Einführung in die Statistik mit Python

RS-eco

Biodiversity & Global Change Lab
Technische Universität München

rs-eco@posteo.de

26. Oktober 2019

Unstatistik des Monats

Immer mehr Nitrat im Grundwasser

- Laut Rheinischer Post nahm der mittlere Nitratgehalt an den 15 am stärksten belasteten deutschen Messpunkten von 2013 bis 2017 um rund 40 Milligramm pro Liter zu.

Immer mehr Nitrat im Grundwasser

- Laut Rheinischer Post nahm der mittlere Nitratgehalt an den 15 am stärksten belasteten deutschen Messpunkten von 2013 bis 2017 um rund 40 Milligramm pro Liter zu.
- Daraus folgt aber nicht, wie fälschlicherweise behauptet, dass die Nitratbelastung insgesamt gestiegen ist. Sie ist vermutlich, weiter gefallen.
- Problem 1: Die Messpunkte sind nicht die gleichen – 2017 wurden verschiedene Messpunkte von 2013 ausgetauscht (mit Bedacht in Hochbelastungsregionen verlegt)
- Problem 2: 2013 gingen die Jahresdurchschnittswerte, im Jahr 2017 aber die Höchstwerte (über ausgewählte Tage) in die Analyse ein.

Gegen einen Austausch der Messstationen wäre nichts einzuwenden, wenn man an der zeitlichen Entwicklung der Maxima interessiert wäre. Zum Betrug wird diese Vorgangsweise, wenn man das Ergebnis als landesweiten Durchschnitt verkauft.

Wiederholung

Wiederholung

- Was habt ihr vom gestrigen Tag mitgenommen?
  - abhängige vs. unabhängige Variablen
  - nominal, ordinale und metrische Daten
  - stetige vs. diskrete Daten
  - häufbare vs. nicht-häufbare Daten
  - Zahlen, Zeichenketten, Listen, ...
- Habt ihr noch Fragen zum gestrigen Tag?

Deskriptive Statistik

Lagemaßzahlen

- **Arithmetischer Mittelwert** = die Summe der gegebenen Werte geteilt durch die Anzahl der Werte (Mittel, Durschnitt)
- **Geometrisches Mittel** = ein Mittelwert, der die zentrale Tendenz oder den typischen Wert einer Reihe von Zahlen unter Verwendung des Produkts ihrer Werte angibt. Dazu werden die Zahlen miteinander Multipliziert und die n-te Wurzel gezogen, wobei n der Anzahl der zu mittelnden Zahlen entspricht. Das geometrische Mittel ist stets kleiner oder maximal gleich dem arithmetischen Mittel. Verwendung im Zusammenhang mit Wachstumsfaktoren von Beständen.
- **Harmonisches Mittel** = findet Verwendung, wenn die Zahlen im Bezug auf eine Einheit definiert sind. Dazu wird die Anzahl der Werte durch die Summe der Kehrwerte der Zahlen geteilt. Verwendung bei Durchschnitt von Raten oder der Berechnung von mittleren Geschwindigkeiten.

Lagemaßzahlen

- **Median** = derjenige Messwert, der genau „in der Mitte“ steht, wenn man die Messwerte der Größe nach sortiert (Zentralwert).
- Kann auf folgende Weise bestimmt werden:
  1. Alle Werte werden (aufsteigend) geordnet.
  2. Wenn die Anzahl der Werte ungerade ist, ist die mittlere Zahl der Median.
  3. Wenn die Anzahl der Werte gerade ist, wird der Median meist als arithmetisches Mittel der beiden mittleren Zahlen definiert.

Lagemaßzahlen

- **Median** = derjenige Messwert, der genau „in der Mitte“ steht, wenn man die Messwerte der Größe nach sortiert (Zentralwert).
- **Modus** = der häufigste Wert, der in einer Stichprobe vorkommt (Modalwert), kann auf nominal, ordinal und kardinal Daten angewendet werden.
- **Quantile** = teilt die Stichprobe so, dass ein Anteil der Stichprobe von p kleiner als das empirische p-Quantil ist und ein Anteil von 1-p der Stichprobe größer als das empirische p-Quantil ist
  - **Quartile** = die beiden Quantile mit p = 0,25 (untere) und p = 0,75 (obere). Zwischen oberem und unterem Quartil liegt die Hälfte der Stichprobe, unterhalb des unteren Quartils und oberhalb des oberen Quartils jeweils ein Viertel der Stichprobe.
  - **Perzentile** = die Quantile von 0,01 bis 0,99 in Schritten von 0,01

Stamm-Blatt-Diagramm

- dient der Visualisierung von Häufigkeitsverteilungen
- Das Diagramm besteht normalerweise aus zwei Spalten. Die linke Spalte enthält als „Stämme“ die Äquivalenzklassen, in die die auf der rechten Seite als „Blätter“ dargestellten Merkmale eingeteilt werden.

Male		**Female**
5, 2, 0	1	5, 8
5, 1	2	1, 6, 9, 9
5, 5, 5, 3, 1	3	
5, 2	4	1, 2, 6, 8
9, 8, 6, 1, 1	5	5
6, 5, 5, 0	6	0, 1
2, 1, 1, 0, 0	7	2

Stamm-Blatt-Diagramm

- dient der Visualisierung von Häufigkeitsverteilungen
- Das Diagramm besteht normalerweise aus zwei Spalten. Die linke Spalte enthält als „Stämme“ die Äquivalenzklassen, in die die auf der rechten Seite als „Blätter“ dargestellten Merkmale eingeteilt werden.
- Typisch ist eine Klassenbildung nach dem Dezimalsystem, aber auch andere Unterteilungen sind möglich, zum Beispiel die ersten beiden Ziffern als Stamm zu wählen.
- Aus einem Stamm-Blatt-Diagramm lassen sich statistische Kennzahlen wie Modalwert, Median und Quantile ablesen.
- Stößt jedoch bei einer großen Zahl von Merkmalen an seine Grenzen

Extremwerte & Spannweite

- **Minimum** = der kleinste Messwert
- **Maximum** = der größte Messwert
- **Spannweite (Range)** = Distanz zwischen dem größten und dem kleinsten Messwert

Streuungsmaßzahlen um das arithmetische Mittel

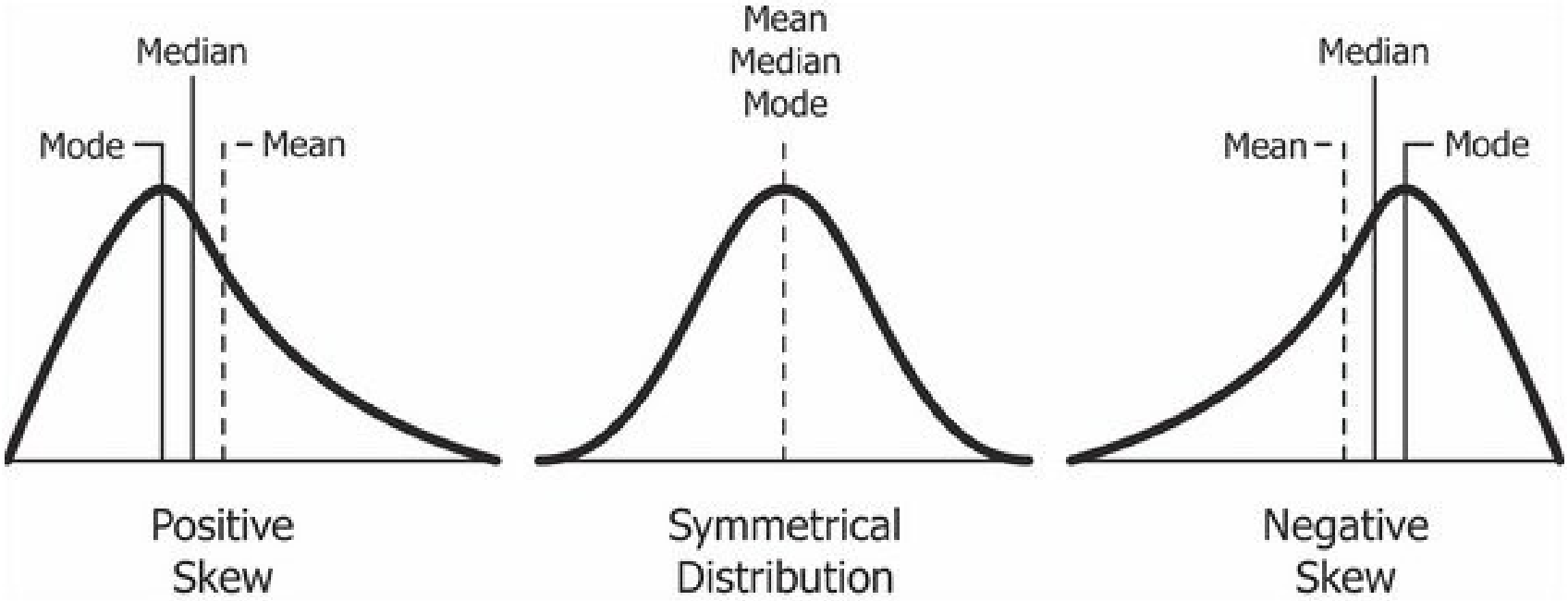
- **Summer der Abweichungsquadrate (SS)** = die Summe der quadratischen Abweichungen der Messwerte von ihrem arithmetischen Mittel
- **Varianz** = Summe der Abweichungsquadrate geteilt durch die Anzahl der Messwerte
- **Standardabweichung (SD)** = die positive Wurzel aus der empirischen Varianz
- **Variationskoeffizient** = Quotient aus Standardabweichung und arithmetischem Mittel

