Data Visualization

Advanced and Interactive Plots

CentraleDigitalLab@Nice

Solutions - Practical Part

01__practical_exercises_solutions.ipynb

Boxenplots

The Boxen plot is an advanced box plot that represents **complex data distributions** more effectively by **displaying additional quantiles**. Designed to visualize data with heavy tail distributions.

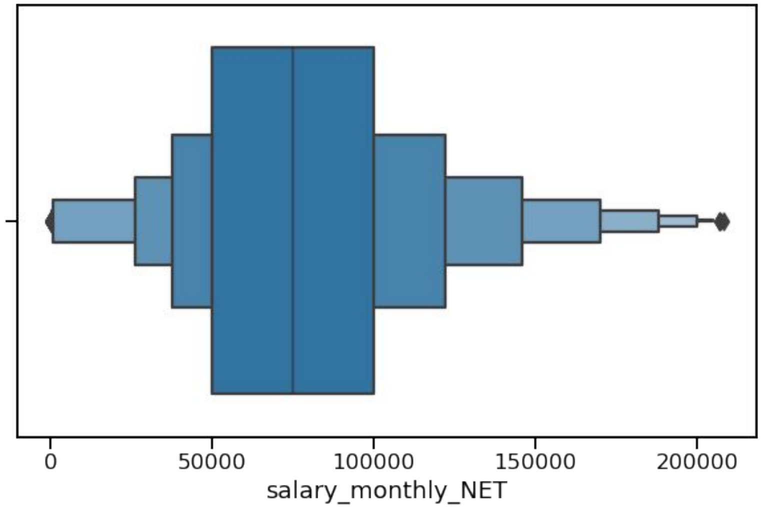
Central Box: Like the traditional box plot, the Boxen plot has a box that represents the interquartile range (IQR), marking the Q1 (25th percentile) and Q3 (75th percentile) boundaries.

Whiskers: Boxen plots have multiple "whiskers" extending from the central box, which show more quartiles or percentiles than a standard box plot.

Outliers: Points beyond the "whiskers" are considered outliers, similar to a box plot, but with more granular control over the definition of "outlier."

Boxenplots

The Boxen plot is an advanced box plot that represents **complex data distributions** more effectively by **displaying additional quantiles**. Designed to visualize data with heavy tail distributions.



Violinplots

A Violin Plot **combines** features of a **box plot** and a **kernel density plot** to provide a rich, compact display of data distribution. It's useful for comparing multiple categories and visualizing the entire probability density of the data.

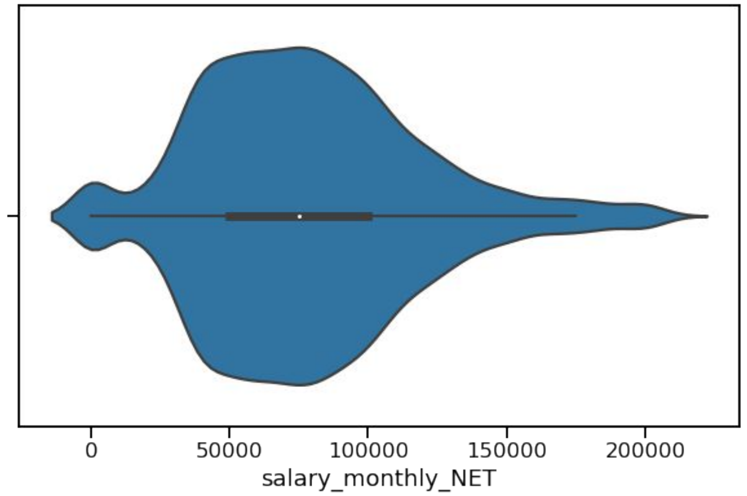
Density Curve: The outer shape represents the kernel density estimate of the data distribution, effectively showing the probability density at different values.

Inner Box Plot: Within the density curve, a miniature box plot is often included, marking the median and the interquartile range (IQR).

Width: The width of the "violin" at different values indicates the density of the data at that value, making it easier to visualize peaks and valleys in the data distribution.

