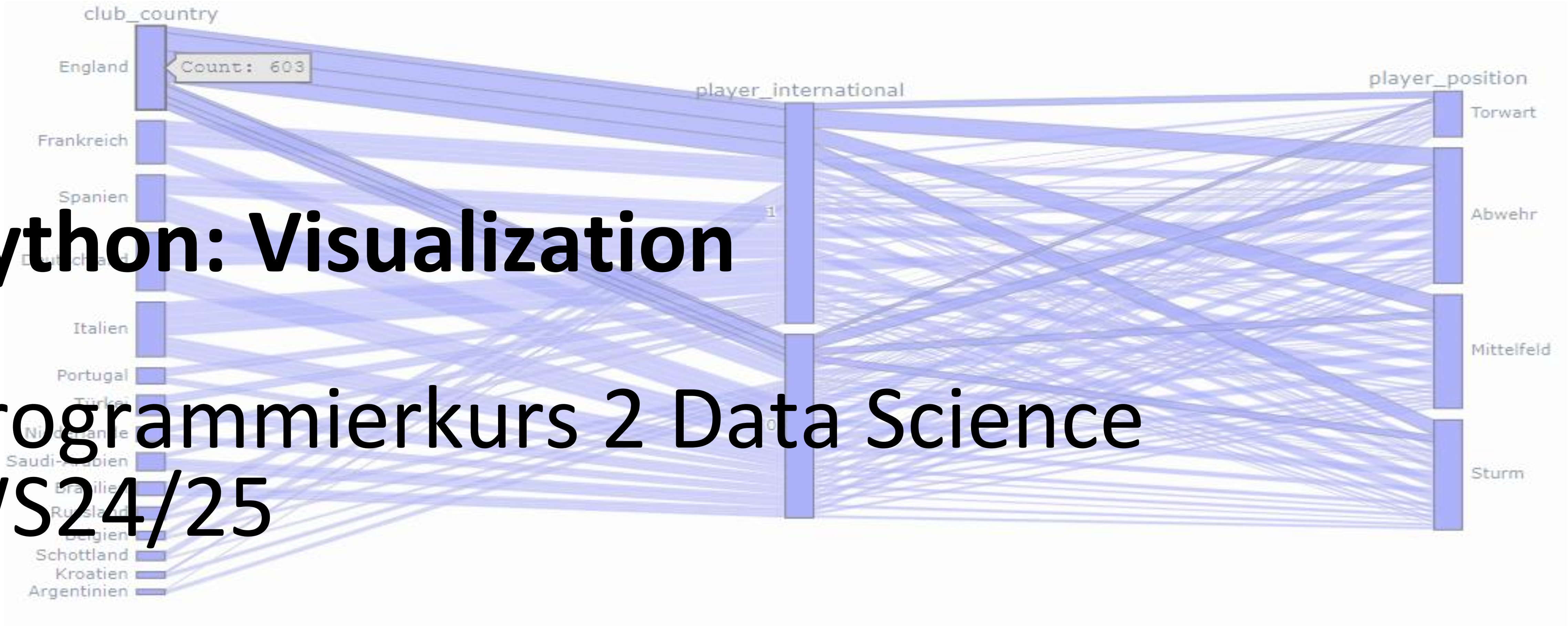
Python: Visualization

Programmierkurs 2 Data Science
WS24/25



Leonard Traeger
M. Sc. Information Systems
leonard.traeger@fh-dortmund.de

Learning Goals Python: Visualization

- **Paraphrase** the two actors, key elements, and processes of the Data Design Guide.
- **Explain** the advantages of the Matplotlib library and list a few extensions.
- Conceptually **design** Line, Scatter, Bar, Histogram, and Pie charts when given an example with multiple attributes.
- **Compare** the impact of Data Transformation processes on visualizations, and how they may affect the decoder interpretation.
- **Dicuss** the value and **draw conclusions** of Boxplot, Heatmap, and Pair-Plots (matrix).
- **Give example** on static plots and a potential extention to an interactive usage.

Encoders
Decoders

The Data Design Guide

https://medium.com/nightingale/the-cycle-of-encoding-and-decoding-f3ff17010631

→ Part of Data-Literacy ☺

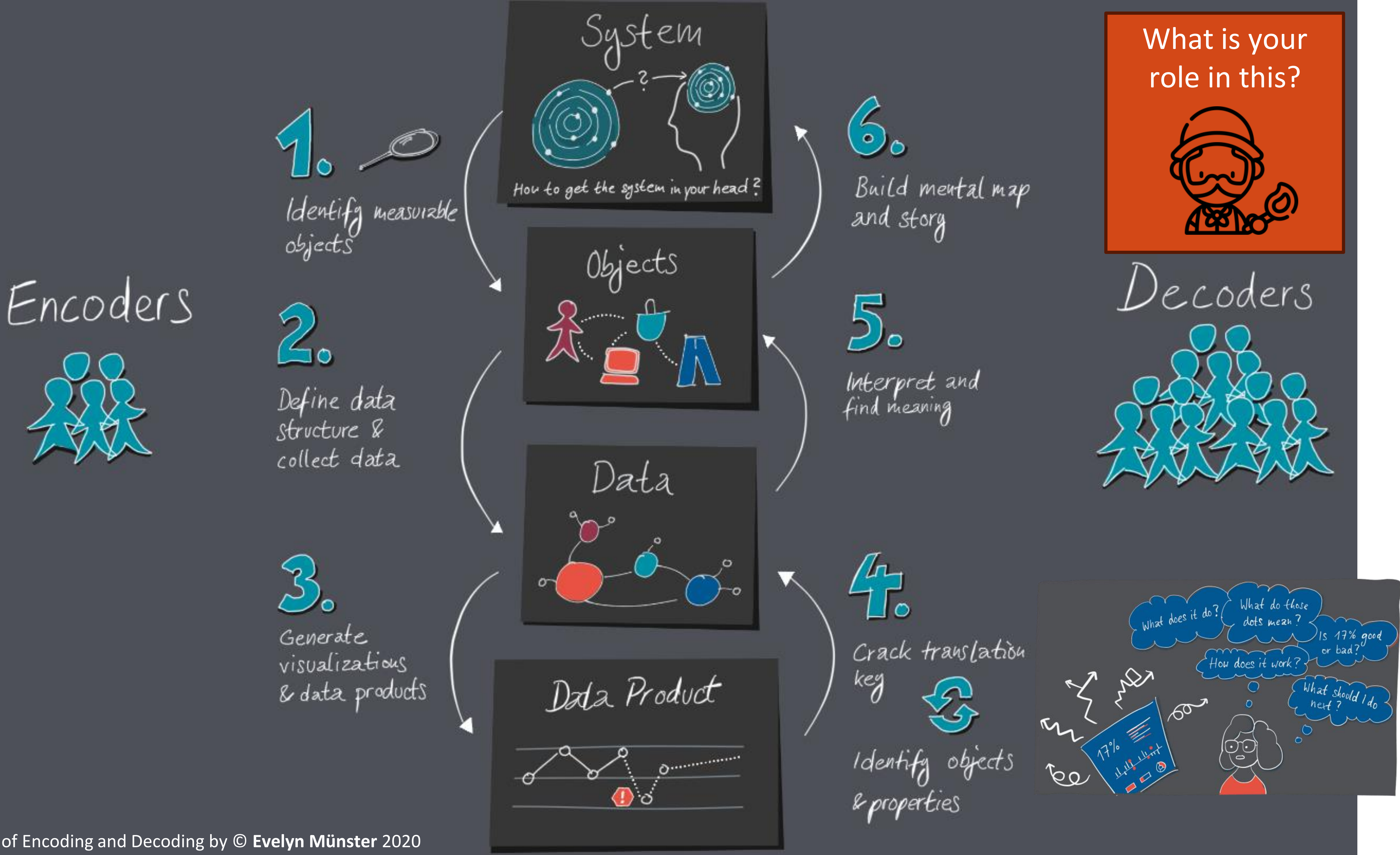
"You can see a lot by looking." – Mason and Wiggins (OSEMN)

But data is abstract and often difficult to understand.

*As Data Scientists, we want to enable others to **read** and **interpret** information.*

1. **System**
   Understand the system (mental map) behind all data including believes and questions.
2. **Objects**
   Identify measurable objects, not every detail of the real-world can be captured.
3. **Data**
   Create the data model and prioritize simplicity rather inflated data sets.
4. **Data Product**
   Graphs do not visualize all the complexity of the system.

What is your role in this?


System
How to get the system in your head?
?
Objects
Data
Data Product
Encoders
Decoders
1.
Identify measurable objects
2.
Define data structure & collect data
3.
Generate visualizations & data products
4.
Crack translation key
Identify objects & properties
5.
Interpret and find meaning
6.
Build mental map and story
What does it do?
What do those dots mean?
Is 17% good or bad?
How does it work?
What should I do next?
17%

The Cycle of Encoding and Decoding by © **Evelyn Münster** 2020

Hints for Data Visualizations

Reverse Engineering

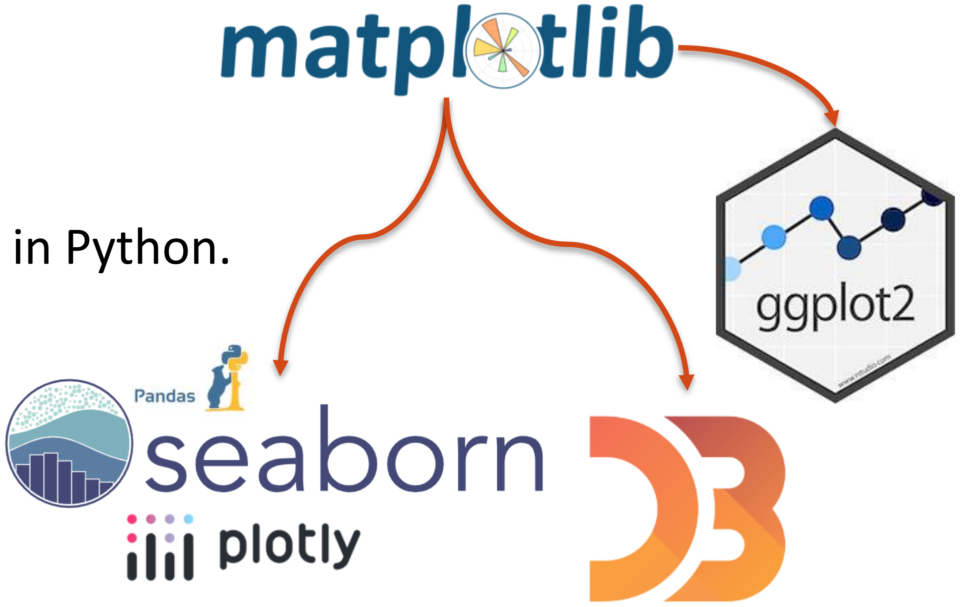
- Most extended libraries built on top of Matplotlib automate labeling, annotation, customization, grid arrangement, and many more and help **encoders** for initial mass exploration.
- Go into detail for visualisations, annotation, and styling that help the **decoder** interpret the data and deliver your key message.

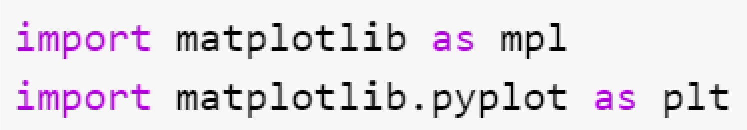
Learning Process

- Learn from examples and focus practicing with real data (your project).
- Some libraries work better for one type of visualization than others.
- Do not underestimate data transformation steps!
- Demand early feedback on your plots and stay updated with developments.
- Explore interactivity (e.g., plotly).

Matplotlib

- Released in 2003 by John Hunter.
- Multi-platform data visualization library.
- Works with containers, NumPy arrays, and Pandas Series in Python.
- Designed to work with the broader SciPy stack.