Streuungsmaßzahlen um den Median

- **Quantilsabstand** = Differenz zwischen p- und (1−p)-Quantil
- **Interquartilsabstand (IQR)** = Breite des Intervalls, in dem die mittleren 50 % der Stichprobeelemente liegen
- **Mittlere absolute Abweichung vom Median (MD)** = wie weit eine Stichprobe im Durchschnitt vom Median abweicht
- **Median der absolute Abweichung (MAD)** = der Median der Abweichung einer Stichprobe vom Median

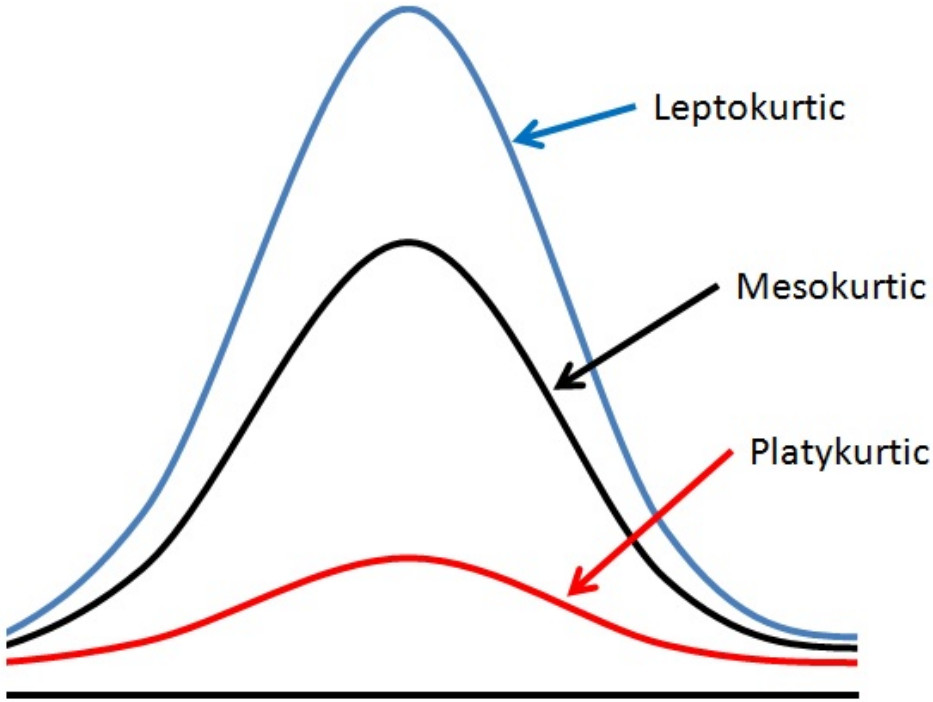
Schiefe & Wölbung (höhere Momente)

- **Schiefe (Skewness)** = Maß der Abweichung (Art und Stärke) von einer symmetrischen Verteilung
- Die „Schiefe“ gibt an, wie stark sich die Verteilung der Daten zur Seite „neigt“.
- Aus statistischer Sicht ist eine symmetrische Verteilung gewünscht (Mittelwert, Median und Modus stimmen überein)



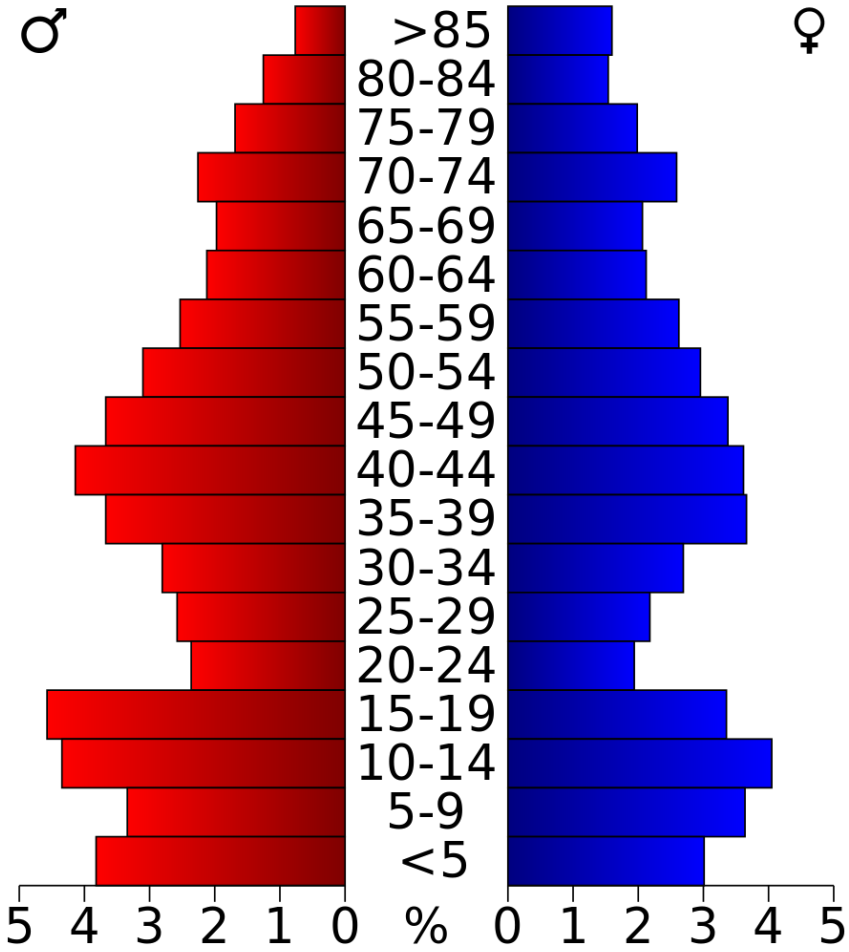
Schiefe & Wölbung (höhere Momente)

- **Wölbung (Kurtosis)** = Maß für die Steilheit bzw. Spitzigkeit einer Verteilung
- Verteilungen mit geringer Wölbung streuen relativ gleichmäßig
- bei Verteilungen mit hoher Wölbung resultiert die Streuung mehr aus extremen, aber seltenen Ereignissen



Übersichtstabellen

- **Häufigkeitsverteilung** = Angabe wie häufig jeder vorgekommene wie auch mögliche Werte vorkommt



Übersichtstabellen

- **Häufigkeitsverteilung** = Angabe wie häufig jeder vorgekommene wie auch mögliche Werte vorkommt
- **Kontingenztafel** = Tabelle, die die absolute oder relative Häufigkeit von bestimmten Merkmalsausprägungen enthält

Produkte \ *Geschlecht*	weiblich	männlich	Summe
Produkt A	660	340	1000
Produkt B	340	660	1000
Summe	1000	1000	2000

Statistische Berechnungen mit Python

Mittelwert berechnen

```
# Funktion definieren
def mittelwert(array):
  n = len(array)
  return sum(array)/n

# Liste erstellen
gehalt = [1000, 1500, 1500, 20000, 3000, 2000]

# Mittelwert berechnen
mittelwert(gehalt)

## 4833
```

Median berechnen

```
def median(array):
  array = sorted(array)
  n = len(array)
  index = int(n / 2)
  if n % 2 == 0:
    lower = array[index - 1]
    upper = array[index]
    median = 0.5 * (lower + upper)
  else:
    median = array[index]
  return median

gehalt = [1000, 1500, 1500, 20000, 3000, 2000]
median(gehalt)

## 1750.0
```

Modus berechnen

```
from collections import Counter

# Funktion definieren
def modus(array):
  counts = Counter(array)
  most_common = counts.most_common(2)
  modus = most_common[0][1]
  if modus == most_common[1][1]:
    return None
  return array[modus]

gehalt = [1000, 1500, 1500, 20000, 3000, 2000]
modus(gehalt)

## 1500
```

Quantile, Quartile and Perzentile

- Um sich Information hinsichtlich der Datenverteilung zu beschaffen, ist zudem die **quantile()**-Methode sehr praktisch.
- Ihr kann ein einzelner Wert, aber auch eine Liste (bzw. ein Tuple) mit Werten übergeben werden.

```
# Import library for iris dataset
import seaborn as sns

# Import dataset
df = sns.load_dataset('iris')

# Quantile
df.quantile([0.01, 0.25, 0.5, 0.75, 0.99])
```

Five-number summary

- Minimum, unteres Quartile (1st), Median, oberes Quartile (3rd), Maximum

```
import numpy as np

def fivenum(data):
  return np.percentile(data, [0, 25, 50, 75, 100],
                       interpolation='midpoint')

moons = [0, 0, 1, 2, 63, 61, 27, 13]

print(fivenum(moons))
## [ 0.   0.5  7.5 44.  63. ]
```

Extremwerte und Spannweite

```
# Define values
values = [4,12,43.3,19,100]

print(min(values)) # Calculate minimum

## 4

print(max(values)) # Calculate maximum

## 100
```

```
# Need to define our own function
def getrange(numbers): return max(numbers) - min(numbers)

print(getrange(values)) # Calculate range

## 96
```

Varianz berechnen

```
import math

def varianz(array):
  n = len(array)
  mn = sum(array) / n
  var = (1 / (n-1)) * sum(
    map(lambda xi: (xi-mn) ** 2 , array))
  return var

gehalt = [1000, 1500, 1500, 20000, 3000, 2000]
varianz(gehalt)
```

Standardabweichung berechnen

```
import math

def standard_abweichung(array):
  n = len(array)
  mn = sum(array) / n
  var = (1 / (n-1)) * sum(
    map(lambda xi: (xi-mn) ** 2 , array))
  std = math.sqrt(var)
  return std

gehalt = [1000, 1500, 1500, 20000, 3000, 2000]
standard_abweichung(gehalt)
```