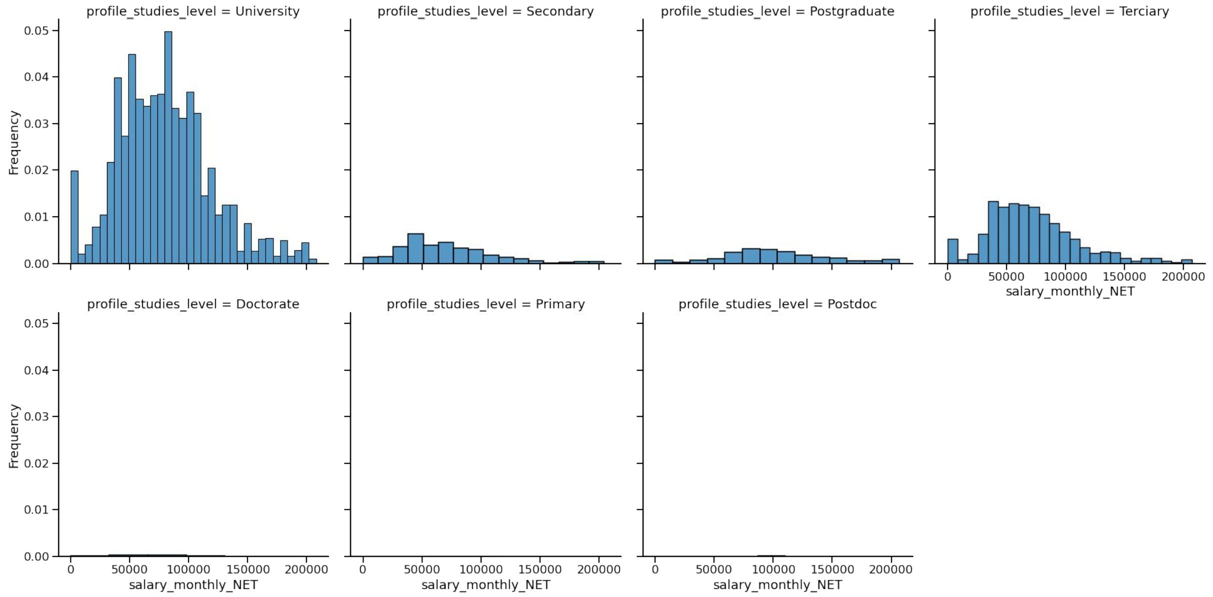
Violinplots

A Violin Plot **combines** features of a **box plot** and a **kernel density plot** to provide a rich, compact display of data distribution. It's useful for comparing multiple categories and visualizing the entire probability density of the data.



Subplots and FacetGrid

Subplots and FacetGrids offer ways to **create multiple plots** within the same figure. They are powerful tools for comparing different slices of data across one or more subplots.



Plot Styles: Coordinate Properties

Keywords: `x, y, xmin, xmax, ymin, ymax`

`x, y`: Define mark's **horizontal** and **vertical** position.

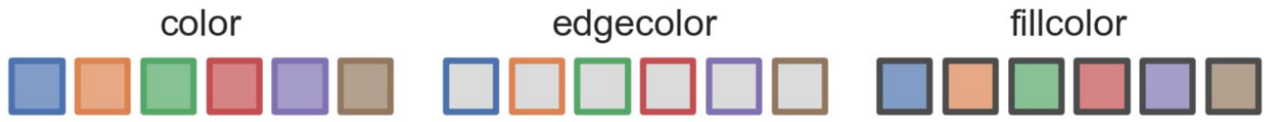
`xmin, xmax, ymin, ymax`: Specify the **span** or **range** for marks.

Plot Styles: Color Properties

Keywords: `color`

`color`: Sets both **edge** and **fill** of a mark.

Scales: Nominal (unordered hues), Continuous (gradients).

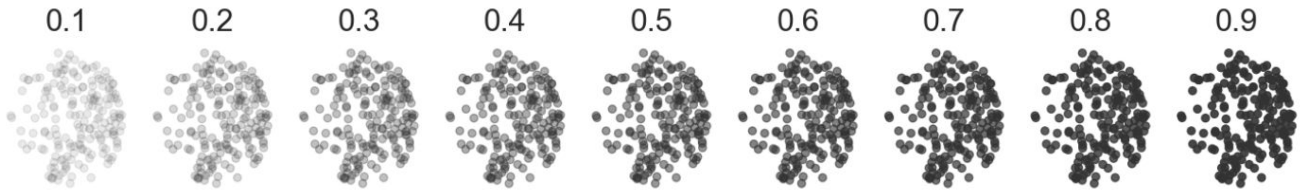


Plot Styles: Alpha, Fillalpha, Edgealpha

Keywords: `alpha, fillalpha, edgealpha`

`alpha`: Controls **mark's opacity**.

`fillalpha, edgealpha`: Fine-tune **opacity for edge and fill**.

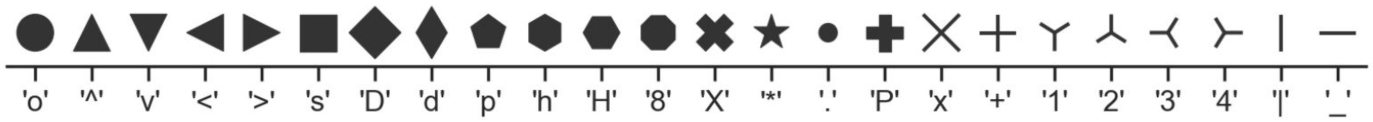


Plot Styles: Style Properties

Keywords: `marker, linestyle, edgestyle`

`marker`: Specifies the **shape of dot marks**.

`linestyle, edgestyle`: **Control line "dashing"** patterns.



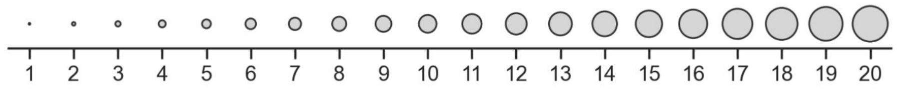
Plot Styles: Size Properties

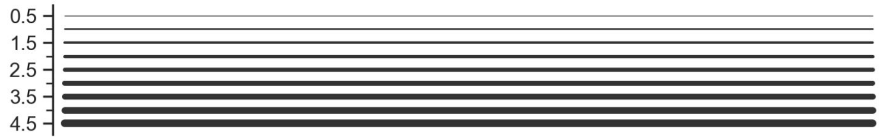
Keywords: `pointsize, linewidth, edgewidth, stroke`

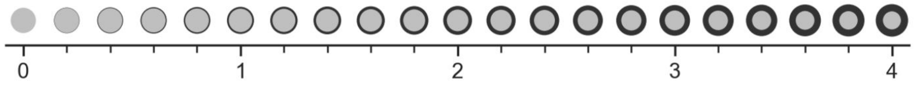
`pointsize`: **Diameter** of dot marks.

