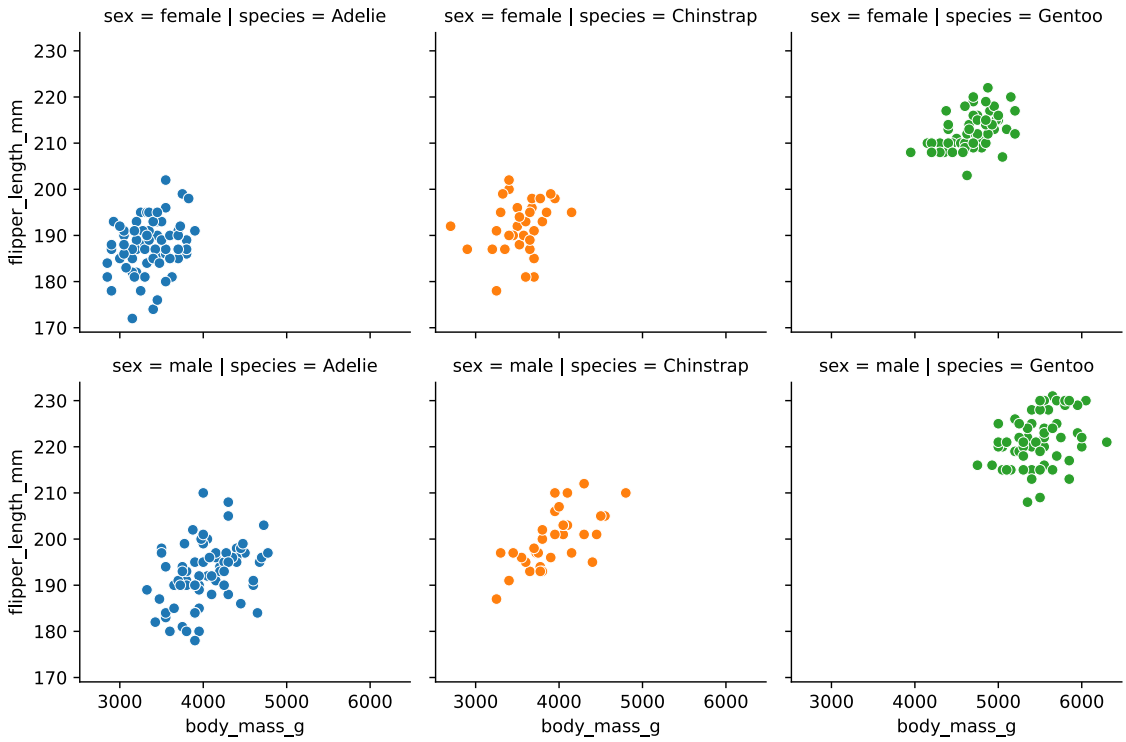

species = Adelie
species = Chinstrap
species = Gentoo
flipper_length_mm
body_mass_g
170
180
190
200
210
220
230
3000
4000
5000
6000

6

zwei Dimensionen

Die Steigerung ist dann, wenn wir sowohl Zeilen als auch Spalten nutzen. Dazu übergeben wir einfach sowohl `col` als auch `row`. Hier eignet sich beispielsweise die Spalte `sex` als zusätzliche Dimension.

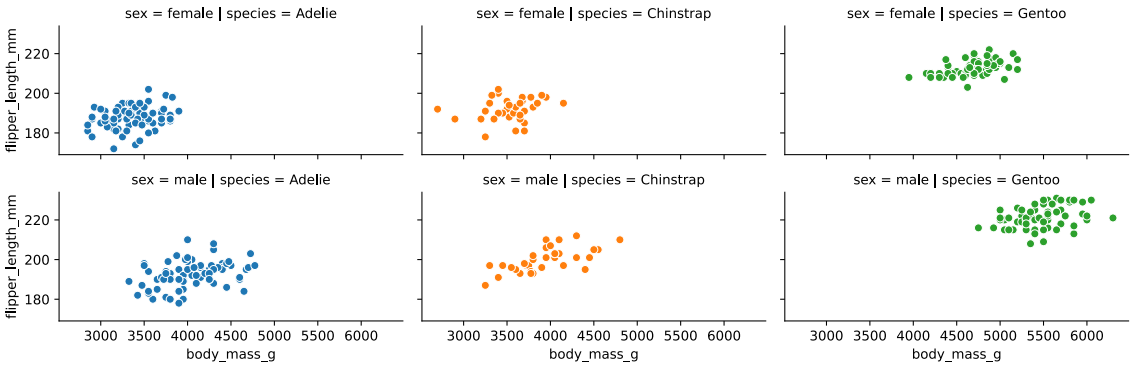
```
plt.figure()
g = sns.FacetGrid(data=df, row='sex', col='species', hue='species')
g = g.map(sns.scatterplot, 'body_mass_g', 'flipper_length_mm')
plt.show()
```



Größe ändern

Ein weiterer Unterschied bei der Arbeit mit solchen Facetten ist, dass wir die Größe der Diagramme nicht mehr direkt über `plt.figure(figsize=(...))` ändern können. Tatsächlich hat das Argument `figsize` keine Auswirkung auf den resultierenden Plot. Stattdessen müssen wir die Größe über die Argumente `height` und `aspect` in `FacetGrid` ändern. `height` gibt dabei die Höhe der einzelnen Facetten an, `aspect` das Verhältnis von Breite zu Höhe. Ein Wert von 1 bedeutet also, dass die Facetten quadratisch sind. Geben wir also beispielsweise `height=2` und `aspect=2` an, so sind die Facetten doppelt so hoch wie breit. Außerdem ist jede Facette 2 Zoll hoch, sodass die gesamte Abbildung 4 Zoll hoch ist (ggf. zuzüglich Platz für Titel usw.).