Berechnung des Variationskoeffizienten (CV) I

```
import math
def var_koeff(array):
  n = len(array)
  mn = sum(array) / n
  var = (1 / (n-1)) * sum(map(
    lambda x: (x - mn) ** 2 , array))
  std = math.sqrt(var)
  cv = std / mn
  if 0 in array:
    cv = cv / math.sqrt((n-1))
  return cv

pizza_de = [4.99, 7.99, 5.99, 4.99, 6.99]
var_koeff(pizza_de)
```

Überblick über die Daten beschaffen

- Mit **describe()** kann man einen Überblick über die Daten erhalten.
- Die Funktion stellt diverse Lagemaße als Ergebnis bereit und kann sowohl auf ein DataFrame als auch ein Series-Objekt angewendet werden.

```
df.describe()
```

Daten zusammenfassen

```
# Read data
df = sns.load_dataset('iris')

# Dimension of the sepal length variable
len(df["sepal_length"])

# Minimum and maximum of the sepal length variable
df["sepal_length"].min(); df["sepal_length"].max()

# Mean of the sepal length variable
df["sepal_length"].mean()
# Variance of the sepal length variable
df["sepal_length"].var()
# Standard deviation of the sepal length variable
df["sepal_length"].std()
```

Daten zusammenfassen

- Außerdem lassen sich die Daten in einfacher Weise aggregieren. Hierzu kann die Methode agg() genutzt werden, die als Parameter entweder eine einzelne Funktion oder eine Liste mit Funktionen übernimmt.
- Hierbei ist es möglich, Funktionen zu verwenden, die dem DataFrame-Objekt als Methode zur Verfügung stehen. Die jeweilige Bezeichnung der Methode wird dann als Zeichenkette angegeben.

```
# Daten aggregieren, eigene Funktionen verwenden
def sqrt_sum(array):
    sum_array = sum(array)
    return np.sqrt(sum_array)

df.sepal_length.agg(["mean", "median", sum, sqrt_sum])
```

Skewness & Kurtosis

```
# Import libraries
import numpy as np
import pandas as pd

# Import kurtosis and skewness functions
from scipy.stats import kurtosis
from scipy.stats import skew

# Create data
data = np.random.normal(0, 1, 10000000)

# Calculate skewness and kurtosis
print("skew : ",skew(data))
print("kurt : ",kurtosis(data))
```

Daten als Tabelle

```
# Filtering iris data
speciesdf = df["species"]

# Group by species
speciesdf = pd.DataFrame(speciesdf.value_counts(sort=True))

# Create new column with cumulative sum
speciesdf["cum_sum"] = speciesdf["species"].cumsum()

# Create new column with relative frequency
speciesdf["cum_perc"] = 100*speciesdf["cum_sum"]/
  speciesdf["species"].sum()

# Show summary table
speciesdf
```

Daten als Tabelle

```
### Division at intervals and corresponding frequencies

# Division of the interval in 10 equal parts,
# and the interval closed on right
table = np.histogram(df["sepal_length"],
                     bins=10, range=(0, 10))

print(table)

# Division at fixed intervals,
# the buckets object indicates the limits of the intervals
# the intervals should be closed on right.
buckets = [0, 2, 4, 6, 8]
table = np.histogram(df["sepal_length"], bins=buckets)

print(table)
```

Do it yourself

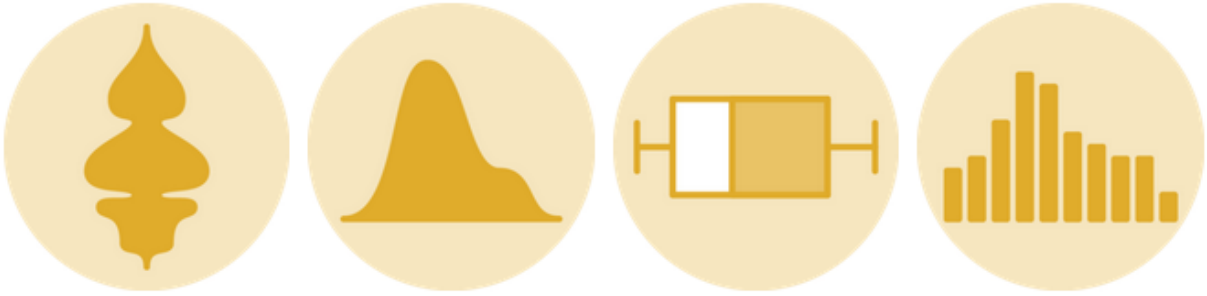
- Liest erneut den Datensatz **size.csv** in Python ein
- Berechnet die Minimal-, Durchschnitts- und Maximalgröße jeweils für Männer und für Frauen
- Wie viele Dateneinträge gibt es jeweils für Männer und für Frauen (Kontingenztafel)
- Berechnet Minimum, unteres Quartil, Median, oberes Quartile und Maximum der Schuhgrößen
- Gibt es eine stärkere Varianz und Standardabweichung in der Schuhgröße von Männern oder von Frauen?

Datenvisualisierung

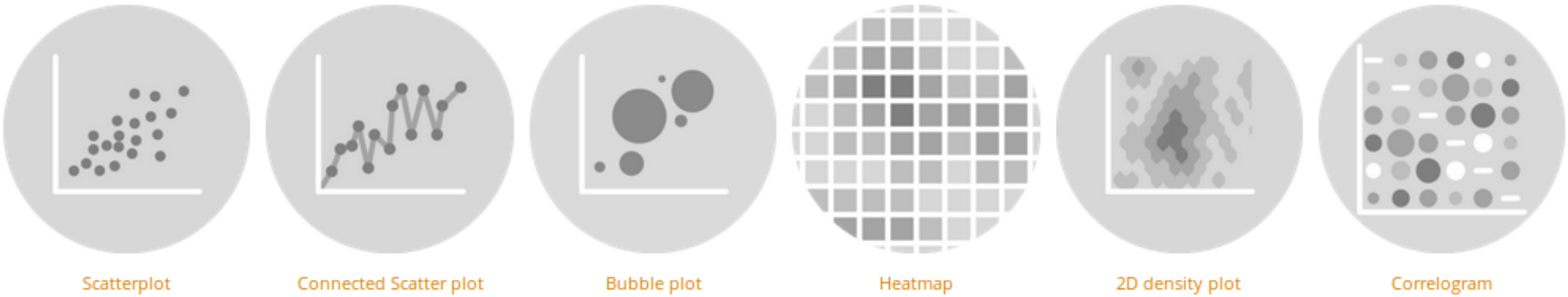
Datenvisualisierung - Eine Kunst für sich