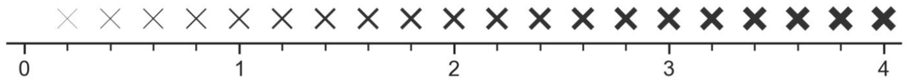
`linewidth`: **Thickness** of line marks.

`edgewidth, stroke`: Similar to **linewidth** but for **edge/fill marks**.



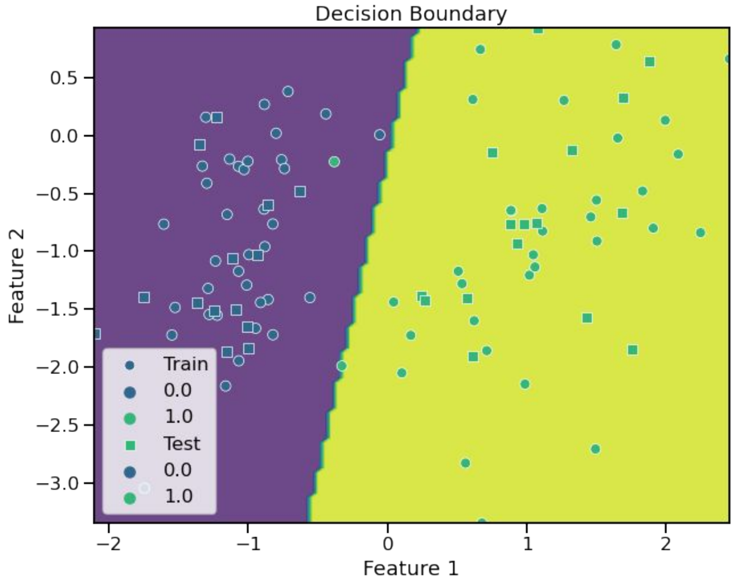






Decision Boundary Plots

A Decision Boundary Plot visually represents the areas where a classification model makes different predictions. It **separates the feature space into regions** assigned to different classes.



Confusion Matrices

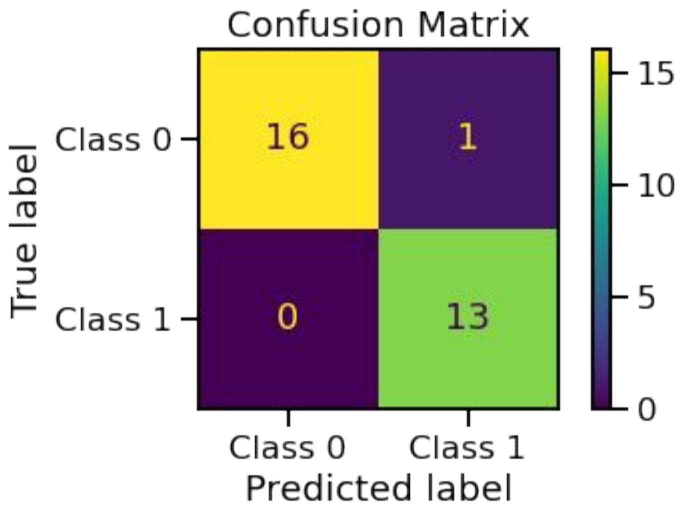
A Confusion Matrix is a table that **quantifies the performance of a classification** model by comparing its predicted and actual labels.

True Positive (TP): Correctly identified as positive.

True Negative (TN): Correctly identified as negative.

False Positive (FP): Incorrectly identified as positive.

False Negative (FN): Incorrectly identified as negative.



Demo with notebook
04_advanced_plots.ipynb

From Seaborn to Plotly

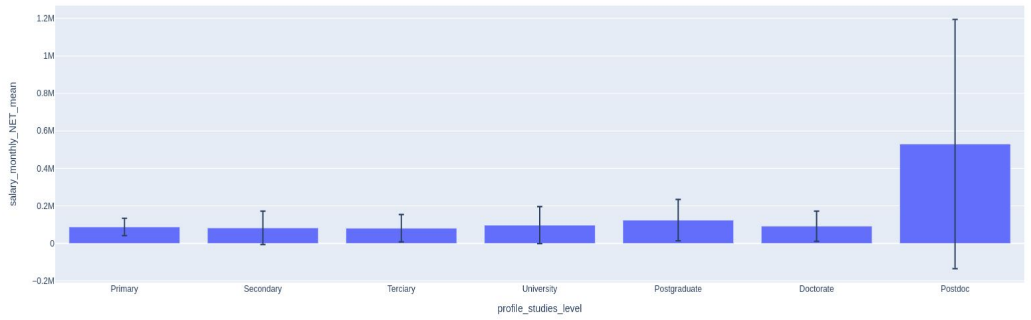
The plotly library is an interactive open-source plotting library that supports the creation of **more personalized plots than seaborn** but at the same time with a harder learning curve.