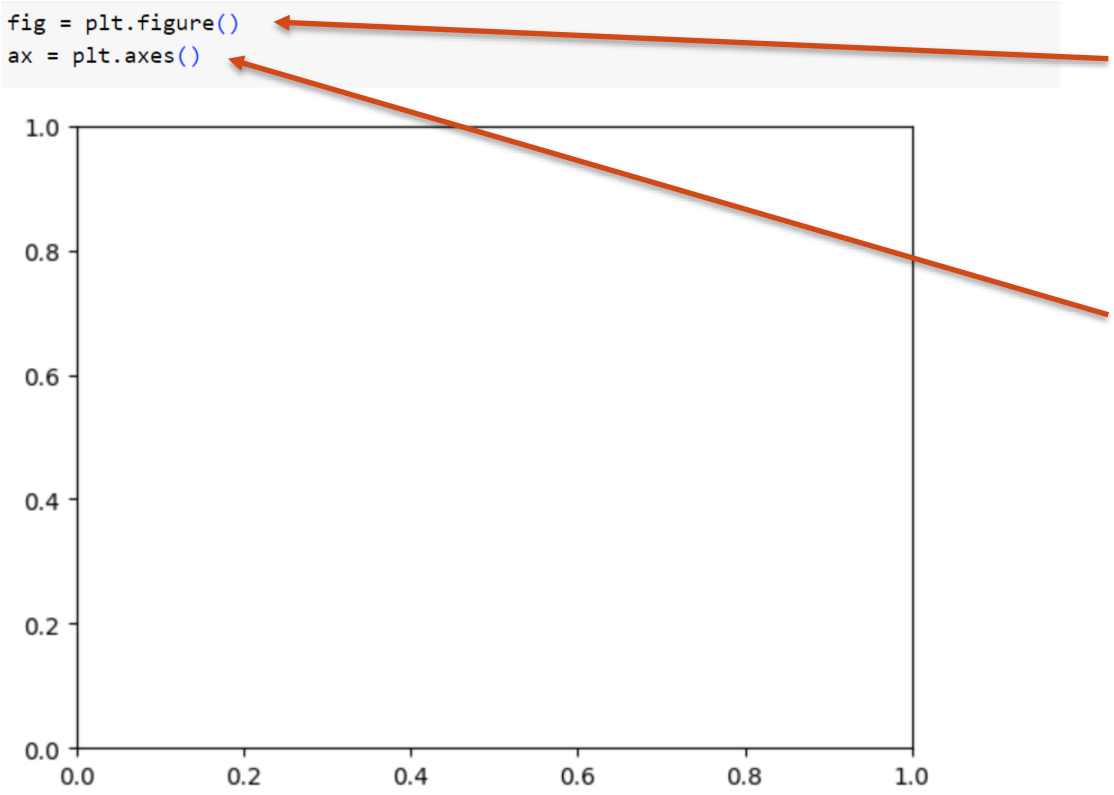




```
import matplotlib as mpl
import matplotlib.pyplot as plt
```

- Powerful extensions such as plotly, seaborn, ggplot, HoloViews, Altair, and Pandas use Matplotlib’s API.
- Matplotlib’s syntax mostly helpful for final plot creation or adjustments.

Matplotlib

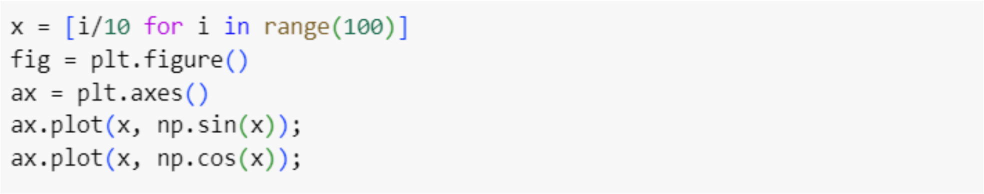


```
fig = plt.figure()
ax = plt.axes()
```

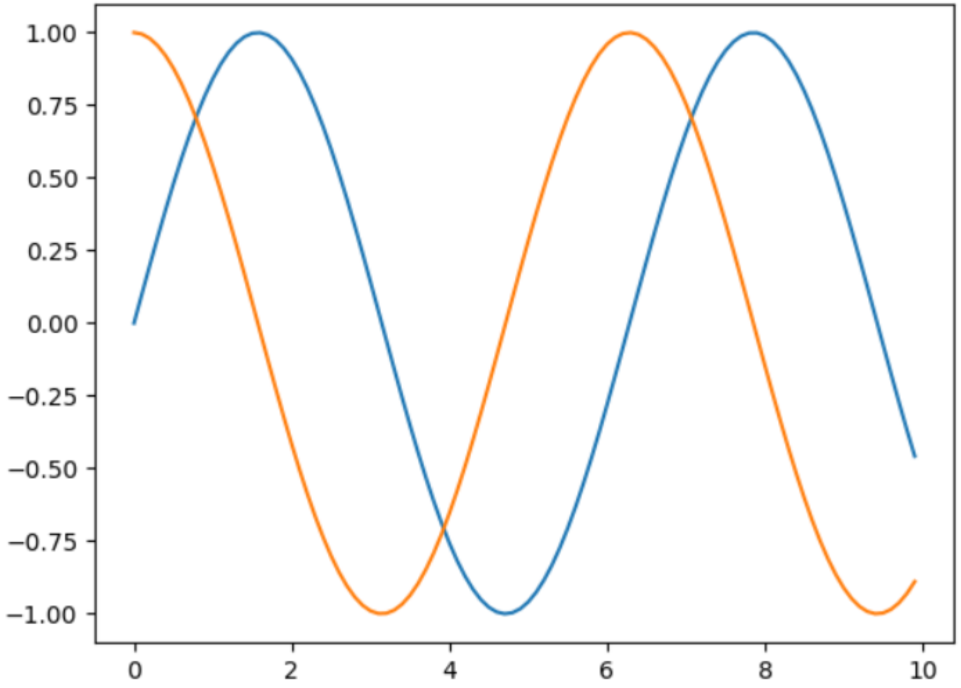
- **Figure instance**
  A single container that contains all the objects representing axes, graphics, text, and labels.
- **Axes instance** (or group)
  A bounding box with ticks and labels containing plot elements for visualization.

Matplotlib Plot

https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.plot.html#matplotlib.axes.Axes.plot



```
x = [i/10 for i in range(100)]
fig = plt.figure()
ax = plt.axes()
ax.plot(x, np.sin(x));
ax.plot(x, np.cos(x));
```



Plot y versus x as lines and/or markers.

```
ax.plot([x], y, [fmt], *,
  data=None, **kwargs)
```

- Coordinates of points or line nodes given by x, y.
- Each plot colored differently per default.

What color is the cos curve?

Matplotlib Plot (fmt)

```
fmt = '[marker]              [line]               [color]'
```

character	description
'.'	point marker
','	pixel marker
'o'	circle marker
'v'	triangle_down marker
'^'	triangle_up marker
'<'	triangle_left marker
'>'	triangle_right marker
's'	square marker
'p'	pentagon marker
'P'	plus (filled) marker
'*'	star marker
'h'	hexagon1 marker
'H'	hexagon2 marker
'+'	plus marker
'x'	x marker
'X'	x (filled) marker
'D'	diamond marker
'd'	thin_diamond marker

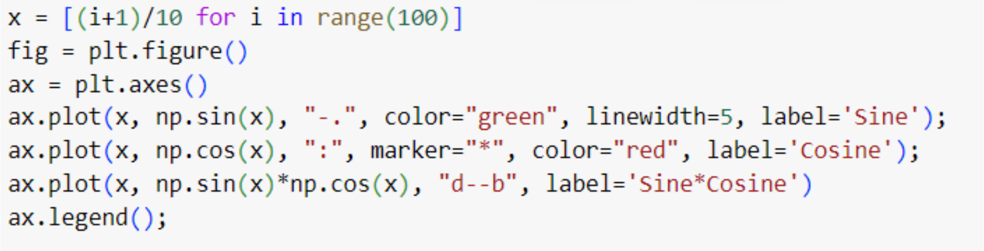
character	description
'-'	solid line style
'--'	dashed line style
'-.'	dash-dot line style
':'	dotted line style

character	color
'b'	blue
'g'	green
'r'	red
'c'	cyan
'm'	magenta
'y'	yellow
'k'	black
'w'	white

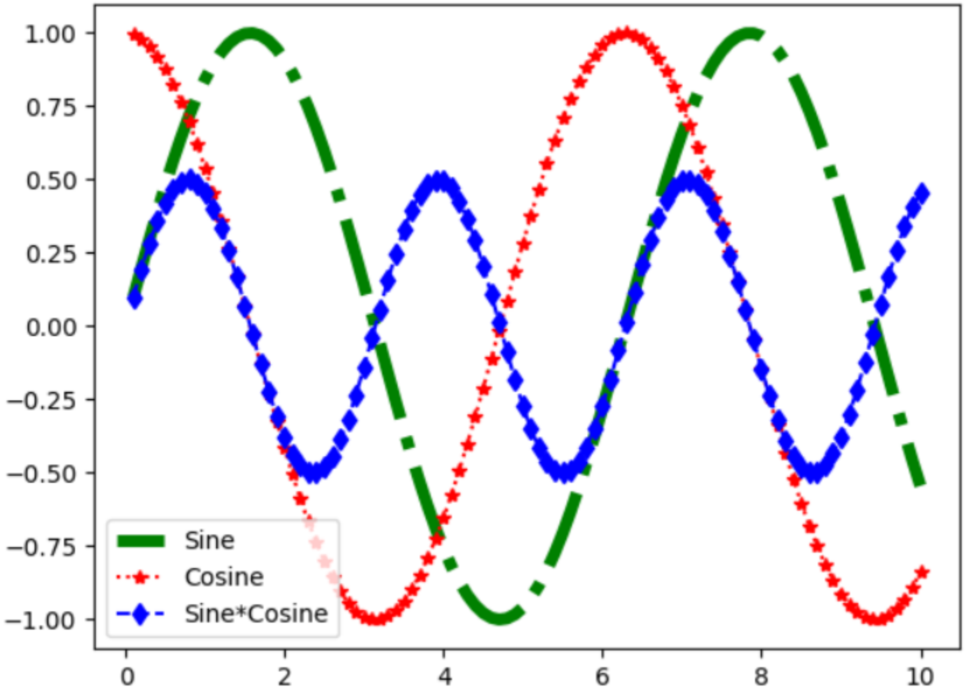
or full names ('green’)
or hex strings ('#008000')

Matplotlib Plot (cont.)

https://matplotlib.org/stable/api/_as_gen/matplotlib.lines.Line2D.html#matplotlib.lines.Line2D



```
x = [(i+1)/10 for i in range(100)]
fig = plt.figure()
ax = plt.axes()
ax.plot(x, np.sin(x), "-.", color="green", linewidth=5, label='Sine');
ax.plot(x, np.cos(x), ":", marker="*", color="red", label='Cosine');
ax.plot(x, np.sin(x)*np.cos(x), "d--b", label='Sine*Cosine')
ax.legend();
```



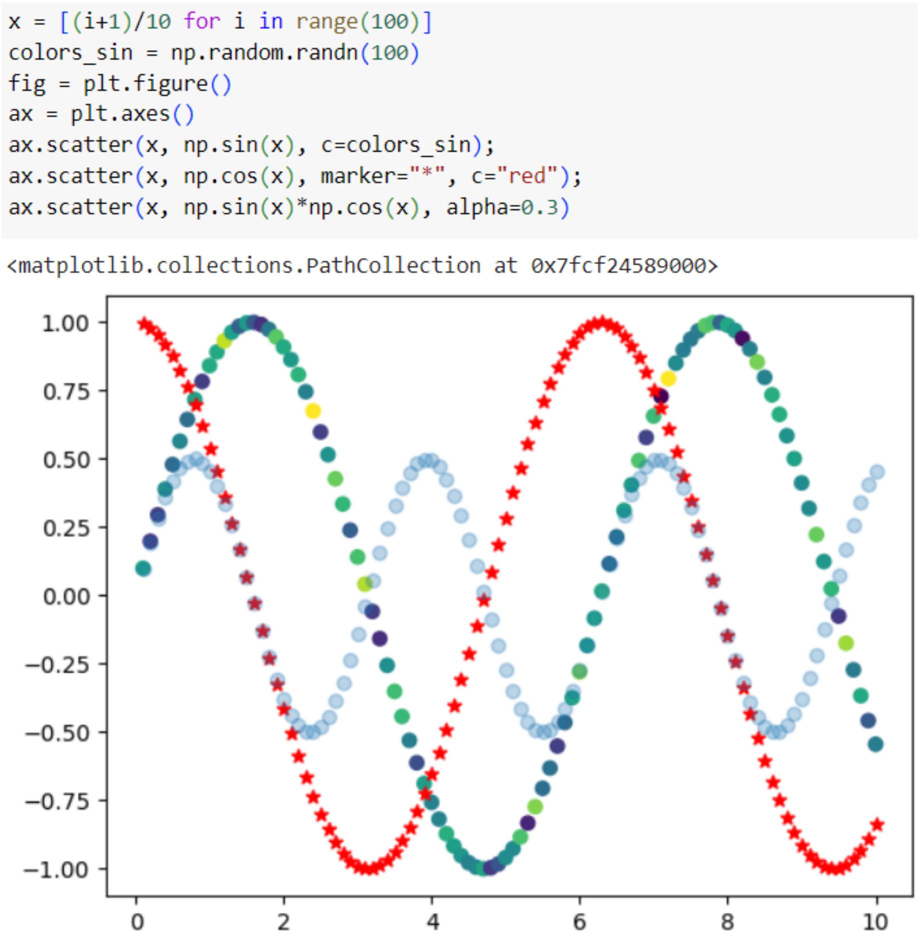
Plot y versus x as lines and/or markers.

```
ax.plot([x], y, [fmt], *,
  data=None, **kwargs)
```

- Coordinates of points or line nodes given by x, y.
- Each plot colored differently per default.
- **fmt** is a convenient color, marker and linestyle formatter.
- **Line2D** provides more parameters such as marker, markersize, linewidth, and many more.
- Add **labels** to plots and display **ax.legend.**

Matplotlib Scatter

https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.scatter.html



```
x = [(i+1)/10 for i in range(100)]
colors_sin = np.random.randn(100)
fig = plt.figure()
ax = plt.axes()
ax.scatter(x, np.sin(x), c=colors_sin);
ax.scatter(x, np.cos(x), marker="*", c="red");
ax.scatter(x, np.sin(x)*np.cos(x), alpha=0.3)
```

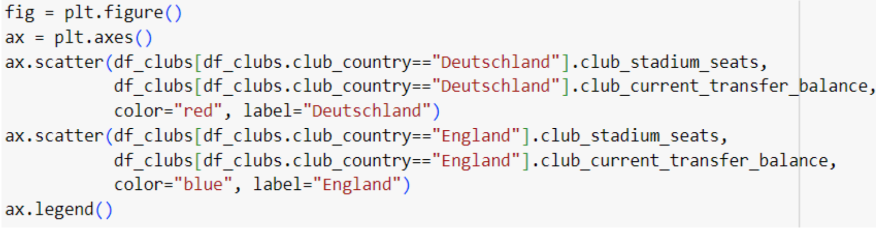
```
<matplotlib.collections.PathCollection at 0x7fcf24589000>
```

Plot y versus x as markers (close plot cousin)

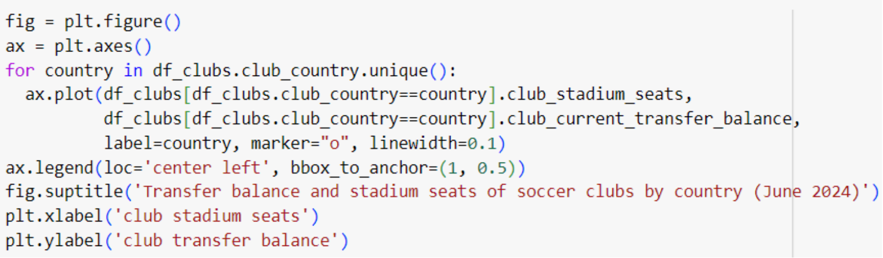
- Allows you to control and configure **points individually** (size, face color, edge color, etc.) mapped to data.
- `plt.plot` should be preferred over `plt.scatter` if extra work for each point does not pay off.