Verteilung & Zusammenhang

DISTRIBUTION



CORRELATION



Rangliste

RANKING

Teil eines Ganzen & Entwicklung

PART OF A WHOLE


Stacked barplot


Tree plot


Venn diagram


Doughnut plot


Pie plot


Tree diagram

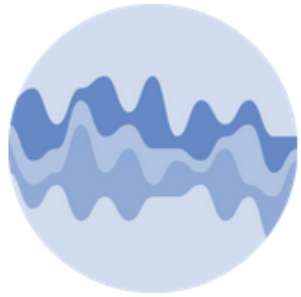
EVOLUTION


Line plot


Area plot


Stacked area plot


Parrallel plot


Streamchart

Karten & Fluss

MAPS

Map

Choropleth map

Connection map

Bubble map

FLOW

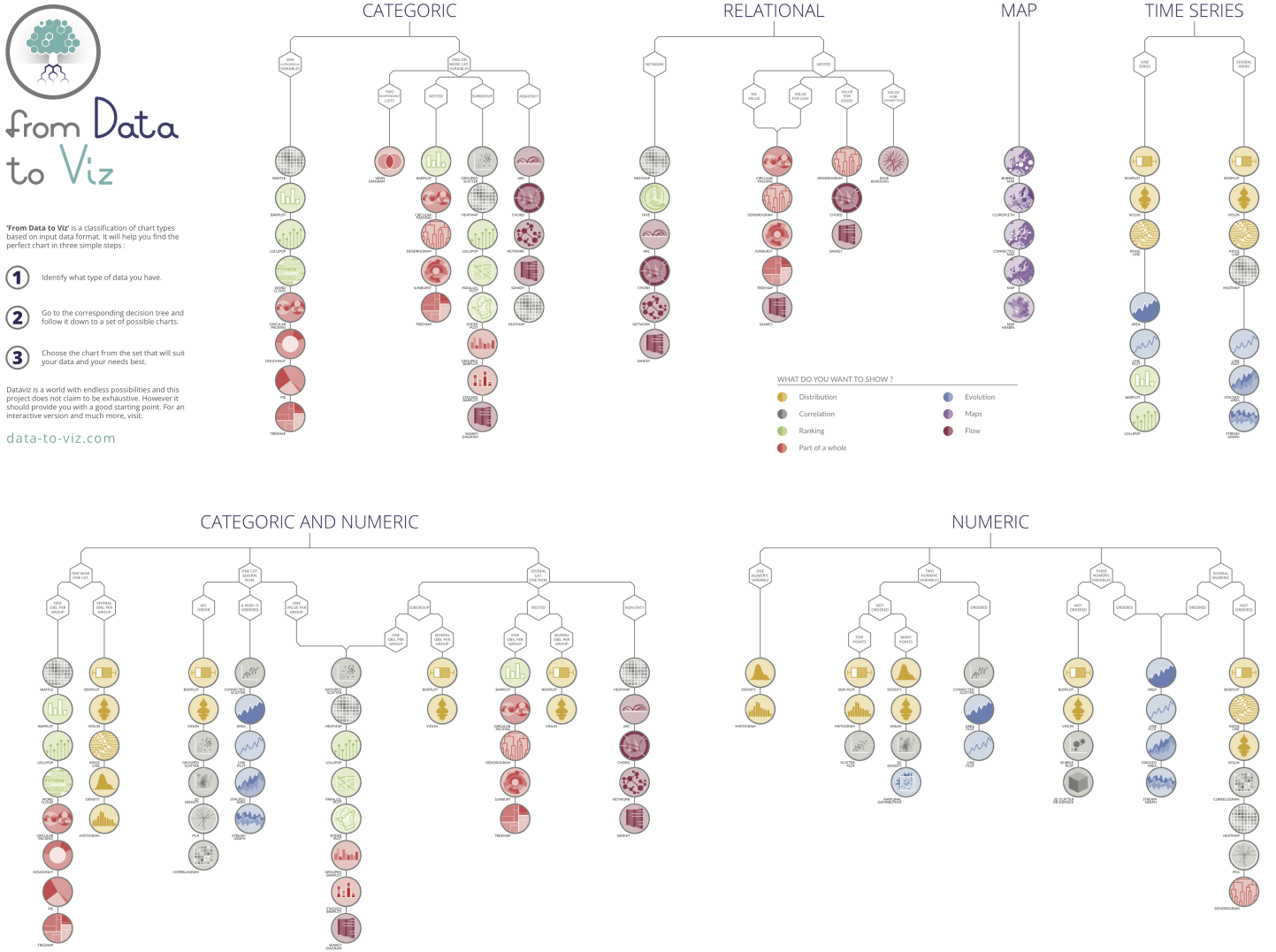
Chord diagram

Network chart

Sankey diagram

Von den Daten zur Visualisierung

Es ist ganz einfach die passende Visualisierung zu deinen Daten zu finden.

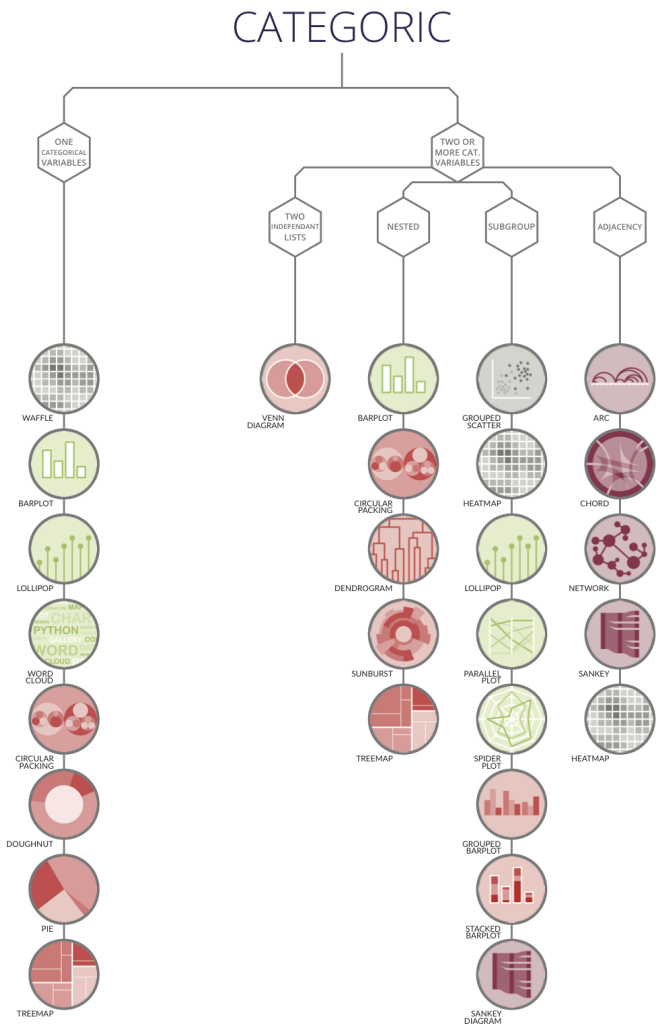


https://www.data-to-viz.com/

Von den Daten zur Visualisierung

Es ist ganz einfach die passende Visualisierung zu deinen Daten zu finden.

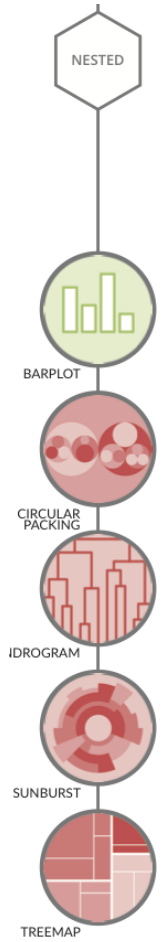
1. Identifiziere den vorliegenden Datentyp, z.B. kategorisch



Von den Daten zur Visualisierung

Es ist ganz einfach die passende Visualisierung zu deinen Daten zu finden.

1. Identifiziere den vorliegenden Datentyp, z.B. kategorisch
2. Wähle den passenden Entscheidungsschlüssel und folge ihm zu einer Auswahl an Abbildungen.



Von den Daten zur Visualisierung

Es ist ganz einfach die passende Visualisierung zu deinen Daten zu finden.

1. Identifiziere den vorliegenden Datentyp, z.B. kategorisch
2. Wähle den passenden Entscheidungsschlüssel und folge ihm zu einer Auswahl an Abbildungen.
3. Wähle die Abbildung die am besten zu deinen Daten und deinen Bedürfnissen passt.

Barplot

Daten visualisieren mit Python

Daten visualisieren mit Matplotlib

- Die Bibliothek Matplotlib ist gewissermaßen der Standard zur Visualisierung von Ergebnissen in Python.
- Das Paket wird konventionell wie folgt eingebunden:

```
import matplotlib.pyplot as plt
```

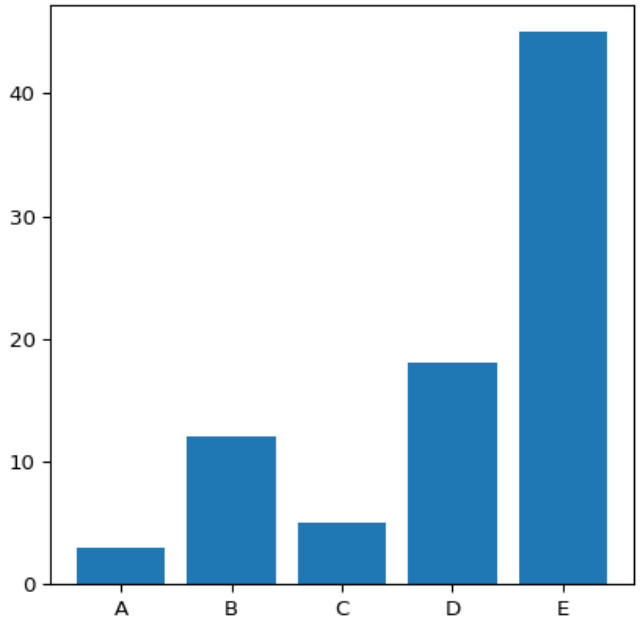
- Zusätliche verwenden wir die Bibliothek seaborn & pandas:

```
import seaborn as sns
import pandas as pd
```

- Diese Schreibweisen haben sich etabliert und sollten auch bei eigenen Projekten übernommen werden
- Die Websiten https://www.data-to-viz.com/ & https://python-graph-gallery.com/ bieten Instruktionen zum **Finden der richtigen Graphik** und stellen **Code-Beispiele für jeden Visualisierungstypen zur Verfügung**.

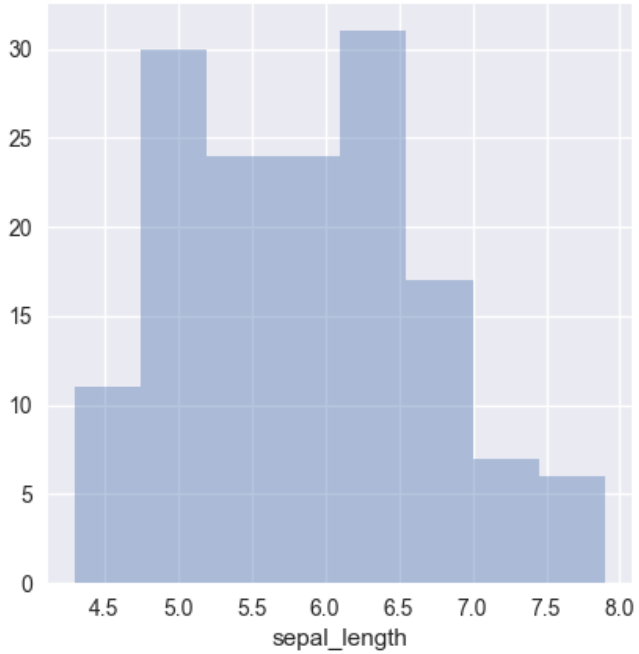
Säulendiagramm

```
import numpy as np
import matplotlib.pyplot as plt

# Make a fake dataset:
height = [3, 12, 5, 18, 45]
bars = ('A', 'B', 'C', 'D', 'E')
y_pos = np.arange(len(bars))

# Create bars
plt.bar(y_pos, height)

# Create names on the x-axis
plt.xticks(y_pos, bars)

plt.show() # Show graphic
```