Components:

- **Plotly Express**: High-level API for simple plots.
- **Figure Data Structure**: Low-level API for detailed customization.

Plotly Express

Plotly Express is a high-level interface for creating a wide range of interactive visualizations quickly and easily.

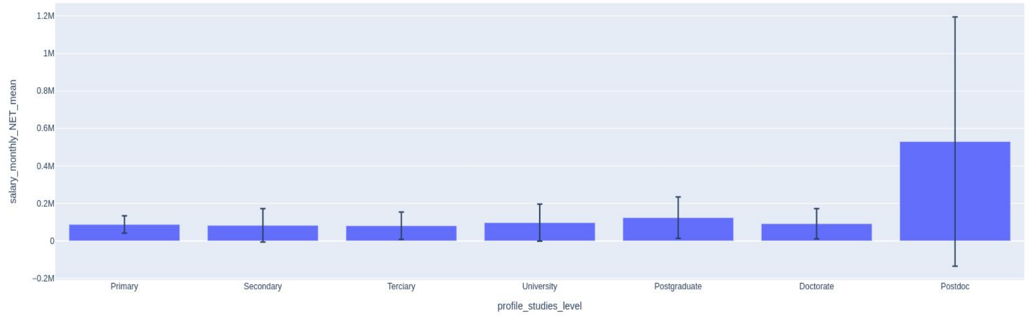


```
fig = px.bar(
df_studies_level_mean,
x='profile_studies_level',
y='salary_monthly_NET_mean',
error_y="salary_monthly_NET_std")
fig.show()
```

Plotly Express

Plotly Express is a high-level interface for creating a wide range of interactive visualizations quickly and easily.

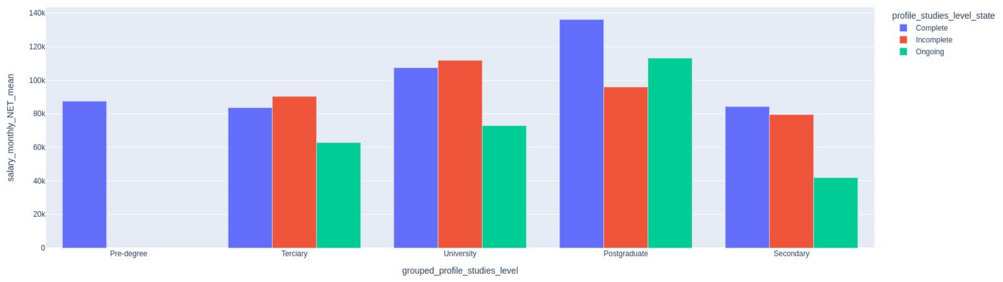
IMPORTANT! Sometimes we need to calculate the aggregation



```
fig = px.bar(
df_studies_level_mean,
x='profile_studies_level',
y='salary_monthly_NET_mean',
error_y="salary_monthly_NET_std")
fig.show()
```

Plotly Express

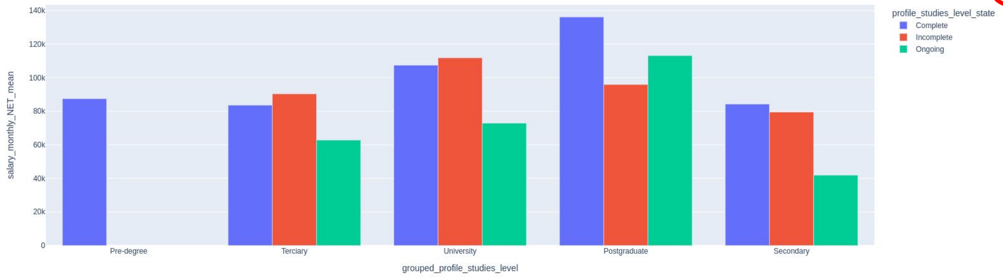
Plotly Express is a high-level interface for creating a wide range of interactive visualizations quickly and easily.



```
fig = px.bar(
df_grouped_studies_level_mean,
x='profile_studies_level',
y='salary_monthly_NET_mean',
color='profile_studies_level_state',
barmode='group')
fig.show()
```

Plotly Express

Plotly Express is a high-level interface for creating a wide range of interactive visualizations quickly and easily.



Dataframe with the studies level, level state, and salary mean

```
fig = px.bar(
df_grouped_studies_level_mean,
x='profile_studies_level',
y='salary_monthly_NET_mean',
color='profile_studies_level_state',
barmode='group')
fig.show()
```

Plotly Figure

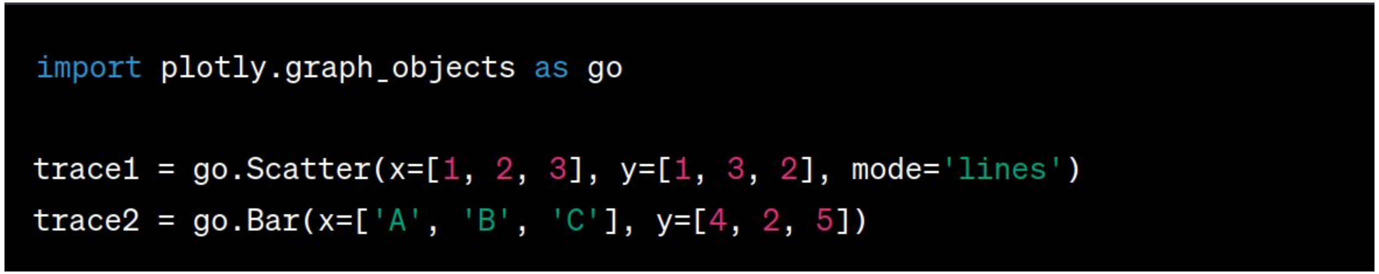
The Plotly Figure Data Structure is a more detailed and flexible way to create, configure, and update Plotly visualizations.