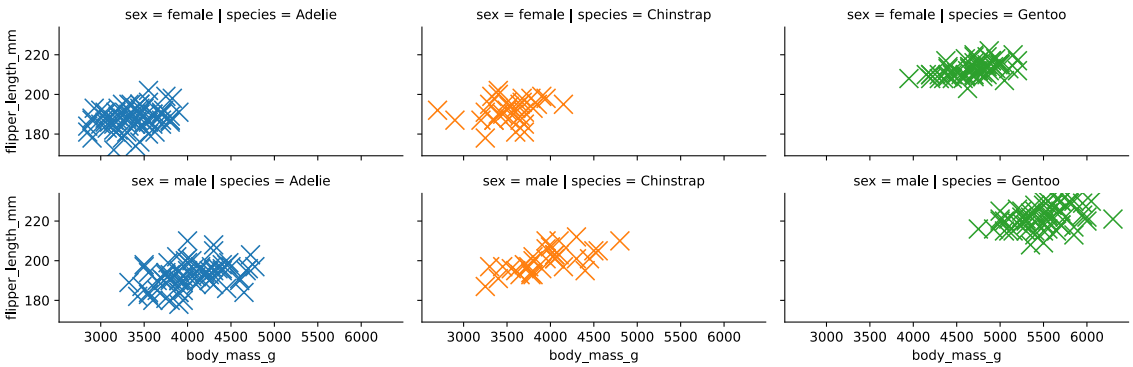
```
plt.figure()
g = sns.FacetGrid(data=df, row='sex', col='species', hue='species', height=2,
aspect=2)
g = g.map(sns.scatterplot, 'body_mass_g', 'flipper_length_mm')
plt.show()
```



.map_dataframe()

Die bisherige Methode `map()` ist die gängigste Methode, um ein Diagramm in die Facetten einzufügen. Es gibt aber auch noch die Methode `map_dataframe()`, die im Prinzip das gleiche macht, aber etwas mehr Flexibilität bietet. Sie kann gleichzeitig auch als intuitiver angesehen werden, da man explizit Argumentnamen wie `x`, `y` aber auch `color`, `marker` usw. übergeben kann. Das ist insbesondere dann hilfreich, wenn die Funktion, die das Diagramm zeichnet (also z.B. `sns.scatterplot`), viele Argumente hat.

```
plt.figure()
g = sns.FacetGrid(data=df, row='sex', col='species', hue='species', height=2,
aspect=2)
g = g.map_dataframe(sns.scatterplot, x='body_mass_g', y='flipper_length_mm',
marker='x', s=200)
plt.show()
```



Weitere Ressourcen

- Seaborn FacetGrid | How to make Small Multiples with Python Seaborn | Titles, Hue, Legend **bis 11:52**
- Plotting on a large number of facets
- Facetting histograms by subsets of data

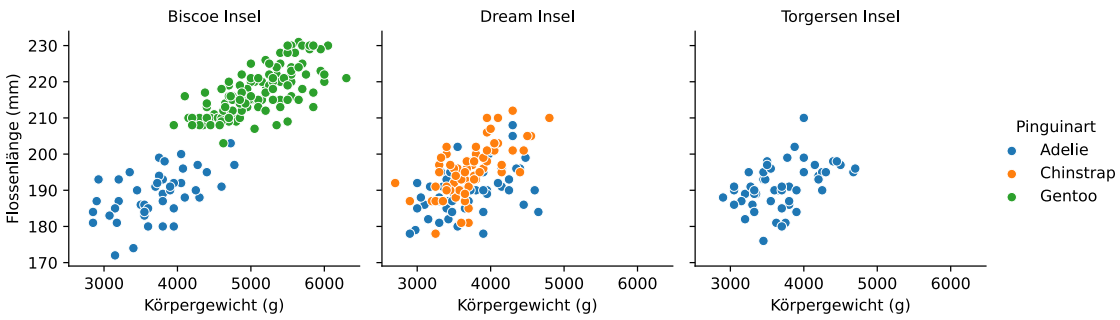
!Optional weil über Kursinhalt hinaus! - Aus Kapitel “4. Visualization with Matplotlib” des frei verfügbaren Buchs “Python Data Science Handbook”

- Abschnitt “8 Multiple Subplots” - Dieses Kapitel beschäftigt sich u.a. mit Facetten in Matplotlib, also ohne Seaborn und demnach etwas umständlicher.

Übungen

Ersetze in folgendem Code die Fragezeichen so, dass die darunter angezeigte Abbildung erzeugt wird.

```
plt.figure()
g = sns.???(data=df, col=???, hue=???)
g = g.???(sns.???, y=???, x=???)
g = g.add_legend(title=???)
g = g.set_axis_labels(???, ???)
g = g.set_titles('{col_name} Insel')
plt.show()
```



- (A) Geschafft

Nutze vorerst `pd.cut(???, ???)` um eine Spalte namens `body_mass_bin` zu erzeugen, die die Werte der Spalte `body_mass_g` in drei gleich große Intervalle unterteilt. Erstelle dann mit `sns.FacetGrid` (wieder ein Scatterplot mit `body_mass_g` auf x- und `flipper_length_mm` auf y-Achse) mit je einer Facette pro erzeugtem Intervall. Noch bevor du die Abbildung erzeugst, überlege dir, wie du die Facetten aussehen werden - insbesondere auf welchen Abschnitten der x-Achse Werte liegen werden.

- (A) Geschafft