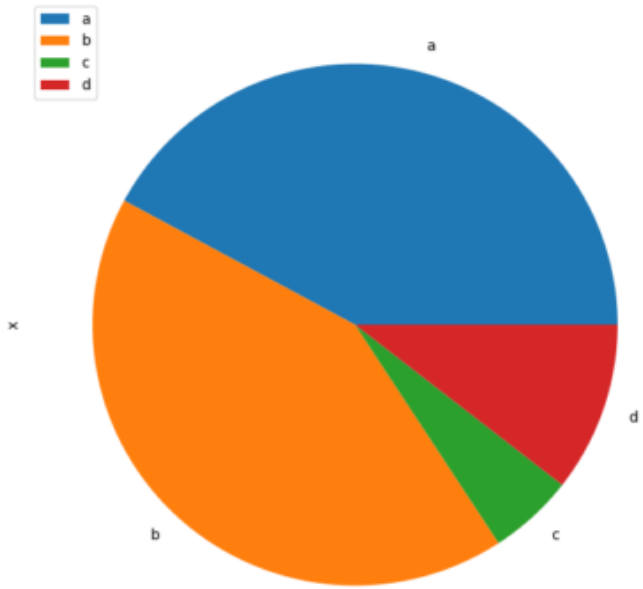
Histogramm

```
# Import library and dataset
import seaborn as sns
df = sns.load_dataset('iris')

# Hist only
sns.distplot( a=df["sepal_length"],
              hist=True, kde=False,
              rug=False )
#plt.show()
```



Kuchen-Diagramm (Pie-Diagramm)

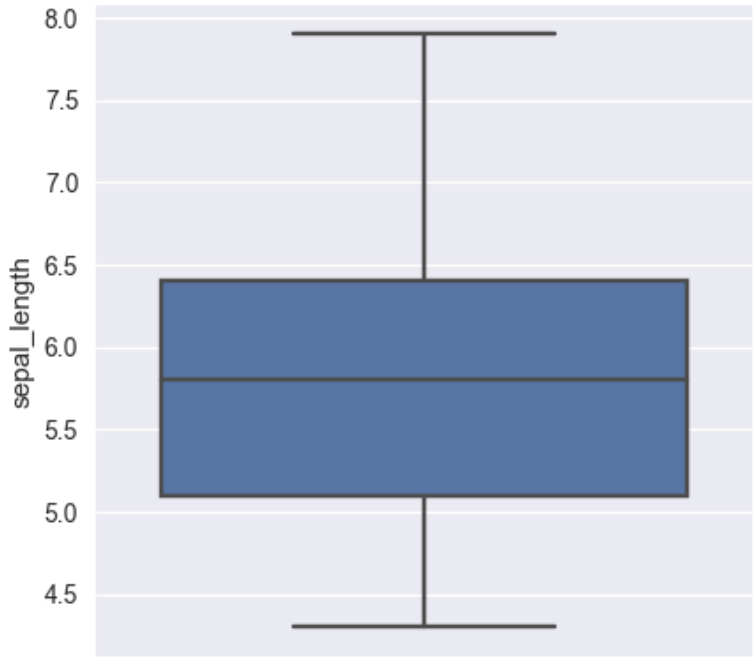
```
import pandas as pd # load Library

df = pd.DataFrame([8,8,1,2], index=['a','b','c','d'],
                  columns=['x']) # just 4 values for 4 groups

df.plot(kind='pie', subplots=True, figsize=(8, 8))
```



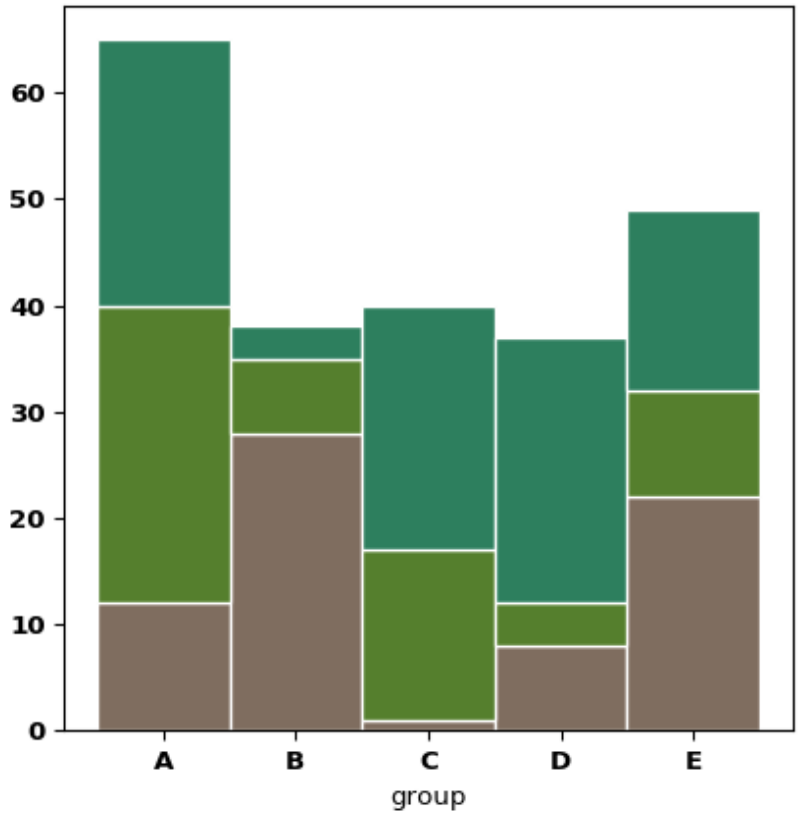
Box-Plot

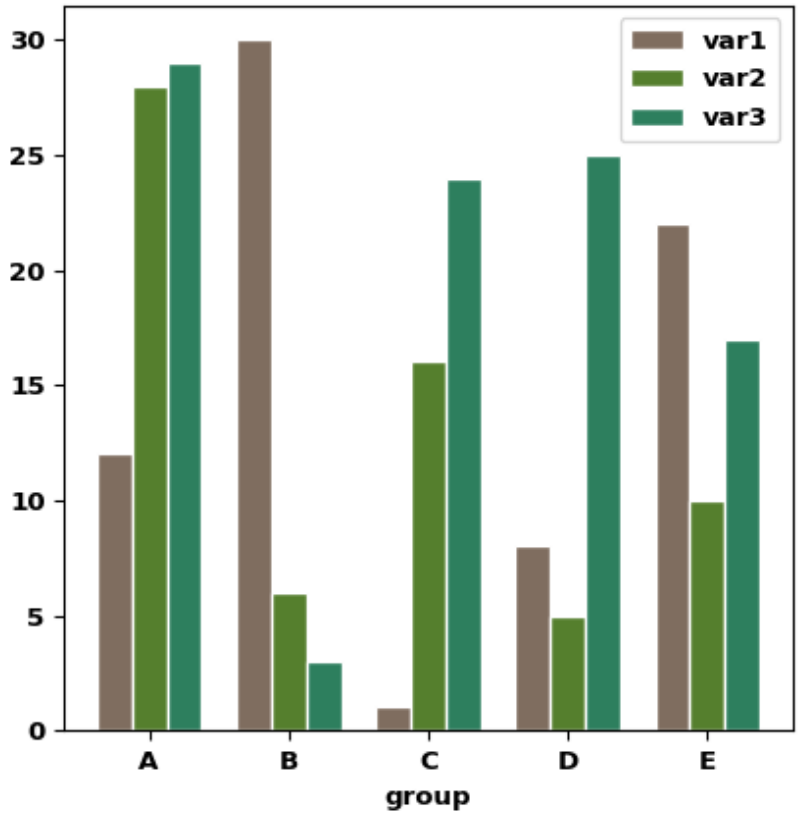
Nur eine numerische Variable

```
# library & dataset
import seaborn as sns

df = sns.load_dataset('iris')

# Make boxplot for one group only
sns.boxplot(y=df["sepal_length"])
#plt.show()
```



gestapeltes & gruppiertes Säulendiagramm

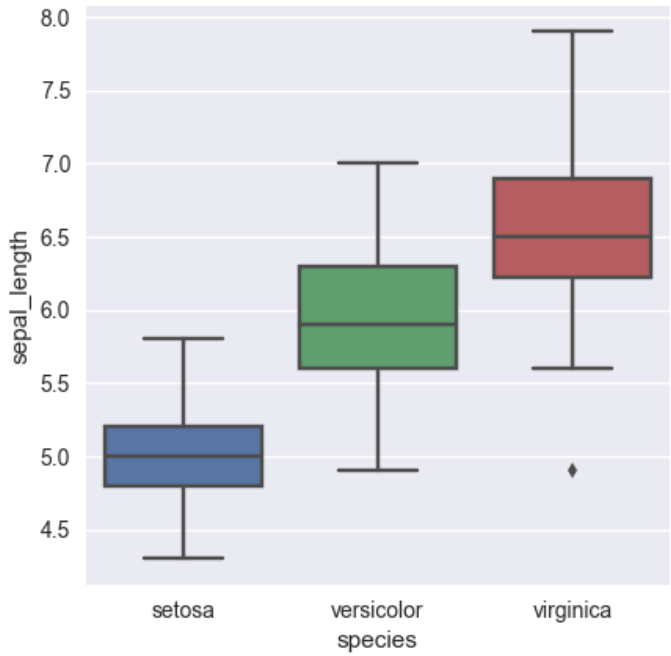




Box-Plot

Eine numerische Variable und mehrere Gruppen

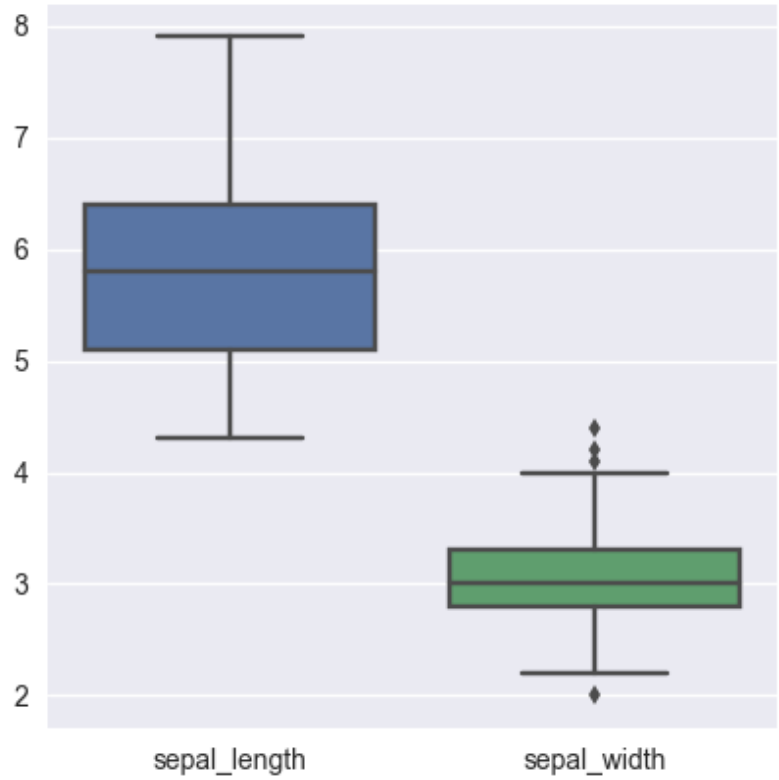
```
# library & dataset
import seaborn as sns

df = sns.load_dataset('iris')

sns.boxplot(x=df["species"],
            y=df["sepal_length"])
# plt.show()
```