Main Parts:

- Data: Described by a list of "traces" (like scatter, line, etc.).
- Layout: Describes how the chart looks (titles, axis labels, etc.).

Plotly Figure: Data - Traces

The Plotly Figure Data Structure is a more detailed and flexible way to create, configure, and update Plotly visualizations.



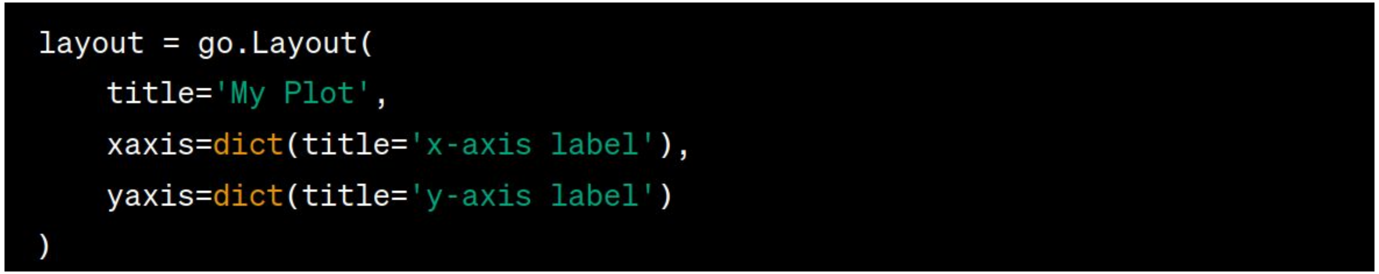
```python
import plotly.graph_objects as go

trace1 = go.Scatter(x=[1, 2, 3], y=[1, 3, 2], mode='lines')
trace2 = go.Bar(x=['A', 'B', 'C'], y=[4, 2, 5])
```

`Scatter` and `Bar` are types of traces.

The **x** and **y** parameters define data points.

Plotly Figure: Layout - Styling

The Plotly Figure Data Structure is a more detailed and flexible way to create, configure, and update Plotly visualizations.



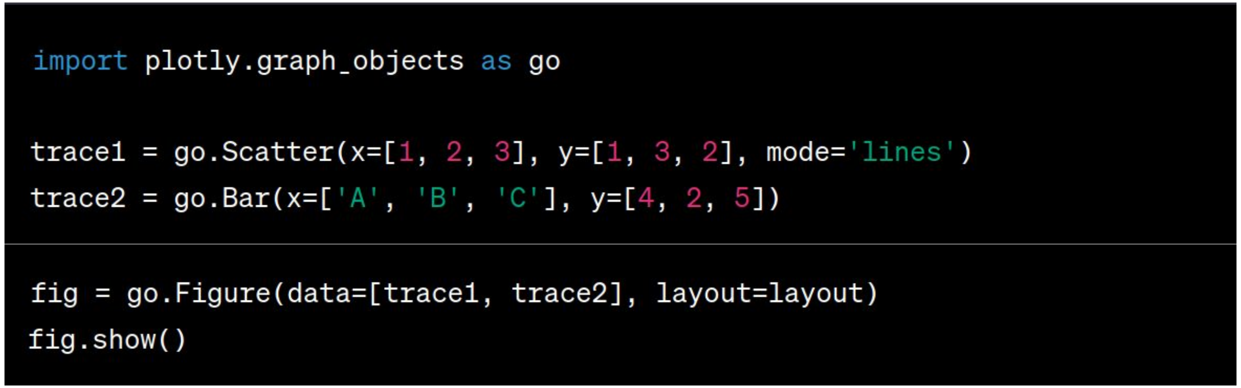
```
layout = go.Layout(
    title='My Plot',
    xaxis=dict(title='x-axis label'),
    yaxis=dict(title='y-axis label')
)
```

`title` specifies the chart title.

`xaxis` and **`yaxis`** are dictionaries for axis styling.

Plotly Figure: Layout - Styling

The Plotly Figure Data Structure is a more detailed and flexible way to create, configure, and update Plotly visualizations.