Matplotlib Plot and Scatter (cont.)

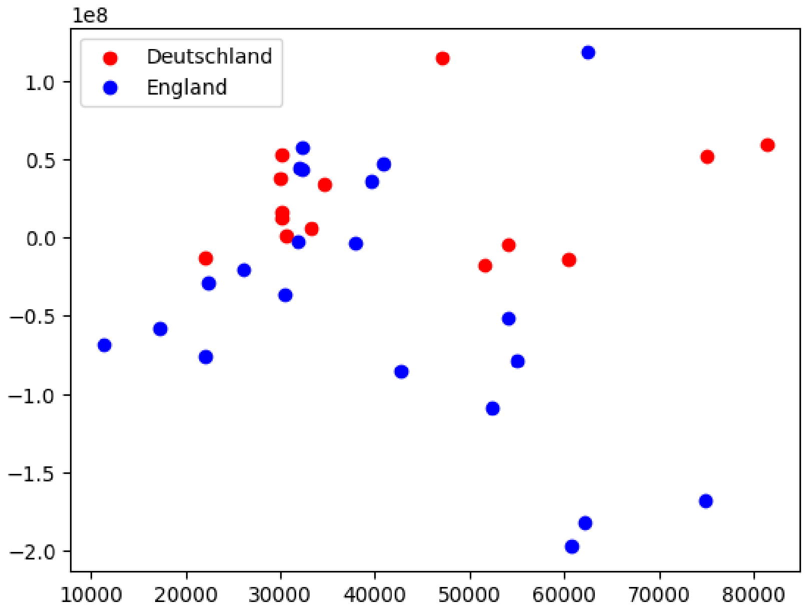
What information can the Encoder interpret?

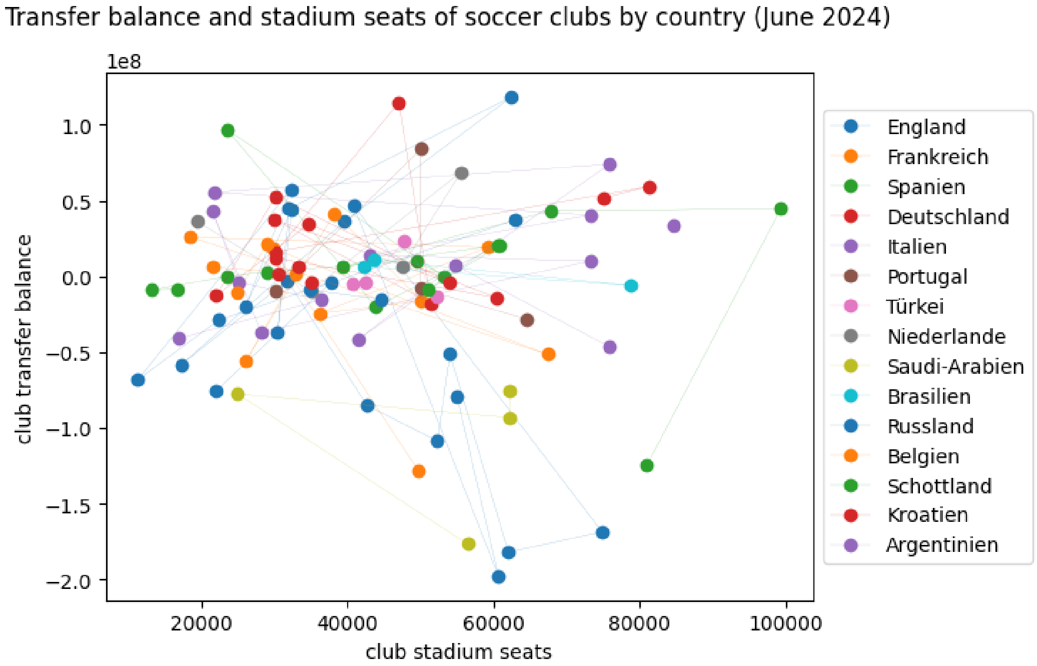


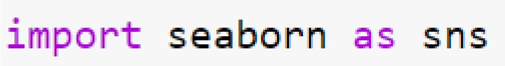
```
fig = plt.figure()
ax = plt.axes()
ax.scatter(df_clubs[df_clubs.club_country=="Deutschland"].club_stadium_seats,
           df_clubs[df_clubs.club_country=="Deutschland"].club_current_transfer_balance,
           color="red", label="Deutschland")
ax.scatter(df_clubs[df_clubs.club_country=="England"].club_stadium_seats,
           df_clubs[df_clubs.club_country=="England"].club_current_transfer_balance,
           color="blue", label="England")
ax.legend()
```



```
fig = plt.figure()
ax = plt.axes()
for country in df_clubs.club_country.unique():
  ax.plot(df_clubs[df_clubs.club_country==country].club_stadium_seats,
          df_clubs[df_clubs.club_country==country].club_current_transfer_balance,
          label=country, marker="o", linewidth=0.1)
ax.legend(loc='center left', bbox_to_anchor=(1, 0.5))
fig.suptitle('Transfer balance and stadium seats of soccer clubs by country (June 2024)')
plt.xlabel('club stadium seats')
plt.ylabel('club transfer balance')
```



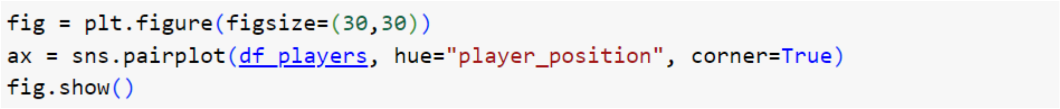




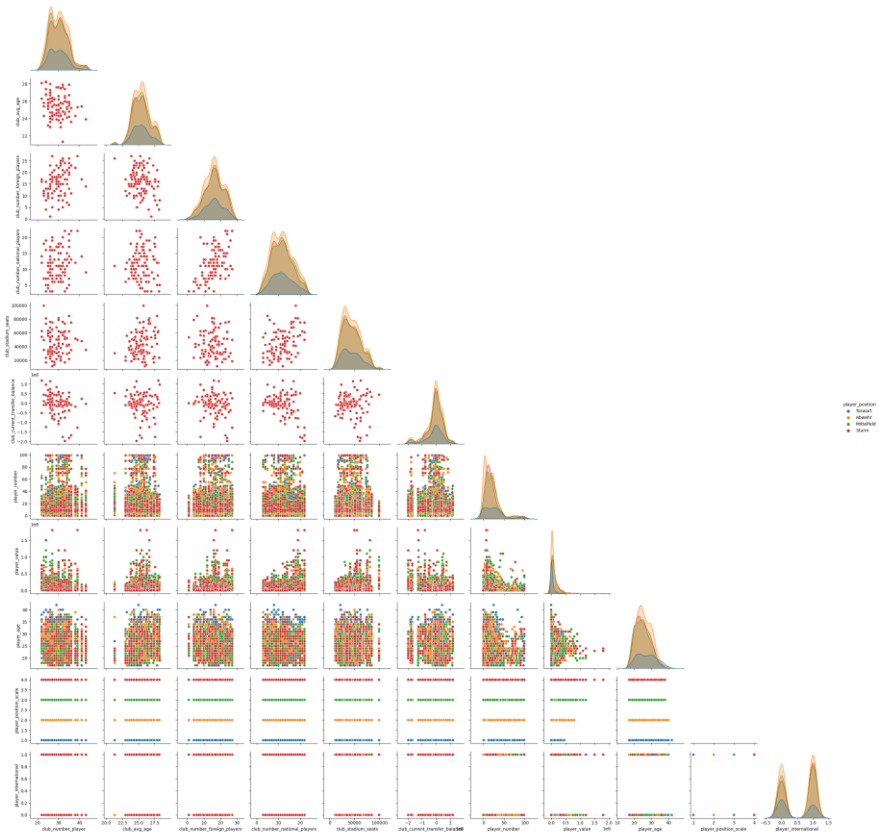
```
import seaborn as sns
```

Matplotlib Scatter-Matrix

https://seaborn.pydata.org/generated/seaborn.pairplot.html



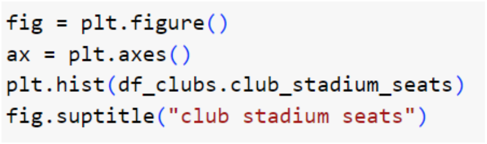
```
fig = plt.figure(figsize=(30,30))
ax = sns.pairplot(df_players, hue="player_position", corner=True)
fig.show()
```



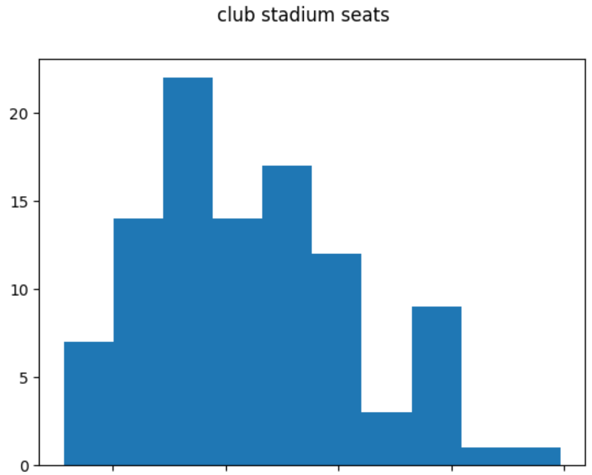
`sns.pairplot(df,hue,…)`

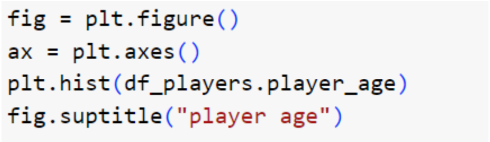
- Plot several pairs of variables and their joint distributions.
- Can be displayed individually and in more detail and examined in more detail by the data scientist.
- **hue** and **marker** can visualize additional dimensions.

Matplotlib Histogram

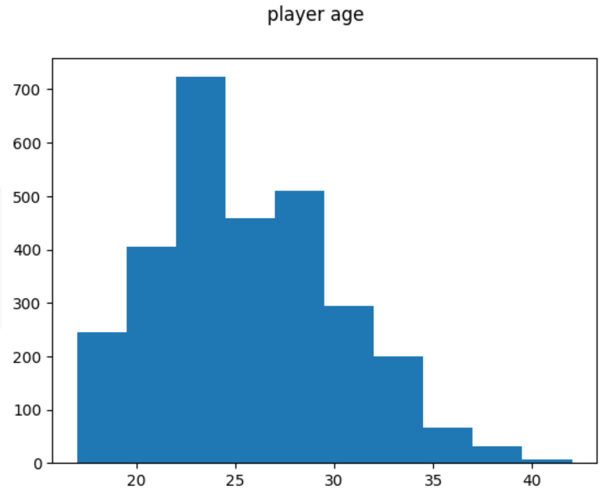


```
fig = plt.figure()
ax = plt.axes()
plt.hist(df_clubs.club_stadium_seats)
fig.suptitle("club stadium seats")
```





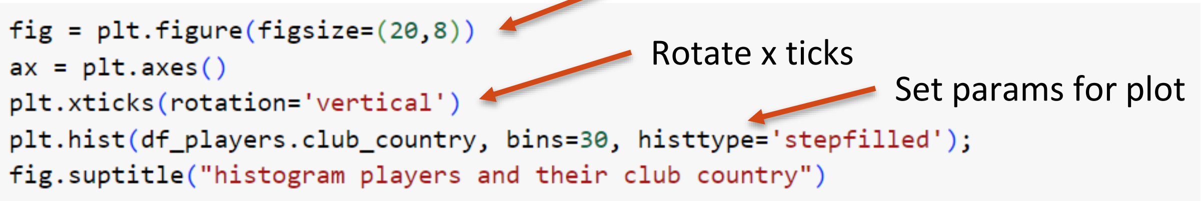
```
fig = plt.figure()
ax = plt.axes()
plt.hist(df_players.player_age)
fig.suptitle("player age")
```



`plt.hist(data, bin, ...)`

- Data can be any iterable container (e.g., list, NumPy array, Series).
- Representation of the distribution of data and frequencies.
- Bin the data in x and count the number of values in each bin.
- Great to get a sense of location, spread and skewness of the data (e.g., unimodal, bimodal or multimodal).