Box-Plot

Mehrere numerische Variablen

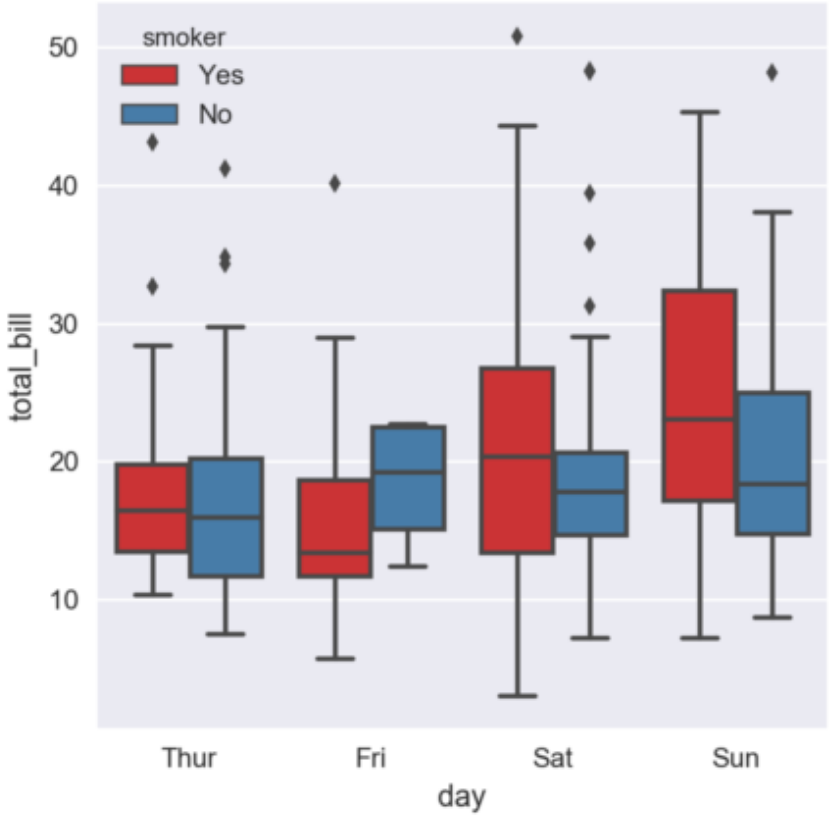
```
# library & dataset
import seaborn as sns

df = sns.load_dataset('iris')

sns.boxplot(data=df.ix[:,0:2])
# plt.show()
```



gruppierter Box-plot

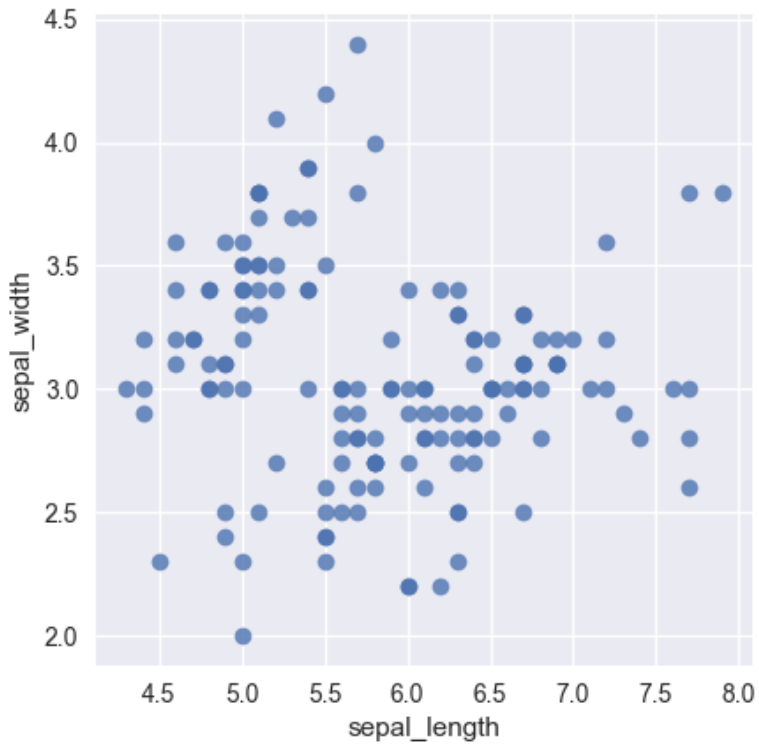
Mehrere numerische Variablen und mehrere Gruppen

```
# library and dataset
import seaborn as sns

df = sns.load_dataset('tips')

# Grouped boxplot
sns.boxplot(x="day",
            y="total_bill",
            hue="smoker",
            data=df,
            palette="Set1")
# plt.show()
```

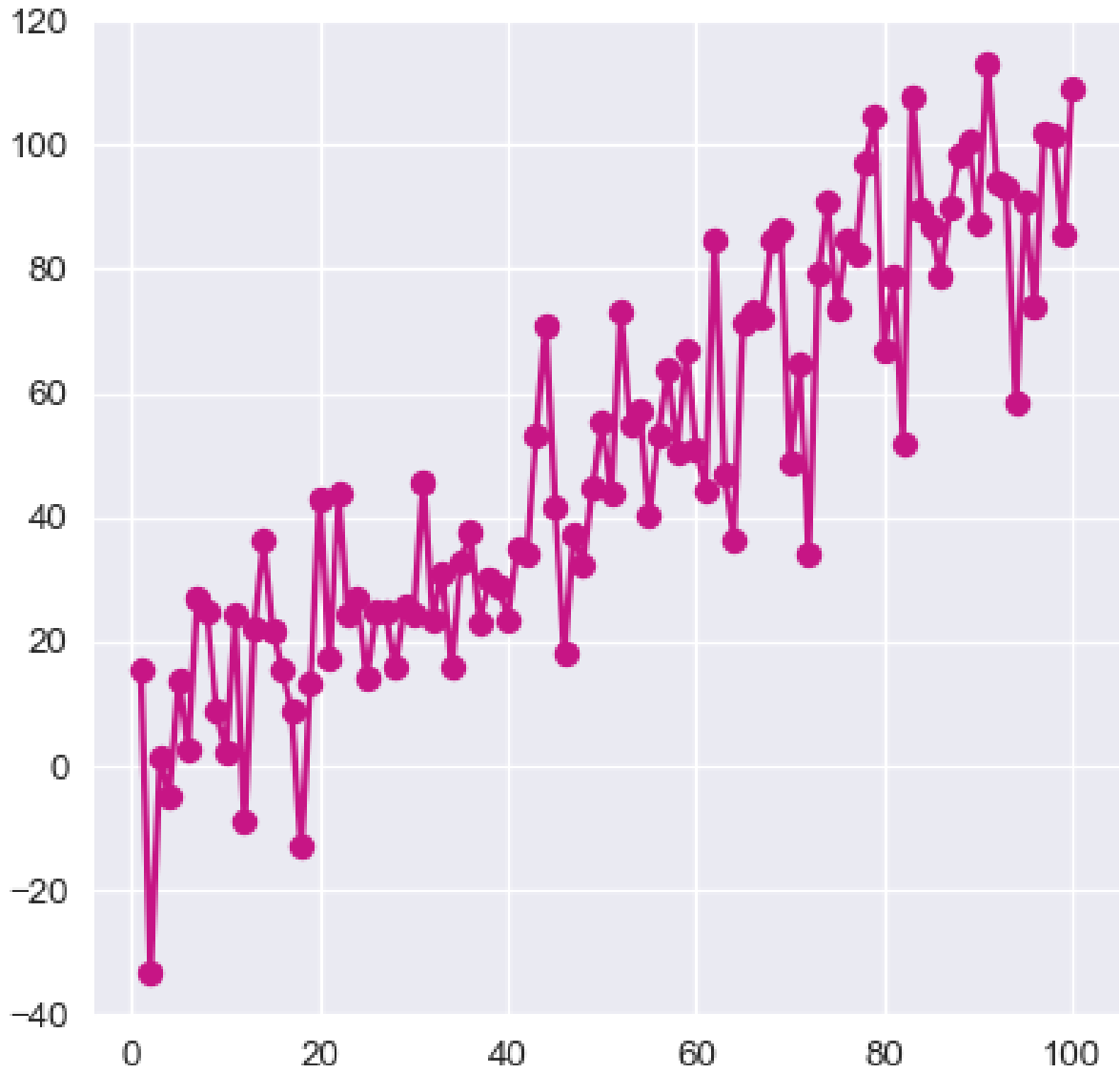


Streudiagramm (Scatterplot)

```
# library & dataset
import seaborn as sns

df = sns.load_dataset('iris')

# use the function regplot
# to make a scatterplot
sns.regplot(x=df["sepal_length"],
            y=df["sepal_width"],
            fit_reg=False)
# plt.show()
```

