Kurze Einführung in pandas

pandas ist eine Python Library für Datenverarbeitung, -Analyse und -Manipulation

```python
# pandas library aktivieren
import pandas as pd

# daten einlesen
df = pd.read_csv('path/to/file.csv')
df = pd.read_excel('path/to/file.xls')

# daten speichern
df.to_csv('path/to/file.csv')
df.to_excel('path/to/file.xls')

# daten gruppieren
(df
 .groupby(['sex', 'age'])
 .mean()
 .reset_index() # optional
)

# direkt abfragen
(df
 .query('sex == "M" and age > 19')
)

# mit Filtermaske selektieren
cond = (
     (df['sex'] == 'M')
   & (df['age'] > 19)
)
df[cond]
```

DataFrame

	sex	age	traveltime
0	F	18	2
1	F	17	1
2	F	15	1
3	F	15	1
4	F	16	1
...	...	...	...
644	F	19	1
645	F	18	1
646	F	18	2
647	M	17	2
648	M	18	3

.columns
Index, MultiIndex

.iloc[1]
Rows (axis=1)

['age'][3]
value

['sex']
Series (axis=0)

.index
Index/MultiIndex

```python
# long => wide format
pd.pivot_table(
    index=['col', …],
    columns=['col', …],
    values=['col', …])
# wide => long format
df.melt()
```

```python
# plotting
Die meisten pandas plots benötigen
einen DataFrame im Wide-Format

# -- bar charts
df.plot.bar(x='age', y='traveltime')
df.plot.barh(x='age', y='traveltime')

# -- pie chart
(df
    .groupby('col1')[col2].mean()
    .plot.pie()
)

# -- scatter plot
df.plot.scatter('age', 'traveltime')

# -- line plot
(df
  .sort_values(['xcol', 'ycol'])
  .plot.line('xcol, ycol')
)
```

References

https://pandas.pydata.org
https://pandas.pydata.org/Pandas_Cheat_Sheet.pdf
https://github.com/tommyod/awesome-pandas

Quelle: Autor und pandas: Data Structures for Statistical Computing in Python, Wes McKinney, 2010

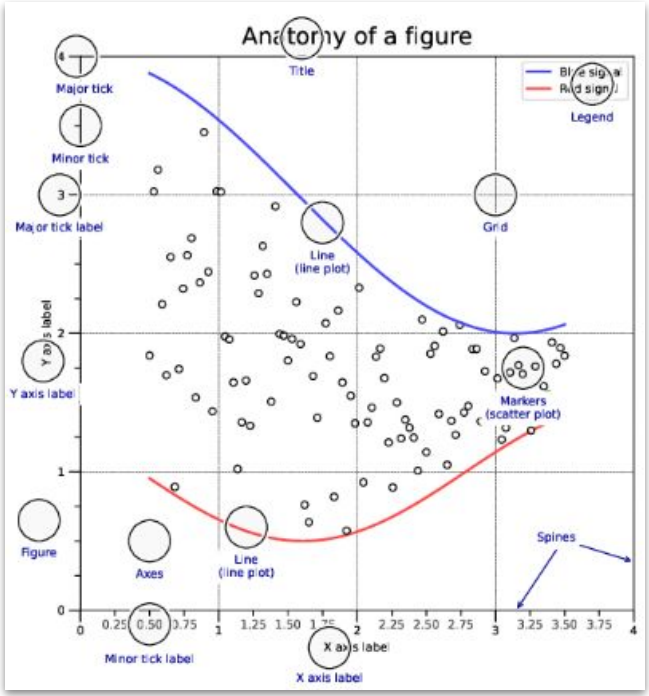
Kurze Einführung in matplotlib

matplotlib ist eine Python Library für die Visualisierung von strukturierten Daten (& mehr)

```
# library aktivieren
import matplotlib.pyplot as plt

# neues Diagramm
fig = plt.figure(figsize=(w, h))

# subplots
fig, axs = plt.subplots(rows, cols, …)

# title, super title
plt.title('title')
plt.suptitle('super title')

# legend placement
plt.legend(loc='upper left'
          , bbox_to_anchor=(x,y))

# axis labeling
plt.axis('on|off')
plt.xticks(rotation=0|45|90)
plt.yticks(rotation=0|45|90)

# axis scaling
current = plt.xlim(), plt.ylim()
plt.xscale('linear|log')
plt.yscale('linear|log')
```