Matplotlib Histogram (cont.)

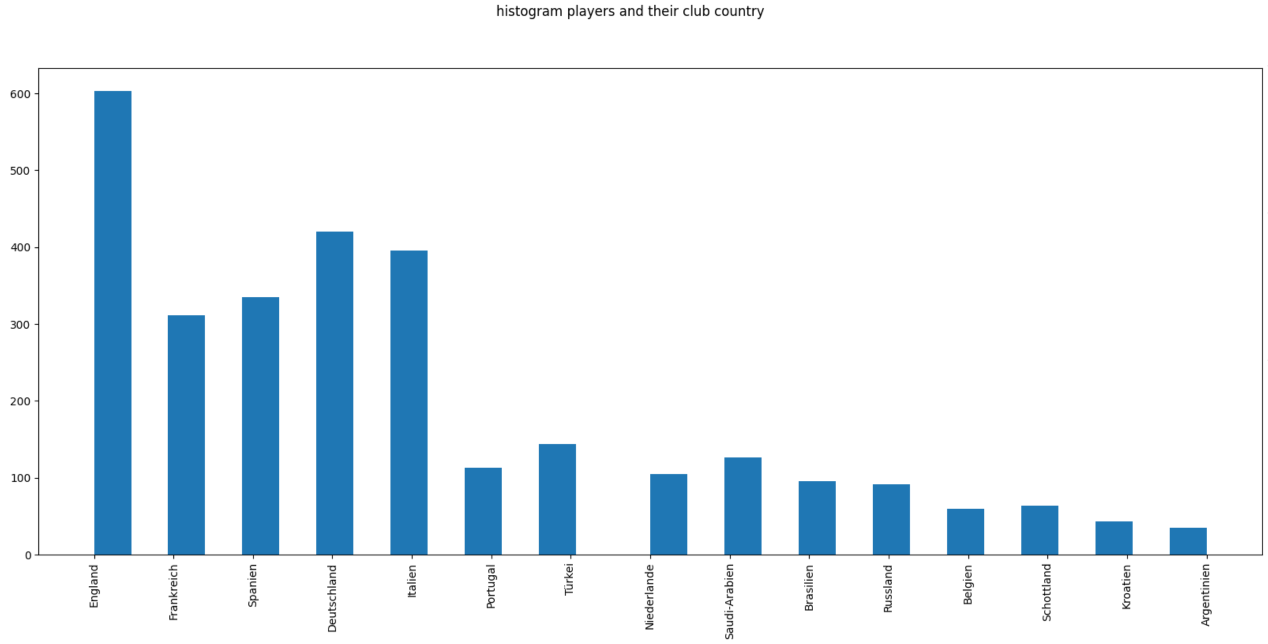


```
fig = plt.figure(figsize=(20,8))
ax = plt.axes()
plt.xticks(rotation='vertical')
plt.hist(df_players.club_country, bins=30, histtype='stepfilled');
fig.suptitle("histogram players and their club country")
```

Set figure size (width, height)

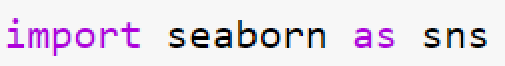
Rotate x ticks

Set params for plot



`plt.hist(data, bin, ...)`

- Data can be any iterable container (e.g., list, NumPy array, Series)
- Representation of the distribution of data and frequencies.
- Bin the data in x and count the number of values in each bin.
- Great to get a sense of location, spread and skewness of the data (e.g., unimodal, bimodal or multimodal).

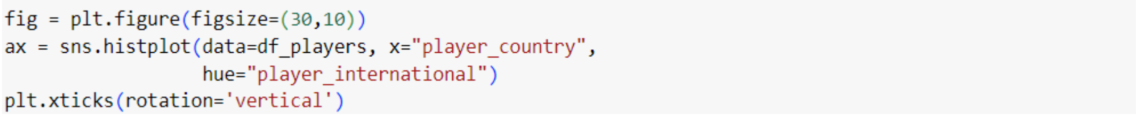


```python
import seaborn as sns
```

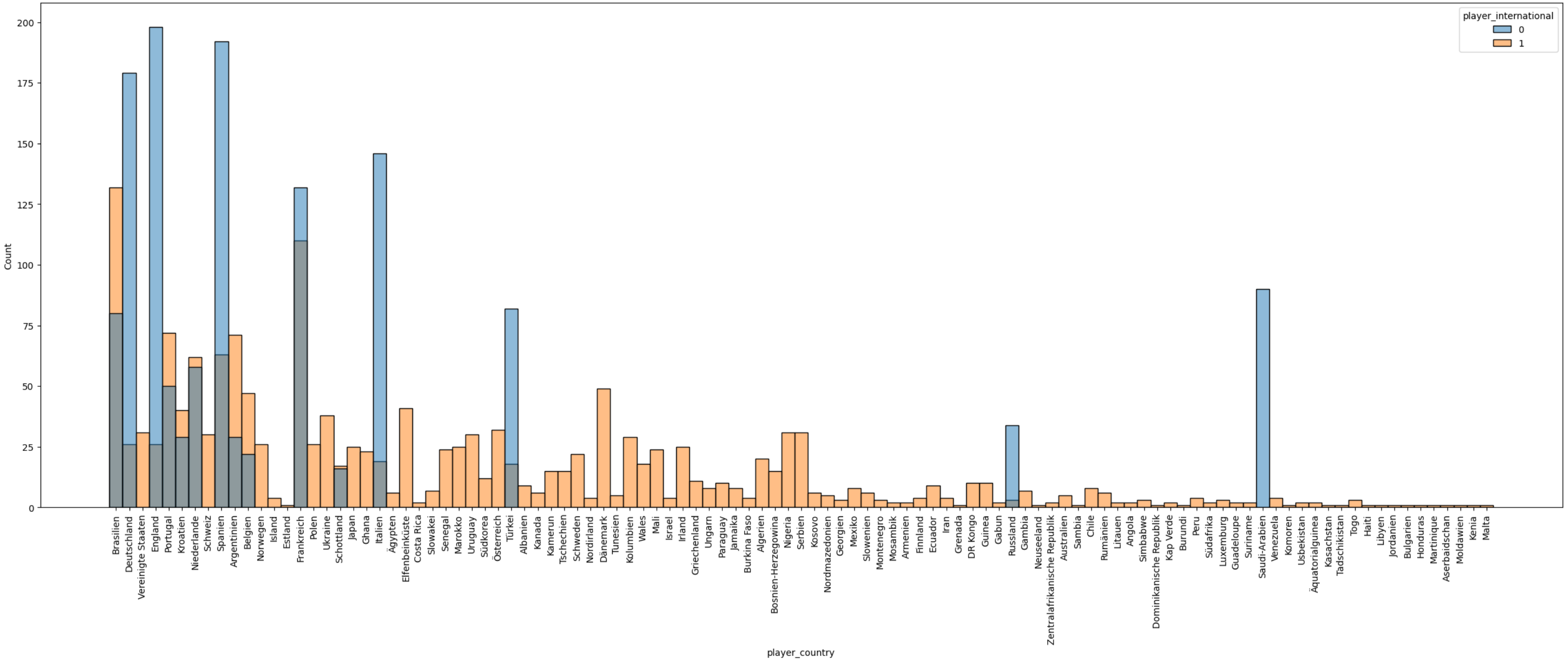
Seaborn Histogram

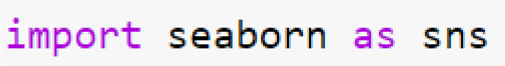
https://seaborn.pydata.org/generated/seaborn.histplot.html

Draw histograms with **hue** mapping and **transparent overlapping** layers.



```python
fig = plt.figure(figsize=(30,10))
ax = sns.histplot(data=df_players, x="player_country",
                  hue="player_international")
plt.xticks(rotation='vertical')
```



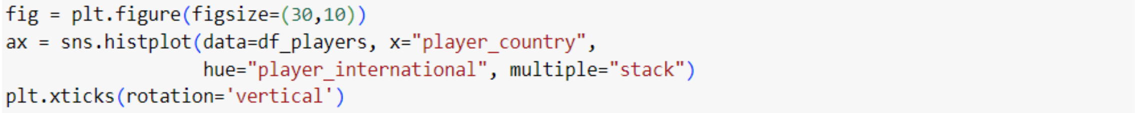


```
import seaborn as sns
```

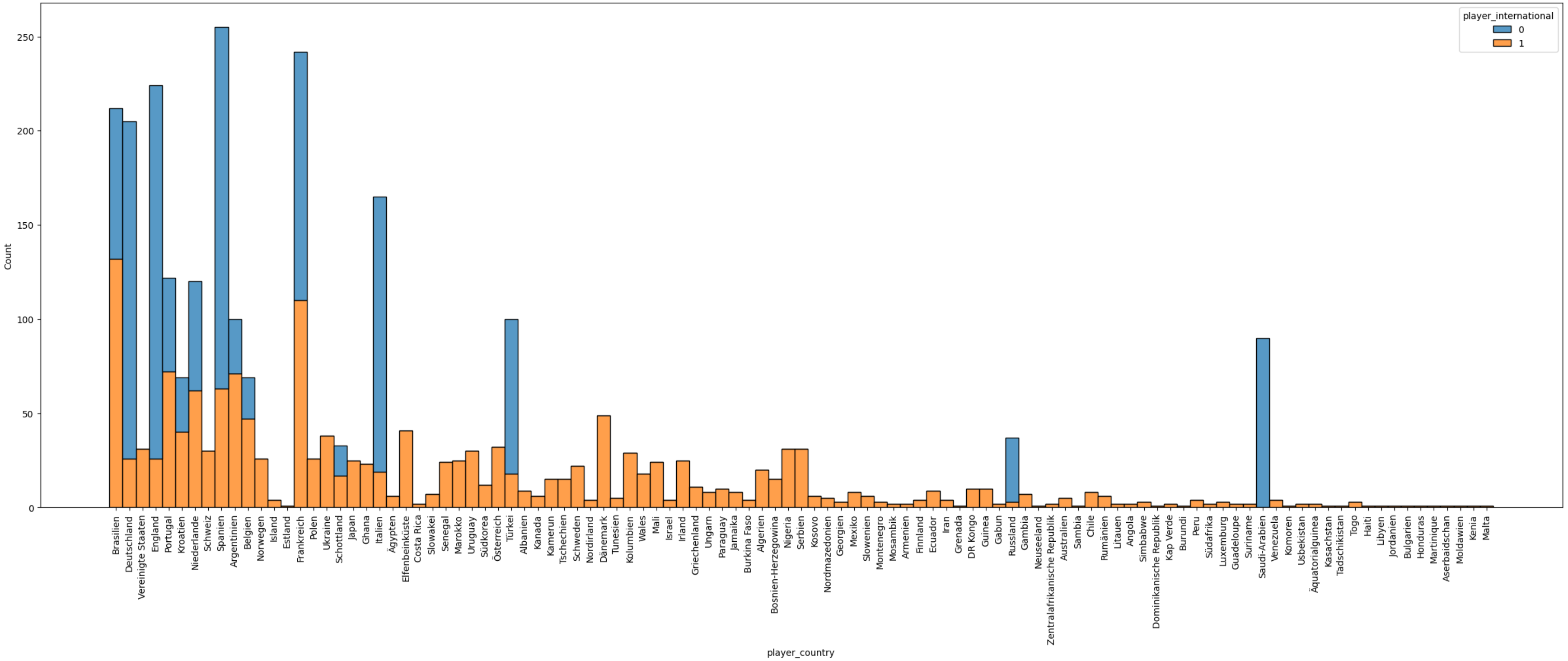
Seaborn Histogram (cont.)

https://seaborn.pydata.org/generated/seaborn.histplot.html

Draw histograms with **stacked hue** mapping.