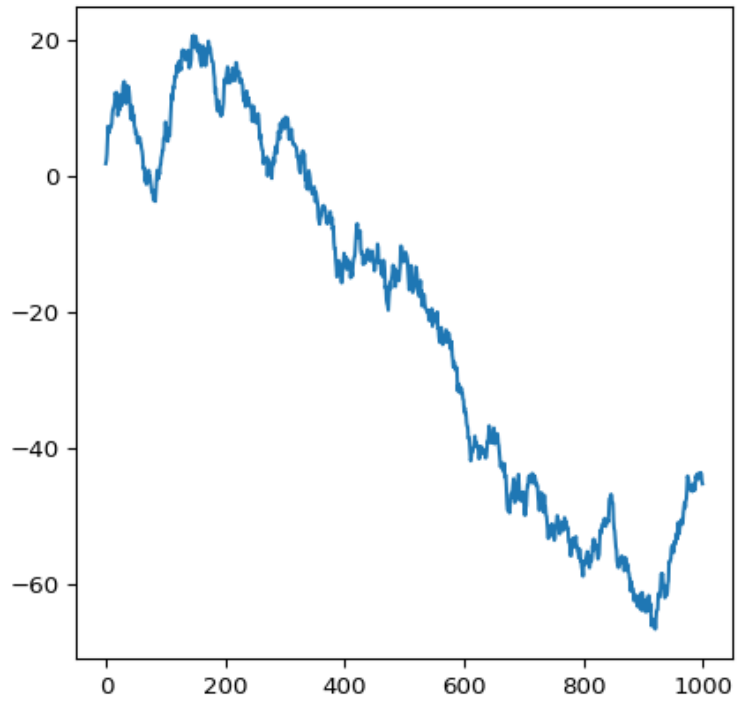


verbundenes Streudiagramm (Scatterplot)



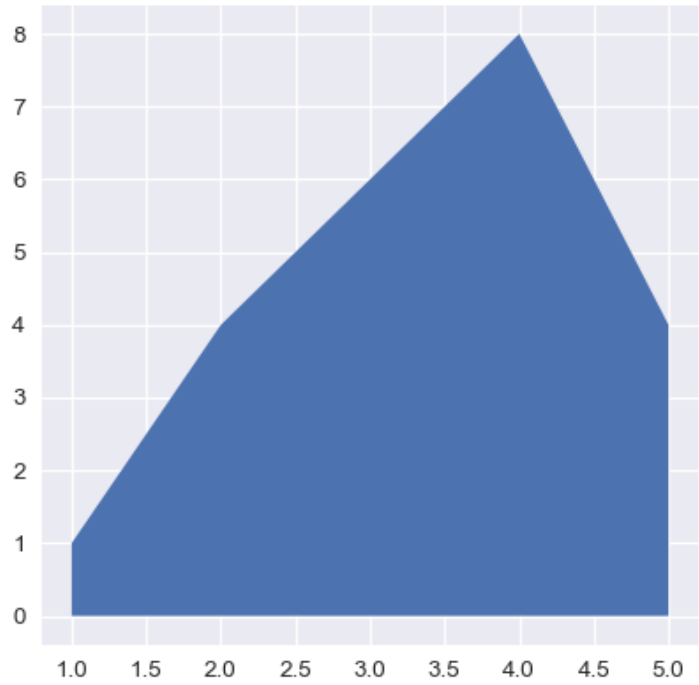
Liniendiagramm

```
# libraries
import matplotlib.pyplot as plt
import numpy as np

# create data
values=np.cumsum(
  np.random.randn(1000,1))

# use the plot function
plt.plot(values)
```



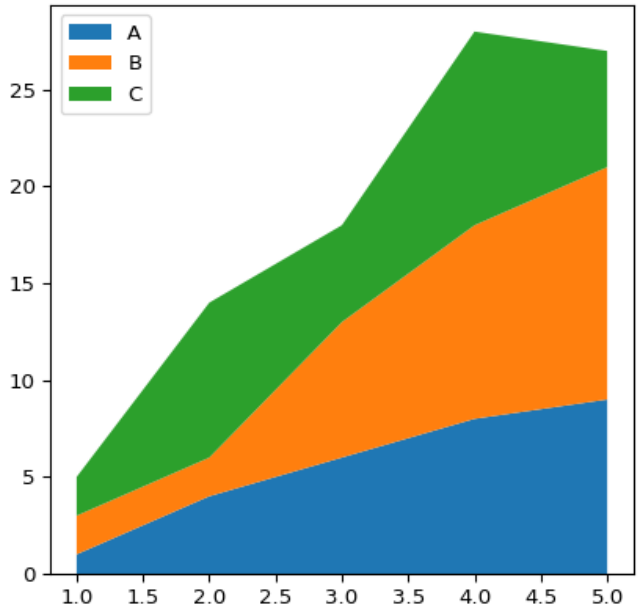
Flächendiagramm

```
# library
import numpy as np
import matplotlib.pyplot as plt

# Create data
x=range(1,6)
y=[1,4,6,8,4]

# Area plot
plt.fill_between(x, y)
plt.show()
```