 # Farbwahl erfolgt über Plot-Funktion

Matplotlib stellt verschiedene Farbpaletten zur Verfügung. Siehe Tutorials und Cheatsheets.




```
# Programmstruktur (nach Bedarf)
plt.figure()
… <plot funktion pandas/seaborn/plotly>
plt.suptitle('super title')
plt.title('title')
plt.axis(...)
plt.legend(...)
plt.yscale(...)  | ax.set_yscale(...)
plt.xticks(...)  | ax.set_xticks(...)
plt.yticks(...)  | ax.set_yticks(...)
plt.xlabel(...)  | ax.set_xlabel(...)
plt.ylabel(...)  | ax.set_ylabel(...)
plt.show()
```

References

https://matplotlib.org/stable/tutorials/index.html
=> Focus on the "Pyplot Tutorial"
https://matplotlib.org/cheatsheets/

Gut zu wissen

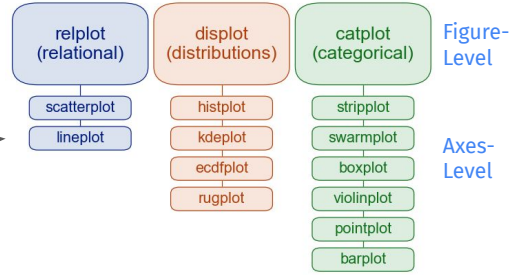
Andere Libraries wie pandas und seaborn bauen auf Matplotlib auf. Es ist daher wichtig, die **Grundbegriffe von matplotlib.pyplot** zu kennen. Den Rest von matplotlib kann man meistens ignorieren.

Quelle: Autor und matplotlib: A 2D graphics environment, Computing in Science & Engineering, J.D. Hunter, IEEE, 2007

Kurze Einführung in seaborn

seaborn ist eine Python Library für statistische Visualisierung

```
# library aktivieren
import seaborn as sns
sns.set_theme()

# API für alle Funktionen
sns.fn(data=df, x='col', y='col', …)

# figure level
# -- multiple plots
sns.relplot|displot|catplot(
    data=..., x=..., y=...,
    col='col', # plot per col value
    hue='col', # color by col value
)

# axes level
# -- a single plot
sns.scatterplot|lineplot|....(
    data=..., x=..., y=...,
    hue='col', # color by col value
)

# weitere interessante Plots
sns.heatmap()   # colored heatmap
sns.lmplot()    # lin.regression
# Lattice/Trellis plotting (by groups)
sns.pairplot()
sns.jointplot()
# Farbwahl
palette='name' # matplotlib palette
hue='col'      # Farbe automatisch
```



Plot-Typen von Seaborn

- Figure-Level erstellen ein oder mehrere Plots auf einmal, kombiniert verschiedene Plot-Typen
- Axes-Level erstellen einen bestimmten Plot-Typ

Funktionen

Die Funktionen von seaborn sind "smarter" als jene von z.B. Pandas. Oftmals ist es einfacher, mit seaborn einen bestimmten Plot-Typen zu erreichen. Mit Parametern wie hue=, row=, col= erstellt seaborn automatisch Gruppierungen und Aggregationen von Daten.
Dies bedeutet:

- Long-Format Daten sind bevorzugt, weil sie einfacher zu aggregieren sind
- Errorbars/Confidence Intervals werden automatisch berechnet (wenn möglich)
- Statistische Auswertung Die Darstellung widerspiegelt eine statistische Sicht, diese kann von den eigentlichen Daten abweichen (z.B. lineplot() stellt den Mittelwert dar)

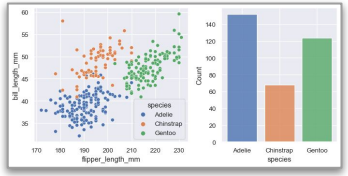
References

https://seaborn.pydata.org/tutorial.html

Die seaborn Dokumentation ist umfangreich und anschaulich, es lohnt sich zu vertiefen.

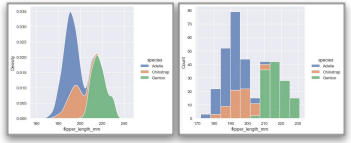
Relations

sns.relplot()
sns.scatterplot()
sns.lineplot()



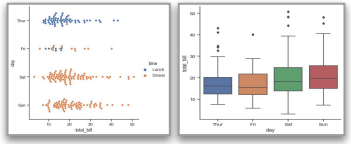
Distributions

sns.displot()
sns.histplot()
sns.kdeplot()



Categorical

sns.catplot()
sns.barplot()
sns.stripplot()
sns.swarmplot()
sns.boxplot()
sns.violinplot()



Kaggle Kurs für seaborn
https://www.kaggle.com/learn/data-visualization

Quelle: Autor und seaborn: statistical data visualization, M. Waskom, Open Journal, 2021

Kurze Einführung in plotly

plotly ist eine Python Library für interaktive Visualisierung und Web-Anwendungen

```
# library aktivieren
import plotly.express as px

# API für alle Funktionen
px.fn(dataframe, x='col', y='col',
      title='chart title',
      labels={'x': '...',
              'Y': '...'},
      width=n, height=n)

# Plots
# -- basic
px.line()
px.bar()
px.pie()

# -- statistics
px.scatter()
px.histogram()
px.box()
px.strip()

# -- maps
px.scatter_geo()
px.line_geo()
px.px.choropleth()
```

Funktionsweise
Plotly Express bietet eine breite Palette von interaktiven Plots. Interaktiv bedeutet, dass die Plot-Punkte bei Hover und Click zusätzliche Informationen anzeigen. Die Programmierung erfolgt in Python, die eigentlichen Plots werden in mit JavaScript erstellt. Deshalb ist Plotly gut geeignet um analytische Funktionen für Web-Anwendungen zu entwickeln.

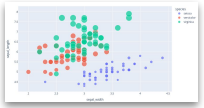
Interactive Plots
Die Plot-Grösse, der gezeigte Daten-Ausschnitt und in die Perspektive können nach der Plot-Erstellung geändert werden.

Dynamische Bedienelemente (Animationen)
Plotly kann in einem Plot zusätzliche UI-Elemente anzeigen, z.B. um Datenselektionen interaktiv durchzuführen. Einige Elemente sind

- Selektions-Listen
- Slider
- Range-Selector
- Buttons
- Automatische Animationen

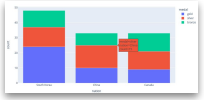
```
# Datenformat
```
Alle Funktionen arbeiten mit **long-format**. Die Statistik-Plots arbeiten zusätzlich auch mit wide-format. Siehe Referenzen.

Relations
px.scatter()
px.line()
px.bar()



Distributions
px.histogram()
px.box()

Categorical
px.histogram()
px.bar()
px.box()



References
https://plotly.com/python/plotly-express/
https://franzdiebold.github.io/plotly-express-cheat-sheet/Plotly_Express_cheat_sheet.pdf

Tutorial
https://towardsdatascience.com/cheat-codes-to-better-visualisations-with-plotly-express-21caece3db01

Quelle: Autor und Collaborative Data Science, Plotly Technologies Inc., 2015