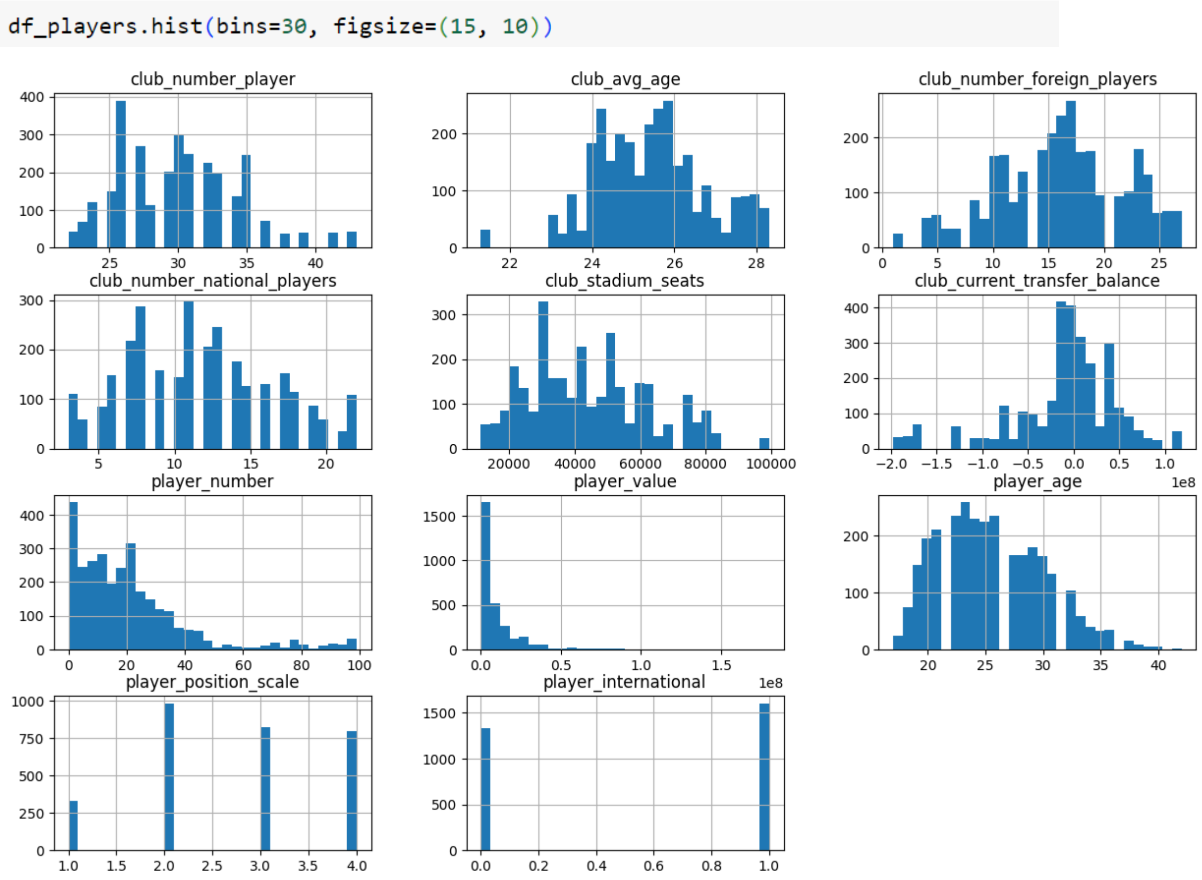


```
fig = plt.figure(figsize=(30,10))
ax = sns.histplot(data=df_players, x="player_country",
                  hue="player_international", multiple="stack")
plt.xticks(rotation='vertical')
```



Pandas Histogram

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.hist.html



```
df_players.hist(bins=30, figsize=(15, 10))
```

`DataFrame.hist()`

- Calls matplotlib.pyplot.hist() on each series in the DataFrame.

Data Transformation (Recap)

*A function that **maps** the entire **set** of values of a given attribute **to** a **new set** of **replacement values**.*

Do not underestimate Data Transformation within your dataset!

Attribute Construction:

- Unary function definition `f(A)` → `A`, where A is a set or
- Binary function definition `f(A, B)` → `A*B`, whera A.index ≡ B.index, and * some operation

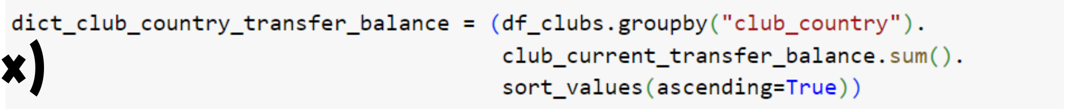
Aggregation: involves grouping and computations such as sum(), mean(), median(), min(), and max(), to generate insights into the nature of numeric values.

Generalization: concept hierarchy climbing.

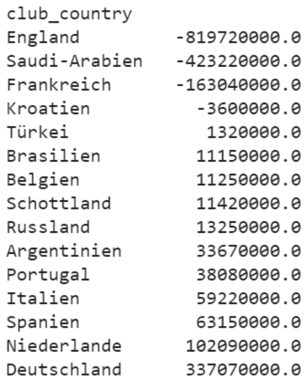
Normalization: series transformation to a scale so values lie within a specified range (usually smaller and positive).

Matplotlib Horizontal Bar

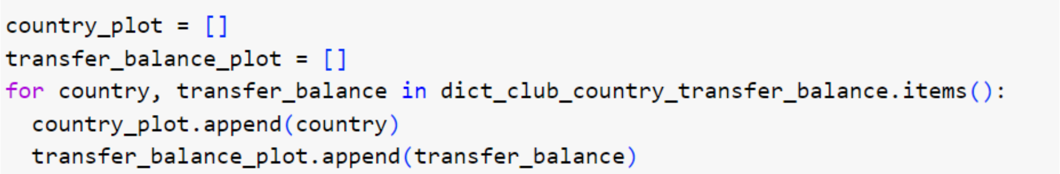
https://matplotlib.org/stable/gallery/lines_bars_and_markers/barh.html



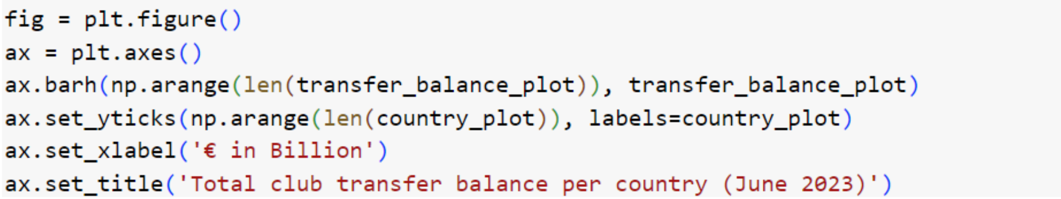
```
dict_club_country_transfer_balance = (df_clubs.groupby("club_country").
                                      club_current_transfer_balance.sum().
                                      sort_values(ascending=True))
```



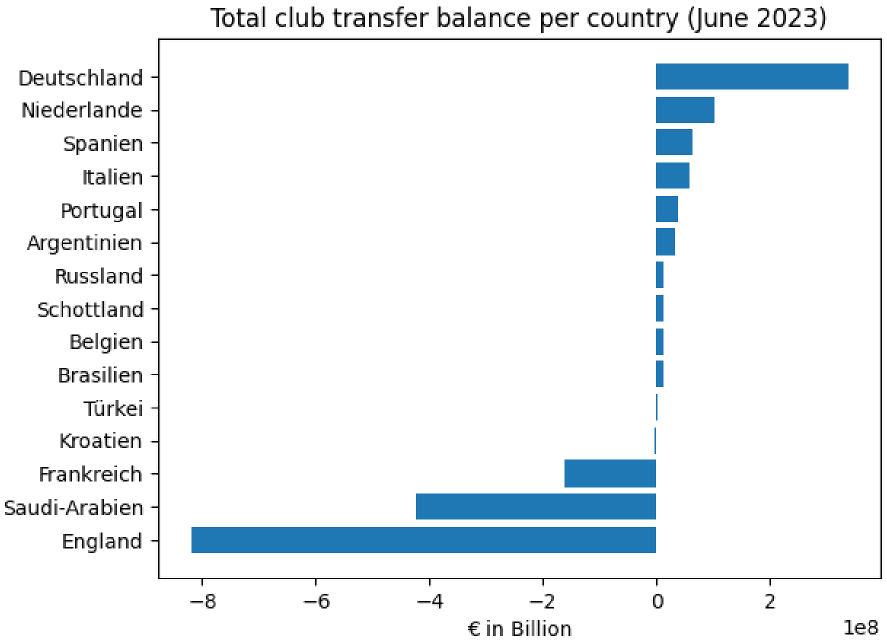
```
club_country
England        -819720000.0
Saudi-Arabien  -423220000.0
Frankreich     -163040000.0
Kroatien         -3600000.0
Türkei            1320000.0
Brasilien        11150000.0
Belgien          11250000.0
Schottland       11420000.0
Russland         13250000.0
Argentinien      33670000.0
Portugal         38080000.0
Italien          59220000.0
Spanien          63150000.0
Niederlande     102090000.0
Deutschland     337070000.0
```



```
country_plot = []
transfer_balance_plot = []
for country, transfer_balance in dict_club_country_transfer_balance.items():
  country_plot.append(country)
  transfer_balance_plot.append(transfer_balance)
```

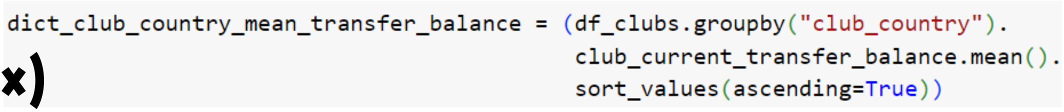


```
fig = plt.figure()
ax = plt.axes()
ax.barh(np.arange(len(transfer_balance_plot)), transfer_balance_plot)
ax.set_yticks(np.arange(len(country_plot)), labels=country_plot)
ax.set_xlabel('€ in Billion')
ax.set_title('Total club transfer balance per country (June 2023)')
```

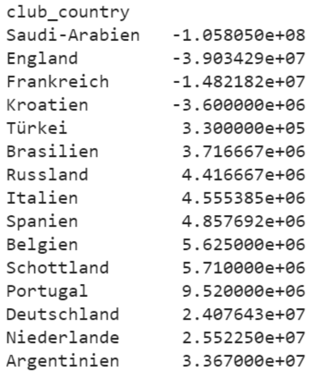


Matplotlib Horizontal Bar (cont.)

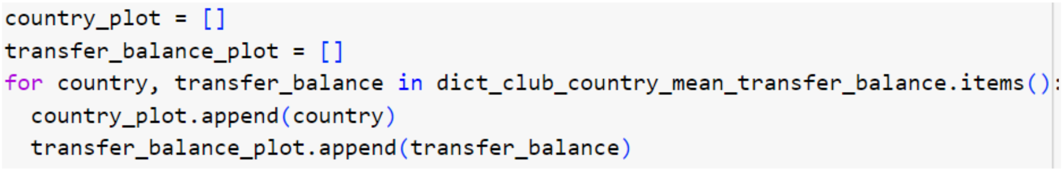
https://matplotlib.org/stable/gallery/lines_bars_and_markers/barh.html



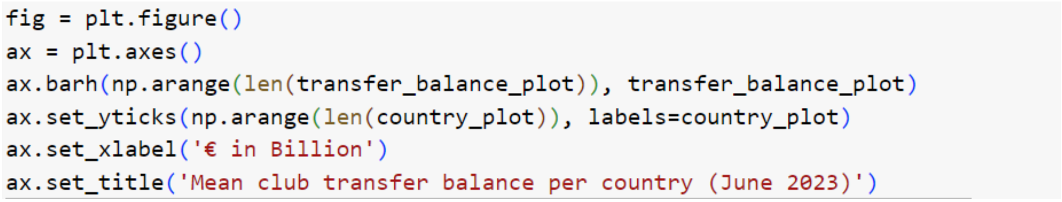
```
dict_club_country_mean_transfer_balance = (df_clubs.groupby("club_country").
                                           club_current_transfer_balance.mean().
                                           sort_values(ascending=True))
```



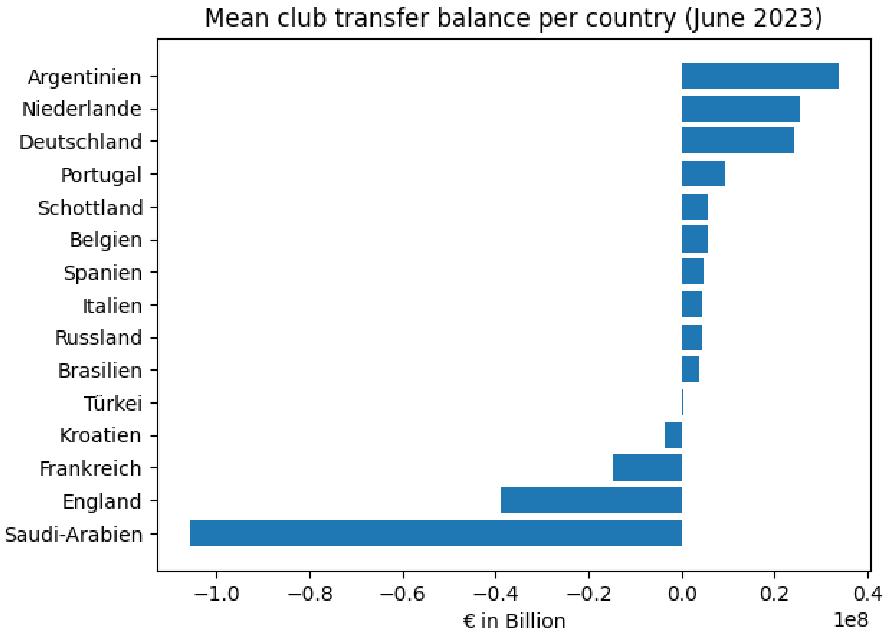
```
club_country
Saudi-Arabien    -1.058050e+08
England          -3.903429e+07
Frankreich       -1.482182e+07
Kroatien         -3.600000e+06
Türkei            3.300000e+05
Brasilien         3.716667e+06
Russland          4.416667e+06
Italien           4.555385e+06
Spanien           4.857692e+06
Belgien           5.625000e+06
Schottland        5.710000e+06
Portugal          9.520000e+06
Deutschland       2.407643e+07
Niederlande       2.552250e+07
Argentinien       3.367000e+07
```



```
country_plot = []
transfer_balance_plot = []
for country, transfer_balance in dict_club_country_mean_transfer_balance.items():
  country_plot.append(country)
  transfer_balance_plot.append(transfer_balance)
```