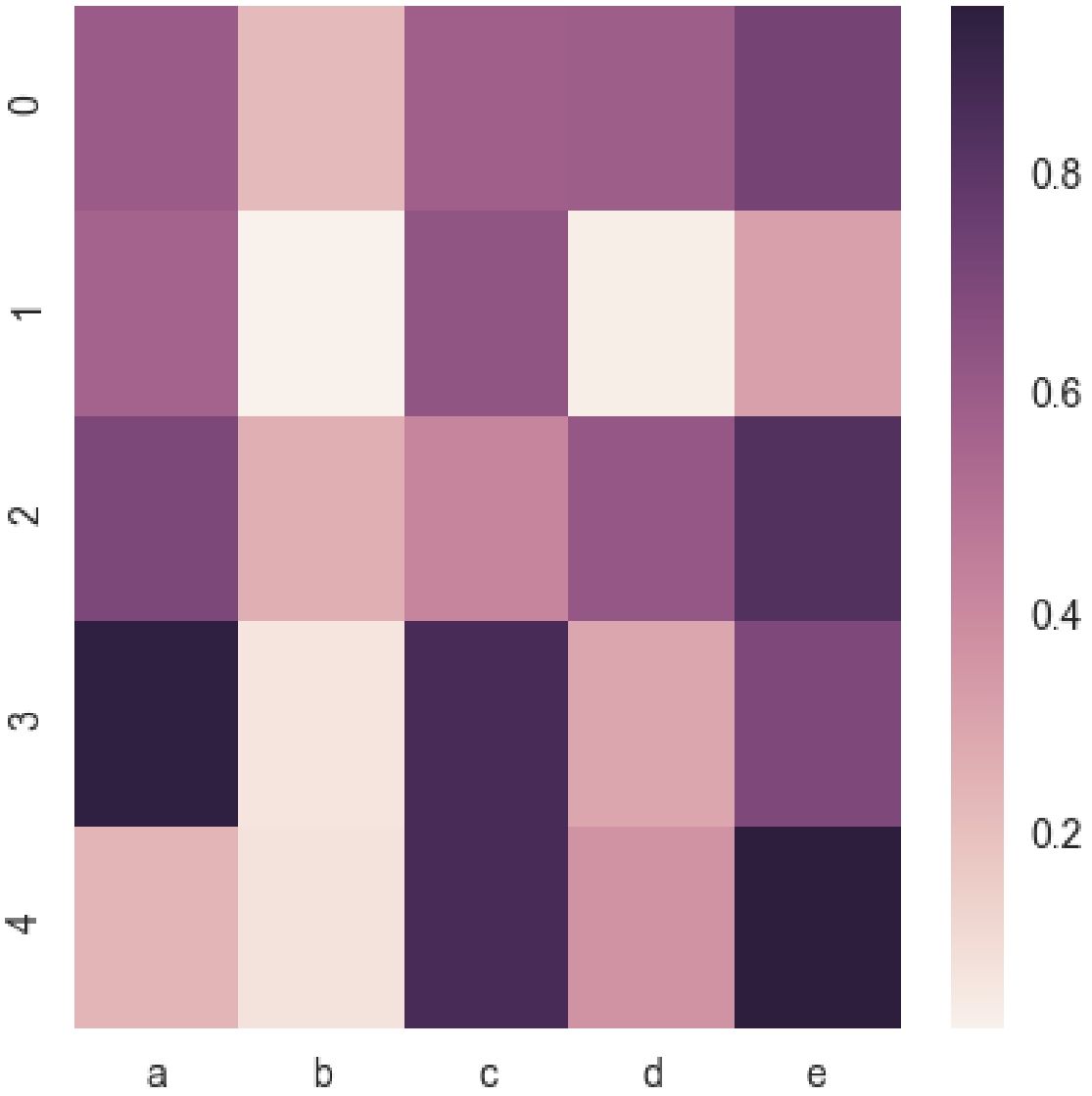


gestapeltes Flächendiagramm

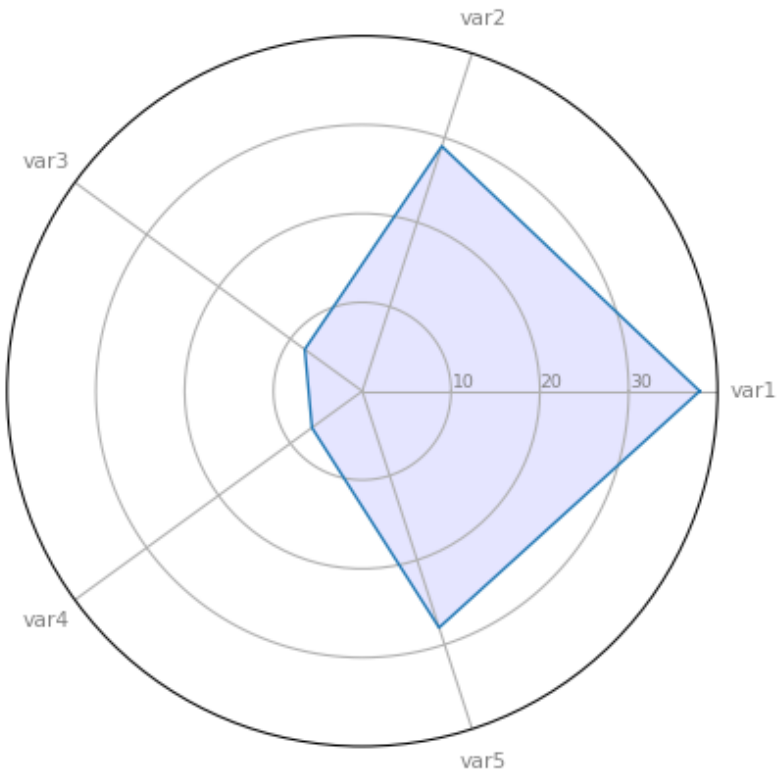
```
# library
import numpy as np
import matplotlib.pyplot as plt

# Your x and y axis
x=range(1,6)
y=[ [1,4,6,8,9],
    [2,2,7,10,12],
    [2,8,5,10,6] ]

# Basic stacked area chart.
plt.stackplot(x,y,
              labels=['A','B','C'])
plt.legend(loc='upper left')
plt.show()
```

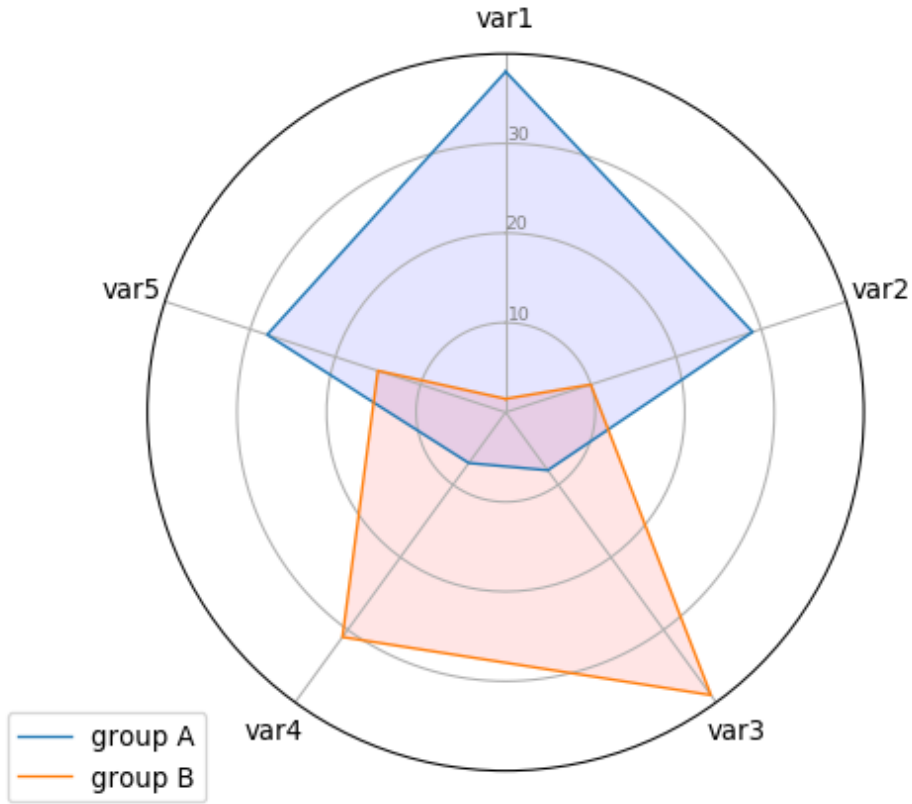


Heatmap

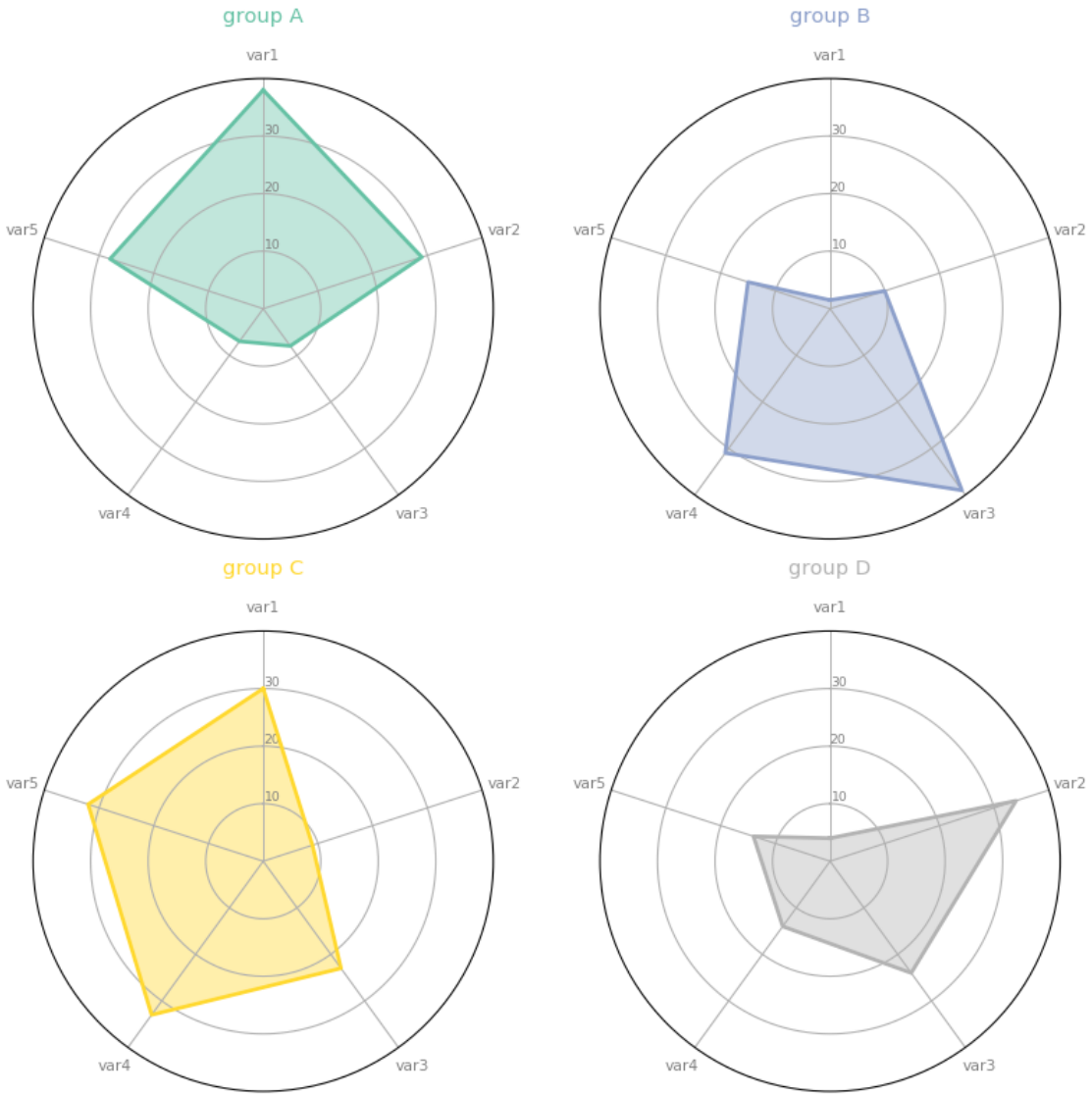


Netzdiagramm





Netzdiagramm



Grafiken speichern

Matplotlib stellt die Funktion **savefig()** bereit, mit der Plots direkt als Datei gespeichert werden können:

```
plt.savefig('plot.png')
```

Um das Datei-Format zu ändern, muss man lediglich die Datei-Endung anpassen:

```
plt.savefig('plot.pdf')
```

Die Datei lässt sich dann öffnen, durch:

```
display plot.png
```

Caveats bei der Datenvisualisierung

- Sortieren der Daten macht das Diagramm viel aufschlussreicher
- Vermeidet Spaghetti-Diagramme (Liniendiagramm mit zu vielen Linien).
- Vermeidet Tortendiagramme, da das menschliche Auge Winkel schlecht lesen kann.
- Das Ändern der Bin-Größe eines Histogramms kann zu unterschiedlichen Erkenntnissen führen.
- Boxplots kaschieren den Stichprobenumfang und dessen Verteilung.
- Vermeidet die Verwendung von Fehlerbalken.
- Vermeidet die Darstellung von zu vielen Verteilungen.
- Zu viele Punkte machen Streudiagramme unlesbar.
- Vermeidet Farben, wenn sie nichts aussagen.
- Bei gestappelten Diagrammen ist es schwierig jede dargestellte Gruppe zu analysieren.

Eine Übersicht der häufigsten Caveats findet ihr hier: https://www.data-to-viz.com/caveats.html

Do it yourself

- Liest erneut den Datensatz **size.csv** in Python ein
- Erstellt einen geeignete Grafik um die Größe darzustellen
- Plottet die Größe in Zusammenhang zum Geschlecht
- Plottet die Schuhgröße gegen die Körpergröße
- Speichert die erstellten Grafiken als Datei

Vielen Dank für eure Aufmerksamkeit!