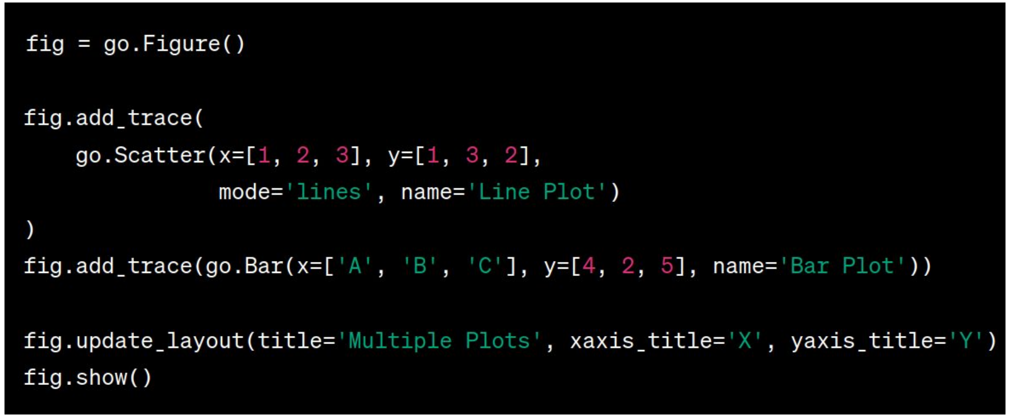
```python
import plotly.graph_objects as go

trace1 = go.Scatter(x=[1, 2, 3], y=[1, 3, 2], mode='lines')
trace2 = go.Bar(x=['A', 'B', 'C'], y=[4, 2, 5])

fig = go.Figure(data=[trace1, trace2], layout=layout)
fig.show()
```

`data` accepts a list of traces. `layout` applies the layout styling.

`show()` renders the plot.

Plotly Figure: Multi-Plot Example

The Plotly Figure Data Structure is a more detailed and flexible way to create, configure, and update Plotly visualizations.



```python
fig = go.Figure()

fig.add_trace(
    go.Scatter(x=[1, 2, 3], y=[1, 3, 2],
               mode='lines', name='Line Plot')
)
fig.add_trace(go.Bar(x=['A', 'B', 'C'], y=[4, 2, 5], name='Bar Plot'))

fig.update_layout(title='Multiple Plots', xaxis_title='X', yaxis_title='Y')
fig.show()
```

`add_trace()` lets you add multiple plots to the same figure.

`update_layout()` allows for updating layout elements dynamically.

Grouping and Aggregation

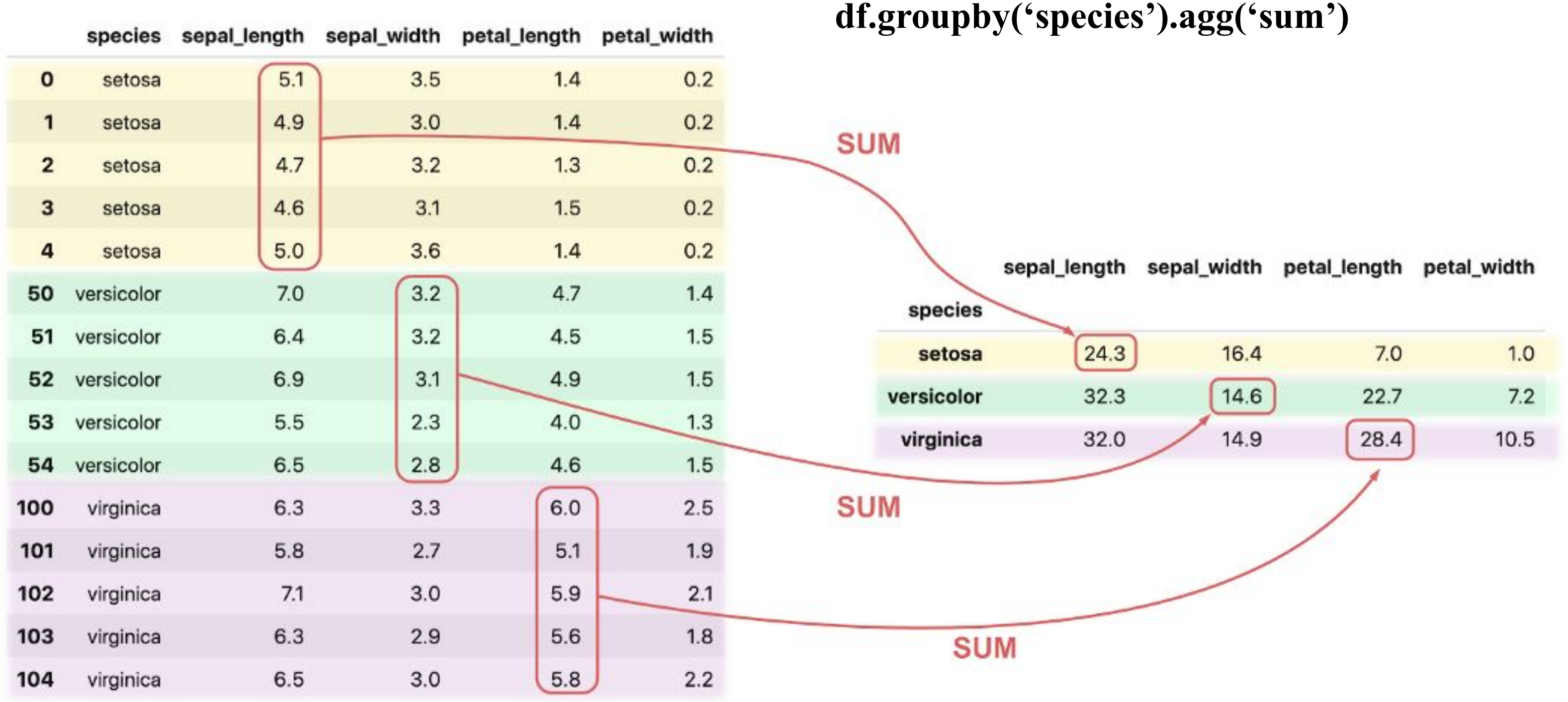
- groupby:
  - Takes a series of columns ***A***, ***B***, ***C***
  - For each combination of column values ***(a, b, c)***, group the rows that have those values.