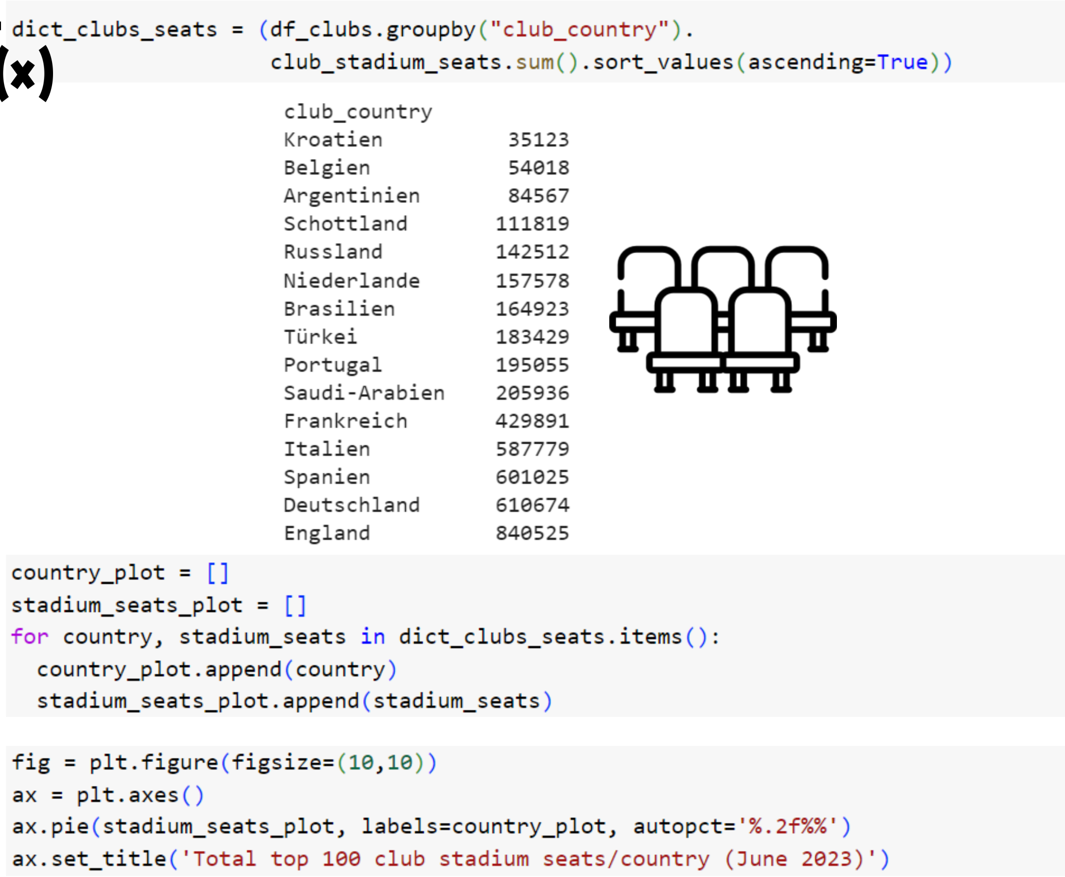
```
fig = plt.figure()
ax = plt.axes()
ax.barh(np.arange(len(transfer_balance_plot)), transfer_balance_plot)
ax.set_yticks(np.arange(len(country_plot)), labels=country_plot)
ax.set_xlabel('€ in Billion')
ax.set_title('Mean club transfer balance per country (June 2023)')
```



Two different messages!

Matplotlib Pie

https://matplotlib.org/stable/gallery/pie_and_polar_charts/pie_features.html

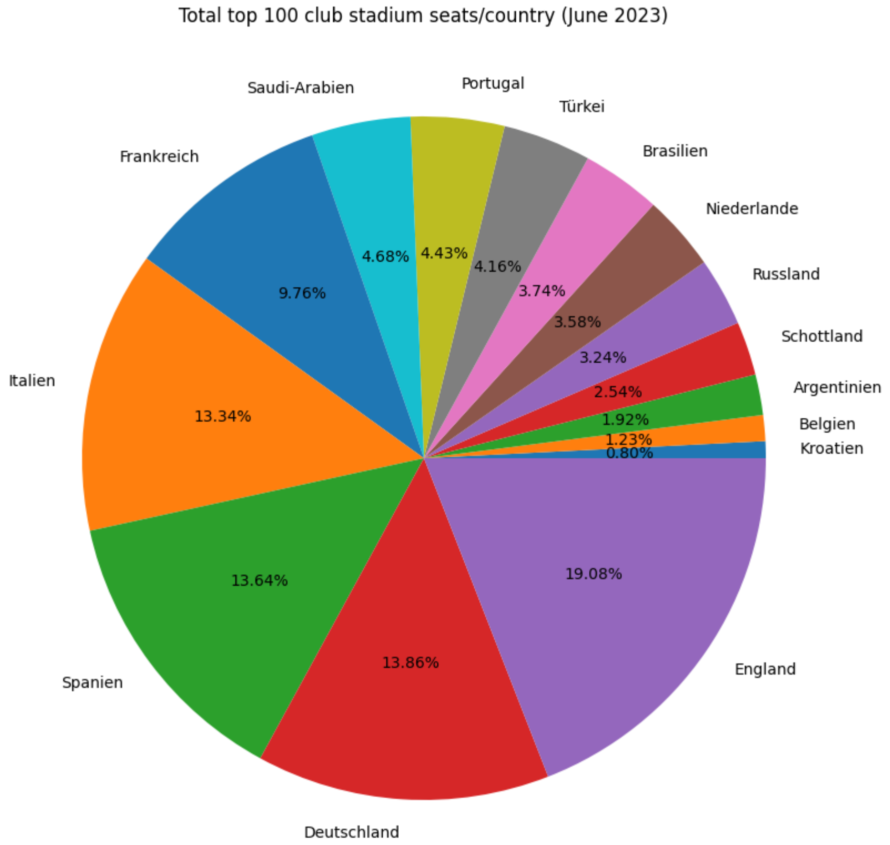


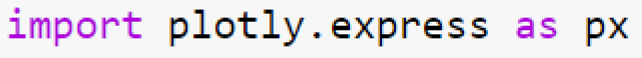
```
dict_clubs_seats = (df_clubs.groupby("club_country").
                    club_stadium_seats.sum().sort_values(ascending=True))
```

```
club_country
Kroatien         35123
Belgien          54018
Argentinien      84567
Schottland      111819
Russland        142512
Niederlande     157578
Brasilien       164923
Türkei          183429
Portugal        195055
Saudi-Arabien   205936
Frankreich      429891
Italien         587779
Spanien         601025
Deutschland     610674
England         840525
```

```
country_plot = []
stadium_seats_plot = []
for country, stadium_seats in dict_clubs_seats.items():
  country_plot.append(country)
  stadium_seats_plot.append(stadium_seats)
```

```
fig = plt.figure(figsize=(10,10))
ax = plt.axes()
ax.pie(stadium_seats_plot, labels=country_plot, autopct='%.2f%%')
ax.set_title('Total top 100 club stadium seats/country (June 2023)')
```

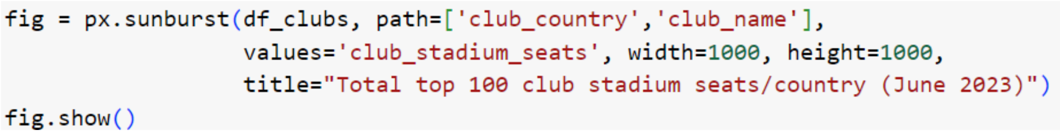




```
import plotly.express as px
```

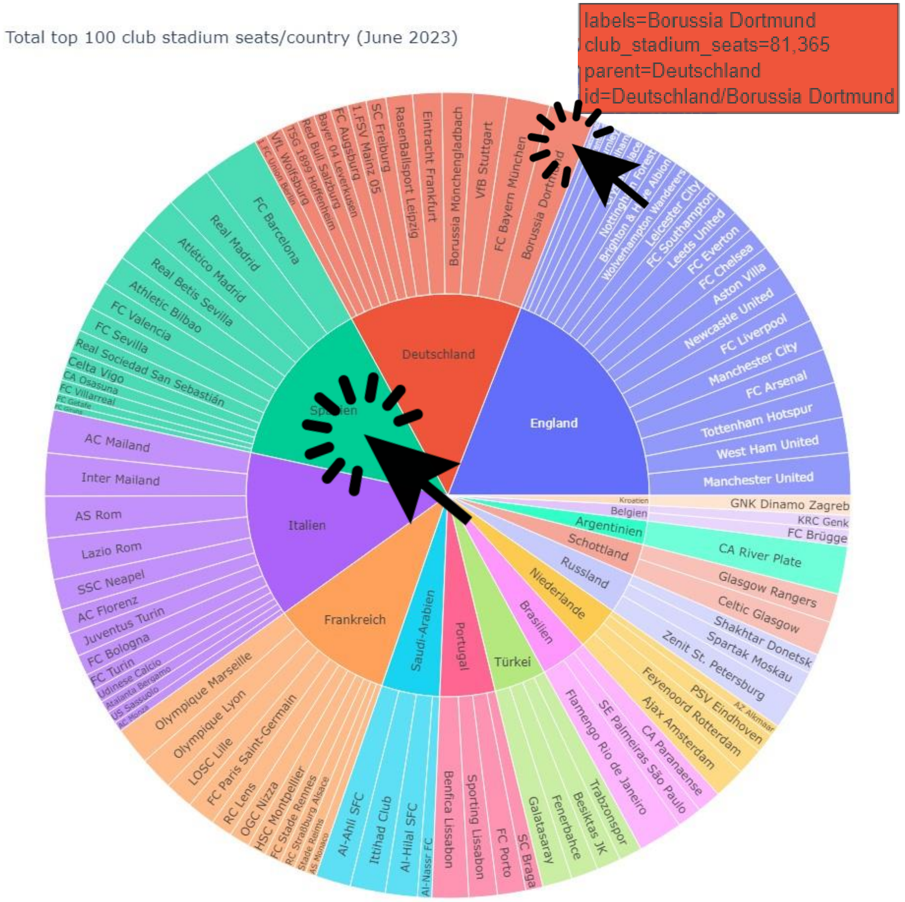
Plotly Sunburst-Pie

https://plotly.com/python/sunburst-charts/



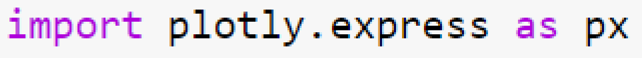
```
fig = px.sunburst(df_clubs, path=['club_country','club_name'],
                  values='club_stadium_seats', width=1000, height=1000,
                  title="Total top 100 club stadium seats/country (June 2023)")
fig.show()
```

- Sunbursts visualize hierarchical data spanning outwards radially from root to leaves (like treemaps).
- Each row of the DataFrame is represented as a sector of the sunburst.
- Path parameter corresponding to a list of series in outward order.



Total top 100 club stadium seats/country (June 2023)

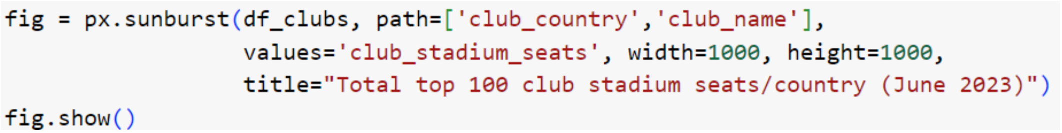
labels=Borussia Dortmund
club_stadium_seats=81,365
parent=Deutschland
id=Deutschland/Borussia Dortmund



```
import plotly.express as px
```