Grouping and Aggregation

- groupby:
  - Takes a series of columns ***A***, ***B***, ***C***
  - For each combination of column values ***(a, b, c)***, group the rows that have those values.
- agg:
  - Takes a function ***F***
  - For each group of rows, apply the function ***F*** to each column.

Grouping and Aggregation

df.groupby('species').agg('sum')



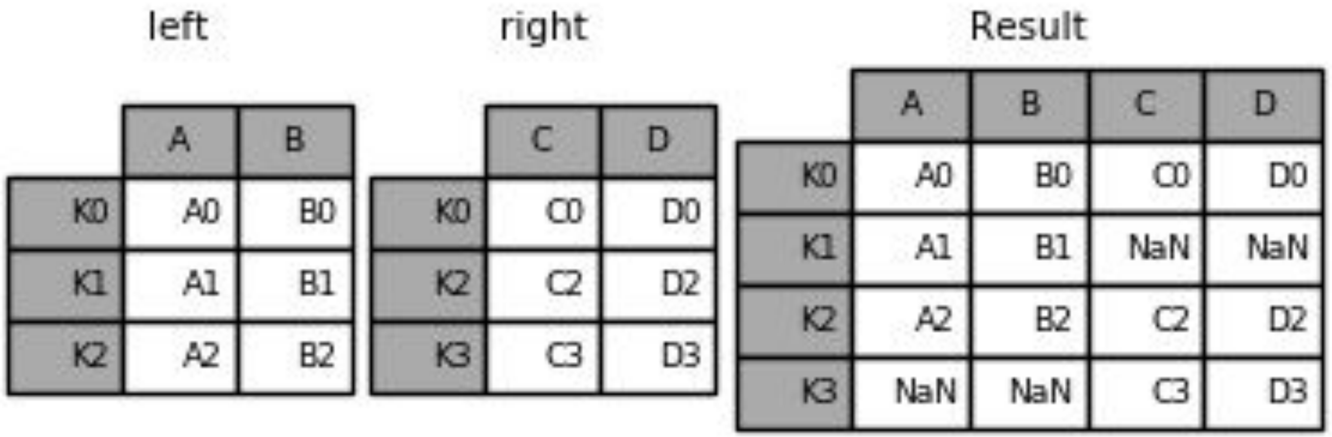
	species	sepal_length	sepal_width	petal_length	petal_width
0	setosa	5.1	3.5	1.4	0.2
1	setosa	4.9	3.0	1.4	0.2
2	setosa	4.7	3.2	1.3	0.2
3	setosa	4.6	3.1	1.5	0.2
4	setosa	5.0	3.6	1.4	0.2
50	versicolor	7.0	3.2	4.7	1.4
51	versicolor	6.4	3.2	4.5	1.5
52	versicolor	6.9	3.1	4.9	1.5
53	versicolor	5.5	2.3	4.0	1.3
54	versicolor	6.5	2.8	4.6	1.5
100	virginica	6.3	3.3	6.0	2.5
101	virginica	5.8	2.7	5.1	1.9
102	virginica	7.1	3.0	5.9	2.1
103	virginica	6.3	2.9	5.6	1.8
104	virginica	6.5	3.0	5.8	2.2

species	sepal_length	sepal_width	petal_length	petal_width
setosa	24.3	16.4	7.0	1.0
versicolor	32.3	14.6	22.7	7.2
virginica	32.0	14.9	28.4	10.5

https://towardsdatascience.com/how-to-use-the-split-apply-combine-strategy-in-pandas-groupby-29e0eb44b62e

Join and Merge

- df1.join(df2, how='outer')
  - Horizontally join the DataFrames and match the rows where the index value is the same