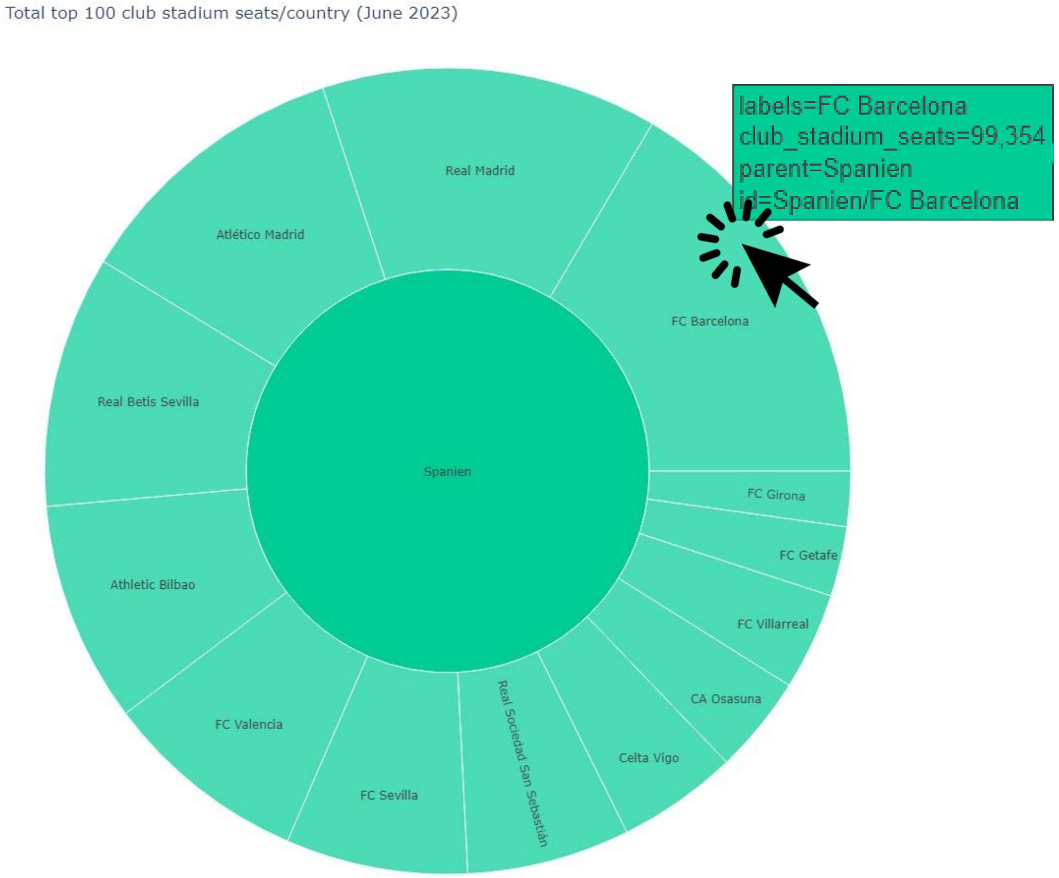
Plotly Sunburst-Pie (cont.)

https://plotly.com/python/sunburst-charts/

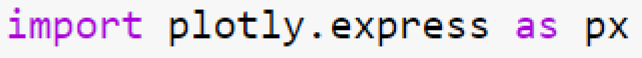


```
fig = px.sunburst(df_clubs, path=['club_country','club_name'],
                  values='club_stadium_seats', width=1000, height=1000,
                  title="Total top 100 club stadium seats/country (June 2023)")
fig.show()
```

- Sunbursts visualize hierarchical data spanning outwards radially from root to leaves (like treemaps).
- Each row of the DataFrame is represented as a sector of the sunburst.
- Path parameter corresponding to a list of series in outward order.
- Plotly charts are interactive and provide settings for hover animations and custom controls.



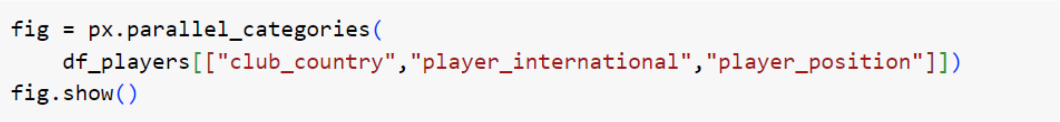
Total top 100 club stadium seats/country (June 2023)



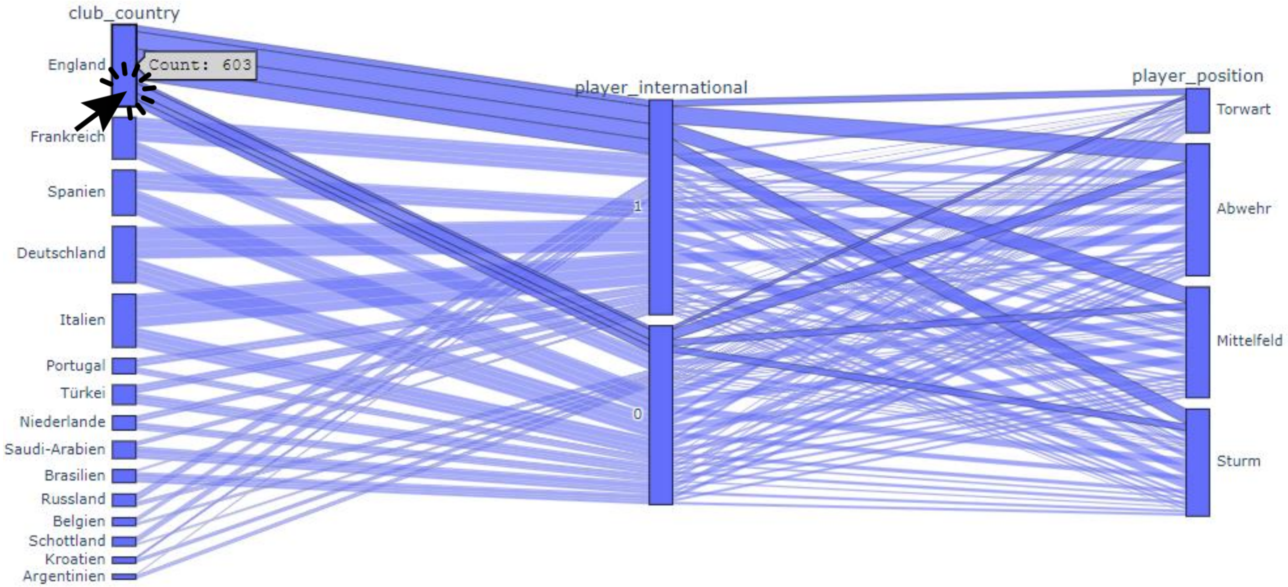
```
import plotly.express as px
```

Plotly Parallel-Sets

https://plotly.com/python/parallel-categories-diagram/



```
fig = px.parallel_categories(
    df_players[["club_country","player_international","player_position"]])
fig.show()
```



Visualization of **multi-dimensional categorical** data sets.

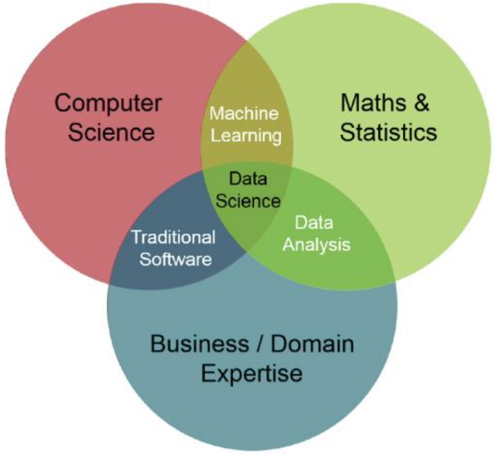
- Each variable represented by a column of rectangles.
- Relative heights of the rectangles reflect the relative frequency of occurrence.
- Ribbons connect rectangles corresponding to the relative frequency of occurrence of the combination given by the order of the DataFrame columns.

Let us recap the past slides...

As Data Scientists, we encode

- A system into objects *(e.g., DFB: increase transparency of soccer player).*
- Objects into data *(e.g., soccer players and clubs).*
- Data into data products *(e.g., Line, Scatter, Histograms, Bar, Pie, and many others).*

To help the decoder interpret data and deliver some message.



The role of a Data Scientists may be a lot, but we can use some statistics and visualizations to guide and help ourselves.

Preprocessing Considerations (Recap)

Think about likely **causes of noise** and errors when **correcting and transforming data**, e.g.,

- Do two extremely similar attributes really represent the same?
- Does a missing value have more meaning in the data context than np.NaN?
- Is this “outlier“ really an outlier, or is there a reasonable explanation for it?
- Does removing an outlier harm or help interpreting the whole data context?

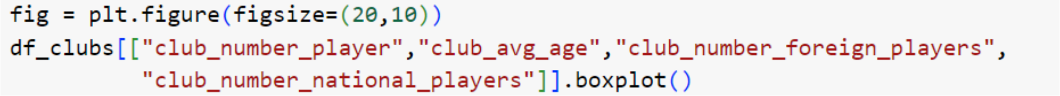
Consider **ethics** when applying Data Integration and Transformations:

- Limit harmful uses
- Reflect diversity / inclusion
- Uphold human rights and values

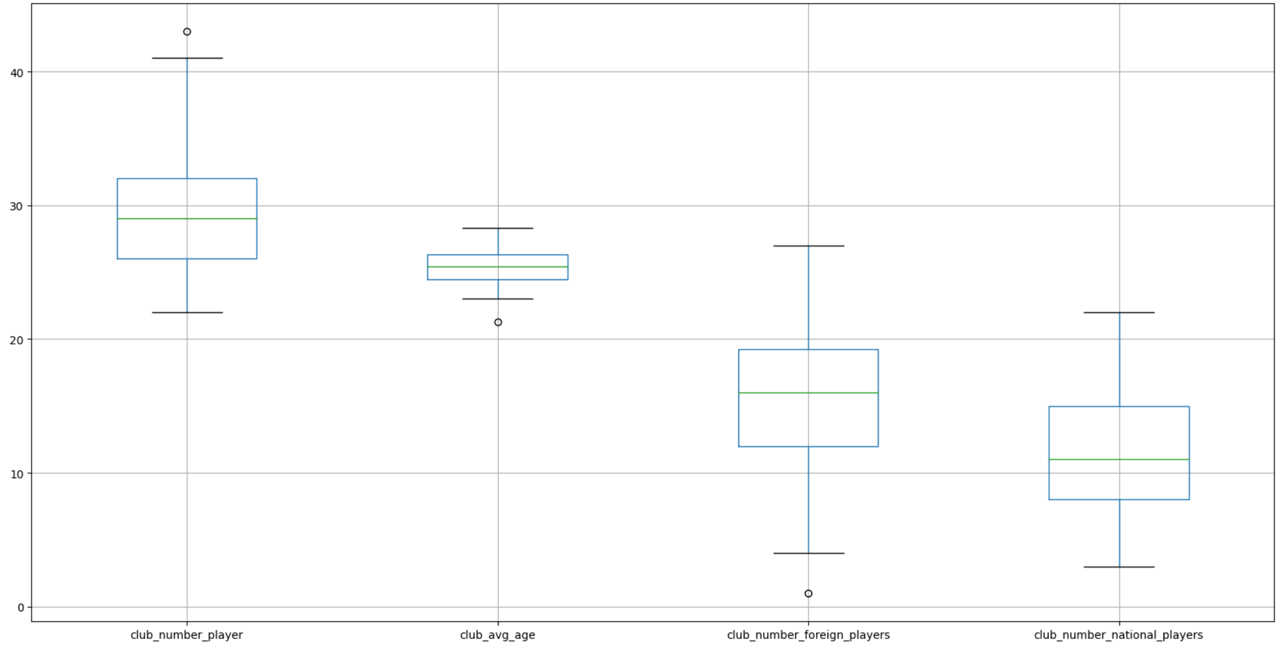
…preprocessing changes the data and introduces new bias.

Matplotlib Boxplot

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.boxplot.html

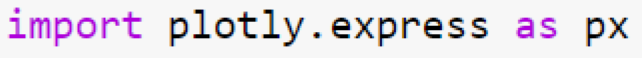


```
fig = plt.figure(figsize=(20,10))
df_clubs[["club_number_player","club_avg_age","club_number_foreign_players",
          "club_number_national_players"]].boxplot()
```



Boxplot graphically depicts groups of **numerical** data:

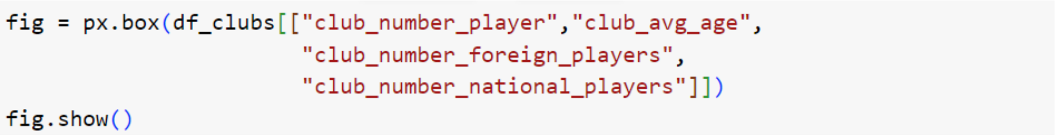
- Median
- Q1 and Q3 quartiles
- 1.5 * IQR (IQR = Q3 - Q1)
- Dots represent outliers



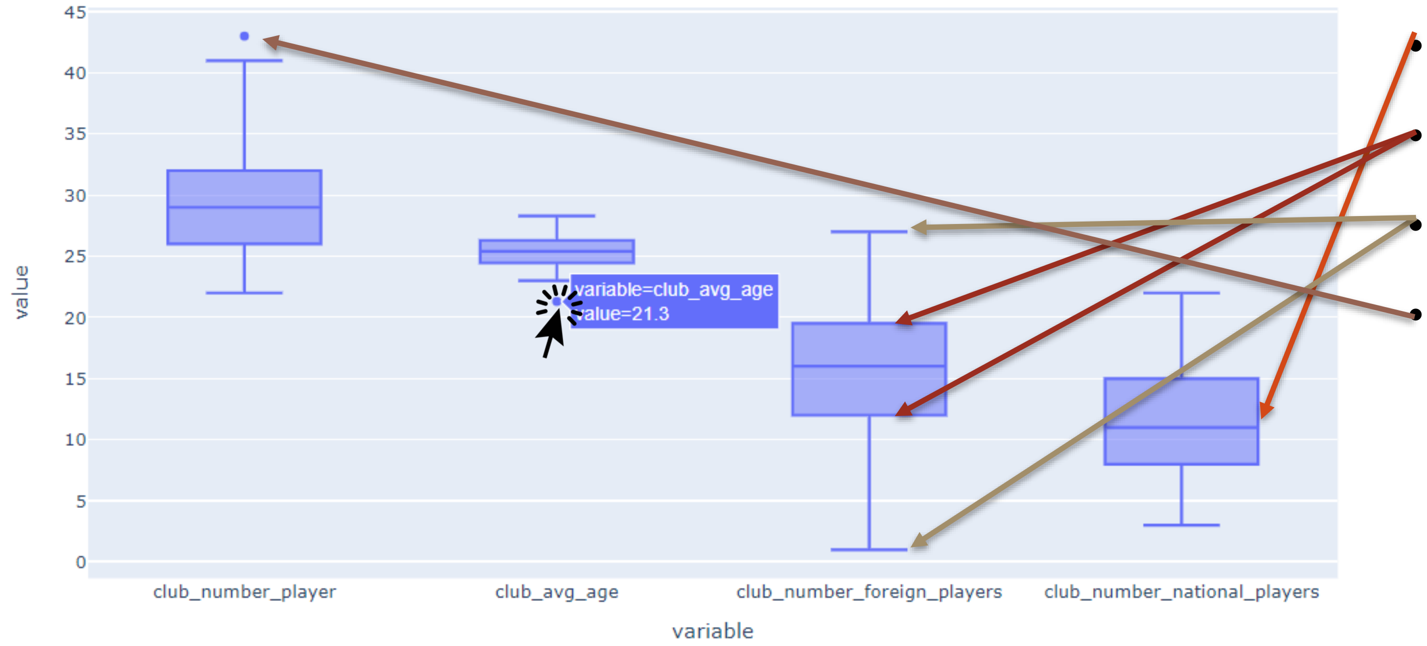
```
import plotly.express as px
```

Plotly Boxplot

https://plotly.com/python/box-plots/

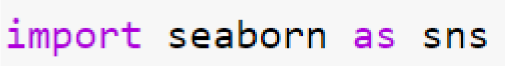


```
fig = px.box(df_clubs[["club_number_player","club_avg_age",
                       "club_number_foreign_players",
                       "club_number_national_players"]])
fig.show()
```