left

	A	B
K0	A0	B0
K1	A1	B1
K2	A2	B2

right

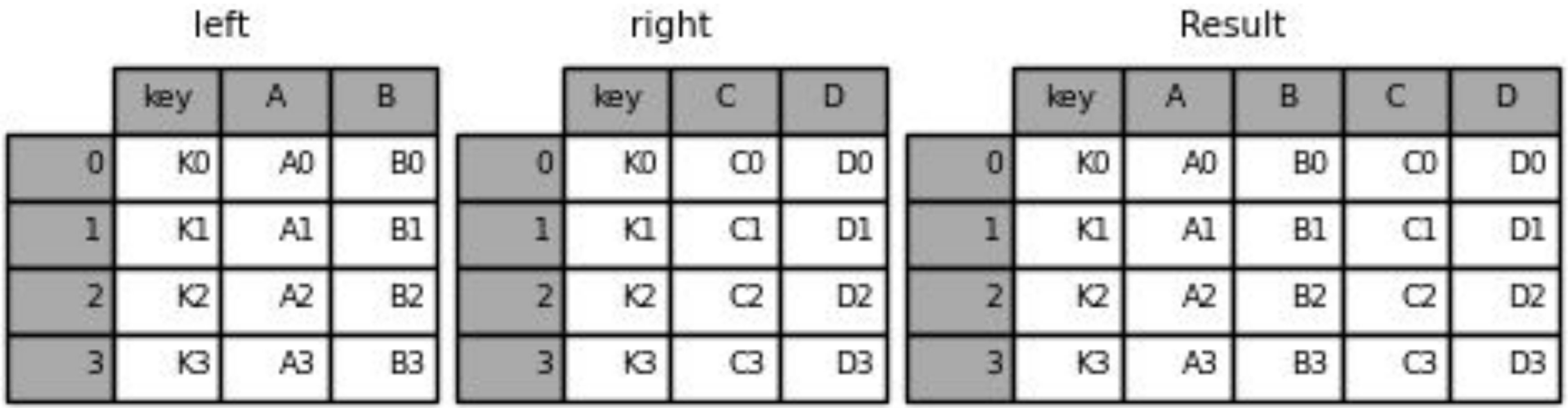
	C	D
K0	C0	D0
K2	C2	D2
K3	C3	D3

Result

	A	B	C	D
K0	A0	B0	C0	D0
K1	A1	B1	NaN	NaN
K2	A2	B2	C2	D2
K3	NaN	NaN	C3	D3

Join and Merge

- df1.merge(df2, on='key')
  - Same as join, but instead of comparing indexes, it compares a set of columns.



left

	key	A	B
0	K0	A0	B0
1	K1	A1	B1
2	K2	A2	B2
3	K3	A3	B3

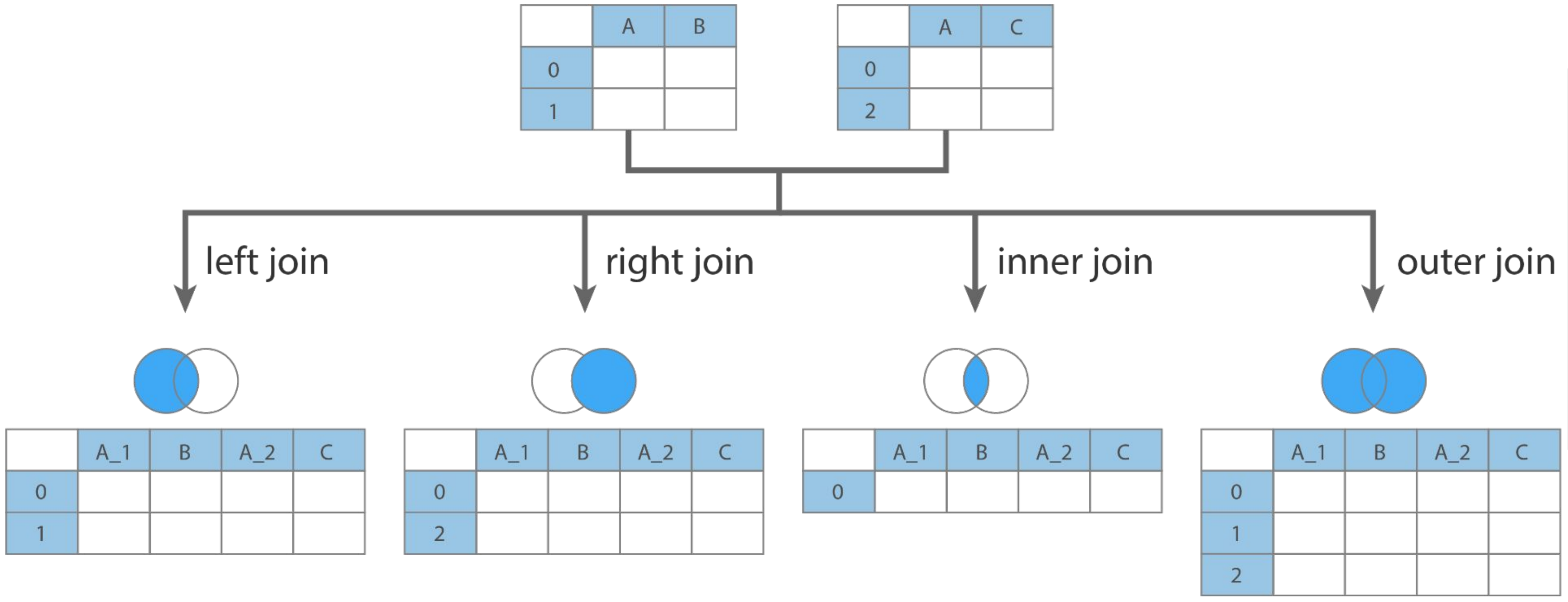
right

	key	C	D
0	K0	C0	D0
1	K1	C1	D1
2	K2	C2	D2
3	K3	C3	D3

Result

	key	A	B	C	D
0	K0	A0	B0	C0	D0
1	K1	A1	B1	C1	D1
2	K2	A2	B2	C2	D2
3	K3	A3	B3	C3	D3

Join and Merge



https://towardsdatascience.com/python-pandas-dataframe-join-merge-and-concatenate-84985c29ef78

Demo with notebook

05_plotly_vs_seaborn.ipynb