Boxplot graphically depicts groups of **numerical** data:

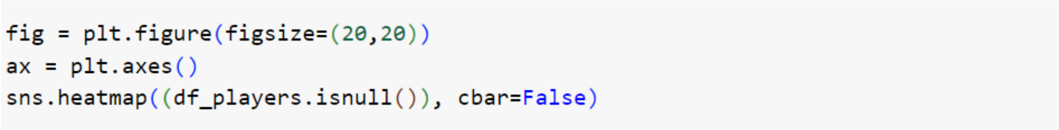
- Median
- Q1 and Q3 quartiles
- 1.5 * IQR (IQR = Q3 - Q1)
- Dots represent outliers



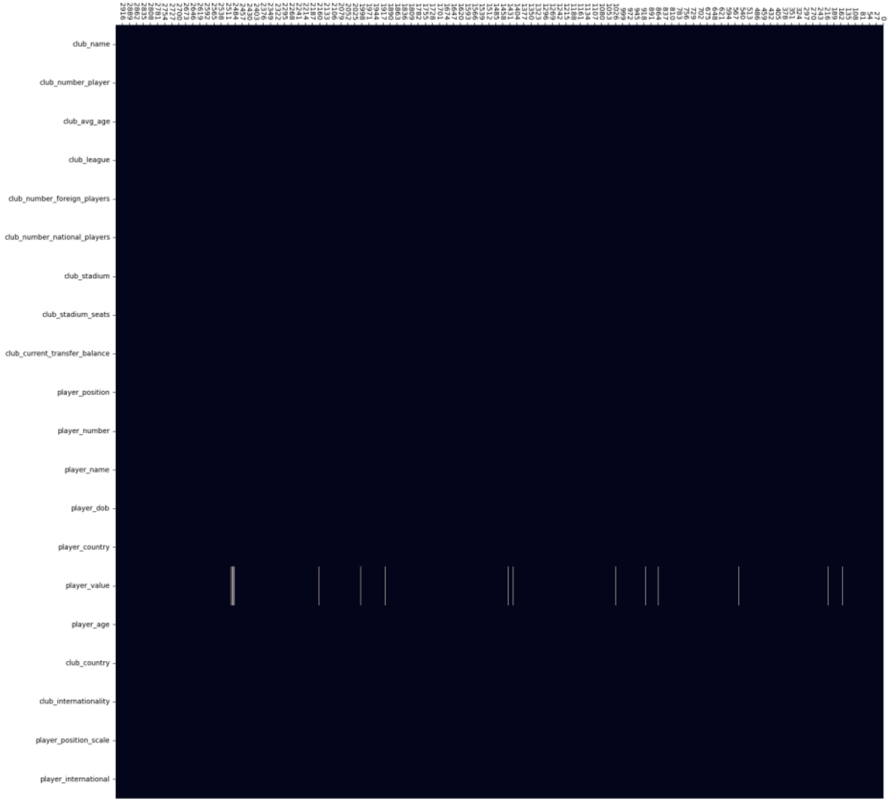
```
import seaborn as sns
```

NaN Heatmap

https://seaborn.pydata.org/generated/seaborn.heatmap.html



```
fig = plt.figure(figsize=(20,20))
ax = plt.axes()
sns.heatmap((df_players.isnull()), cbar=False)
```



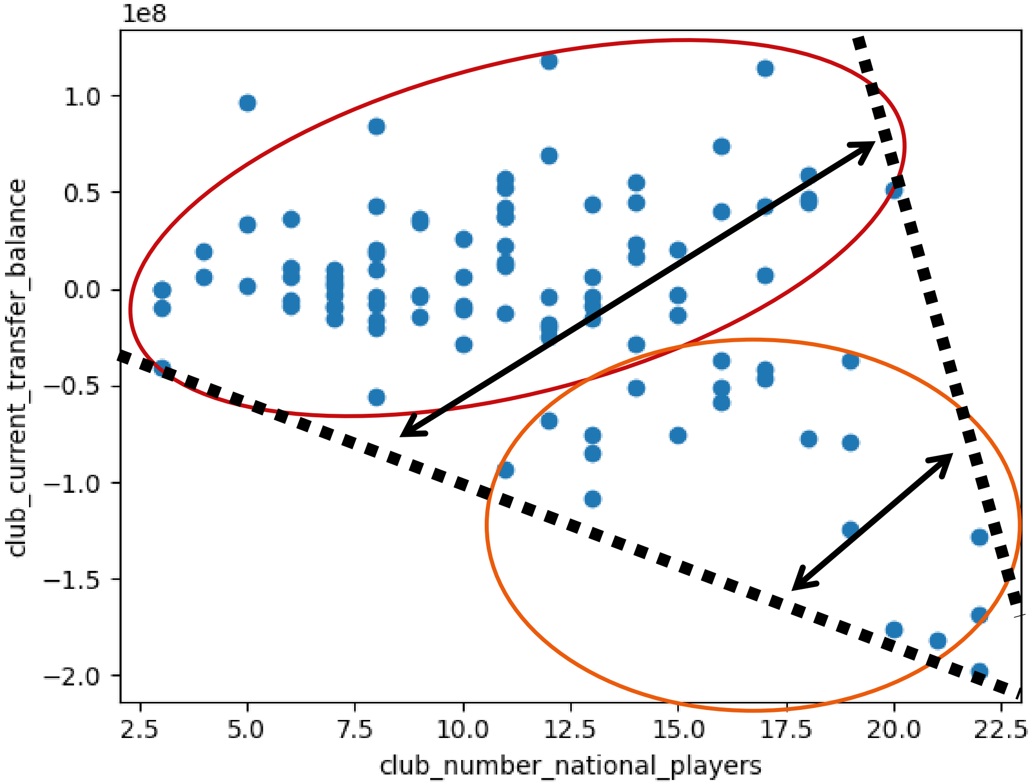
Graphical representation of data that uses a system of color coding to represent different values.

- Values can be **boolean** or **numeric**.
- **cbar** plots a colormap next to the graph.

Correlation

In reference of statistics slides by Yibi Huang (University of Chicago)

How would you interpret this scatter plot? Do you see a trend, tendency or groups between the two variables?



r = -0.38
(negative weak association)

Correlation *r* is a **numerical measure** ranging between [-1 (strong), 0(nothing), 1 (strong)].

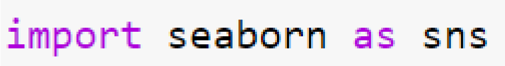
It describes the **direction** and **strength** of the **linear relationship** between two numerical variables.

Various methods exist, but most are based on the **sum** of **standard deviations** and **mean** between X and Y.

- Weak Association
  large spread of Y when X is known.
- Strong Association
  small spread of Y when X is known.

Important notes:

- Correlation is very sensitive to outliers!
- Correlation indicates association – not causation!



```
import seaborn as sns
```

Correlation Heatmap

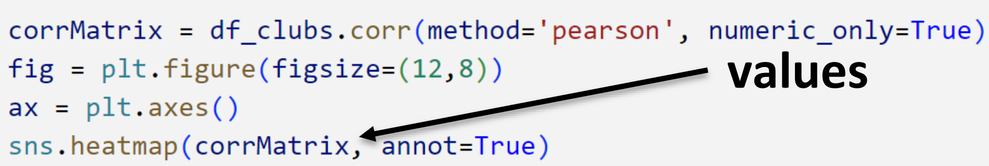
What does the correlation between two attributes mean?

1

0

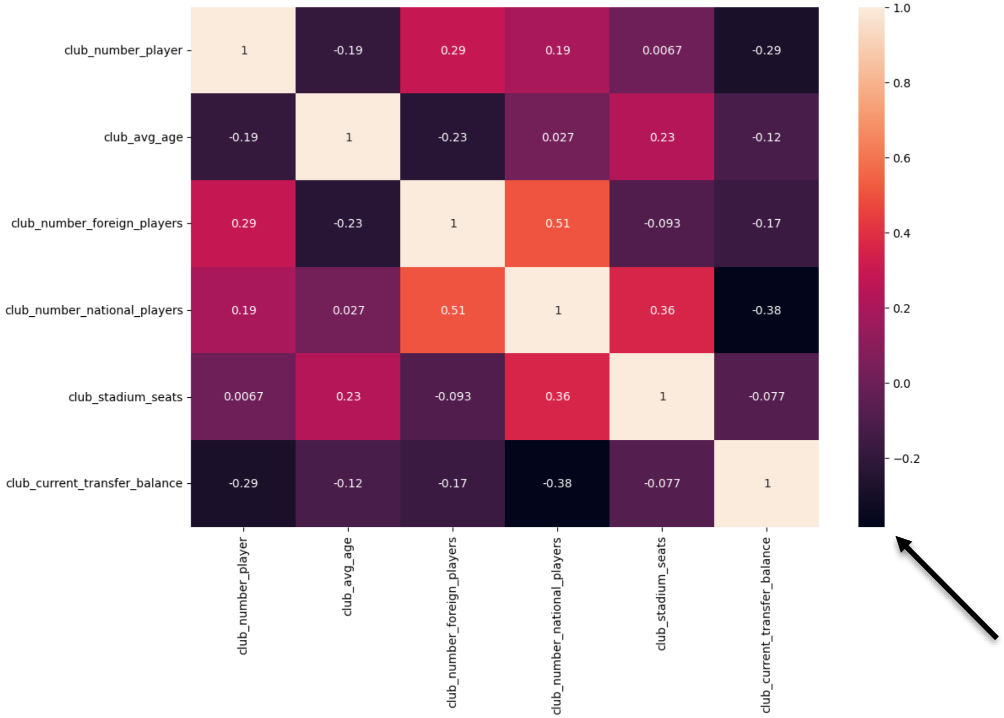
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.corr.html

https://seaborn.pydata.org/generated/seaborn.heatmap.html



```
corrMatrix = df_clubs.corr(method='pearson', numeric_only=True)
fig = plt.figure(figsize=(12,8))
ax = plt.axes()
sns.heatmap(corrMatrix, annot=True)
```

values



cbar

One famous correlation factor implemented into Pandas DataFrame is the **Pearson coefficient**.

Linear ratio between the covariance of two variables and the product of their standard deviations.

$$cov_{x,y} = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{N-1}$$

- **Bar** mean of elements
- **N** number of values

Please take these considerations with a little grain of salt ☺

Scenario	Data Preprocessing	Data Transformation	Data Visualization
Few attributes	Attribute Integration	Attribute Generalization	Detailed & Interactive
Few instances	Instance Integration		Detailed & Interactive
Many attributes	Attribute Reduction	Normalization for Analysis Attribute Summarization	Radar, Heatmaps, Matrix-Plots, XYZ-axis with hue and markers
Many instances	Sampling Outlier Analysis NaN Strategy	Grouping and Aggregation	Box-Plot, Scatter, Heatmaps, Histograms, Bubble
Numerical Data	Homogenous Formatting	Aggregation (Grouping with categorical data) Normalization	Scatter, Line, Waterfall, Violin, Correlation(!)
Categorical Data	Scaling	Attribute Generalization Aggregation (Counting)	Stacked Bar, Pie, Donut, Sunburst, Parallel Sets

Takeaways

- **Matplotlib** is the cross-platform **basis** for **most library extensions** in other statistical softwares.
- If your **data set** is **large**, make use of **heatmaps** and **pair plots** (matrix) to preprocess, transform, and find interesting patterns.
- Do not underestimate **data transformation** steps (use them), as they change your dataset and **help you deliver** a **message**.
- Learn from **examples**, ask for early **feedback** from your peers, **explore** other (interactive) **libraries**.

Outlook

- In the next weeks, we will see how things work in R.

- After Christmas holidays, we will have our project poster and recap day.
- Ilias Survey in January to schedule the oral exams.

See you again next week!

Questions?