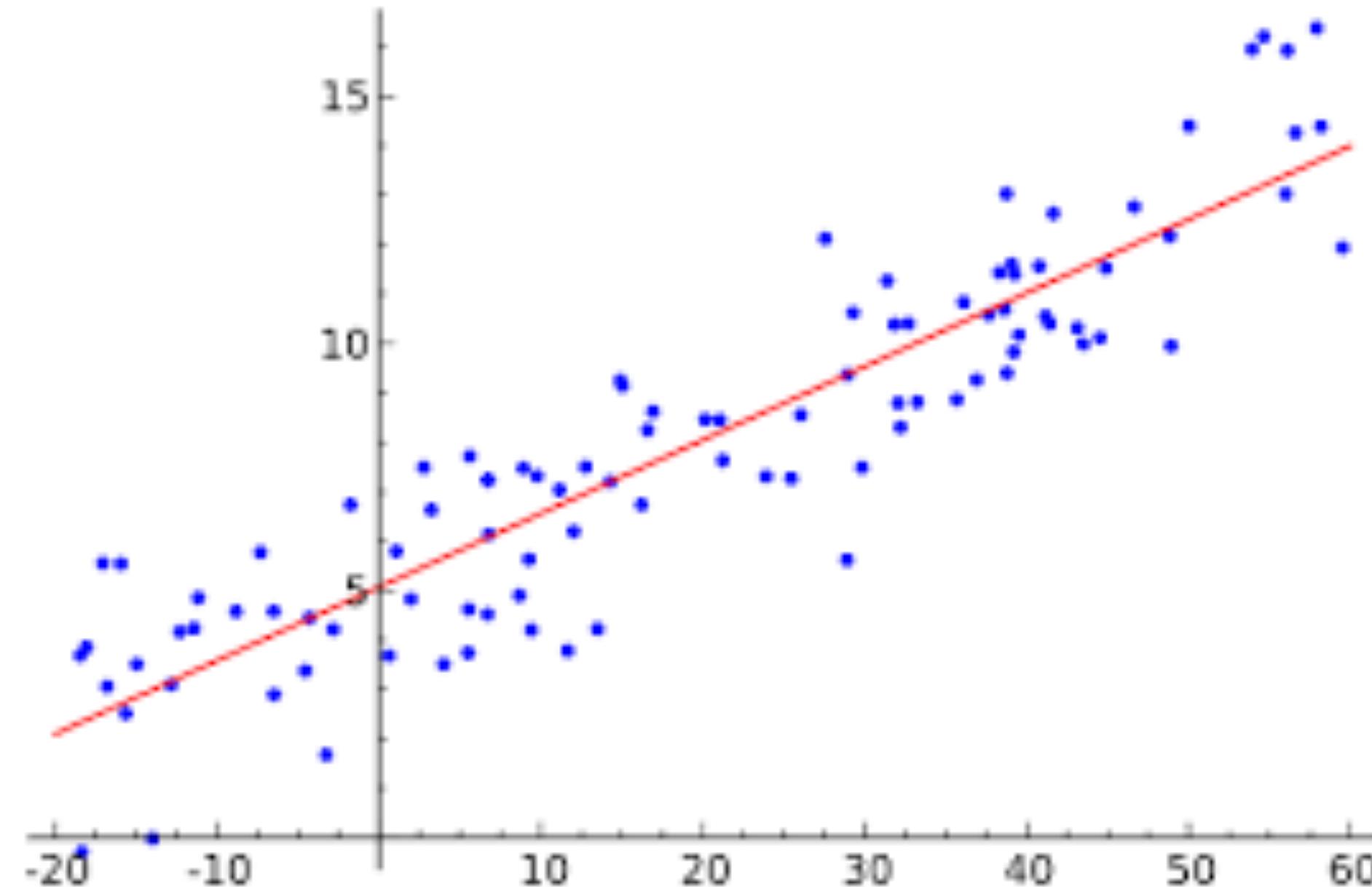


Data visualisation

Why create visuals?

- Visuals provide a better way for observation
- Explore data
- Discover patterns
- Discover trends
- Story-telling



Value of visualisation

- **Exploratory visualization**
 - Develop and assess hypotheses
 - Find patterns / Discover errors in data
 - Identify trends and clusters, spotting local patterns, evaluating modeling output
- **Explanatory Visualization**
 - Share and Present Results
 - Stimulate Research, Collaborations and Ideas

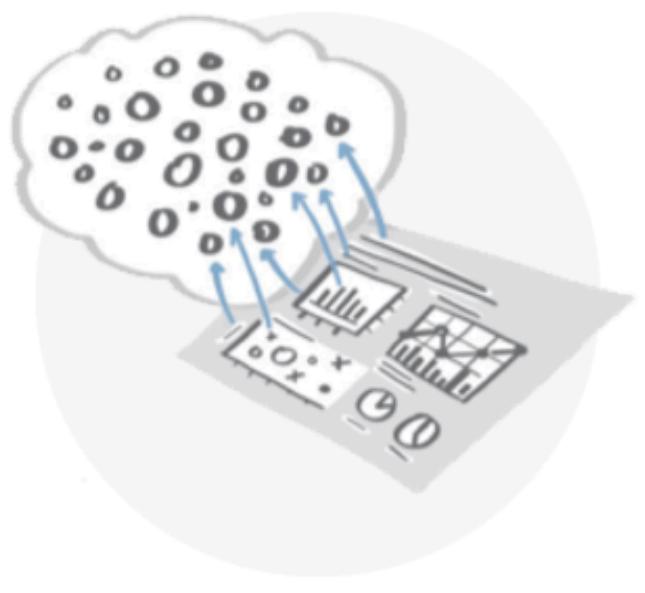
Ideas to story



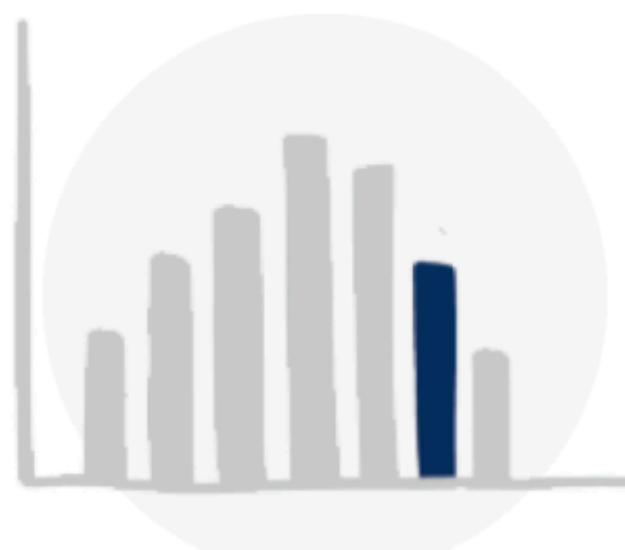
understand the
context



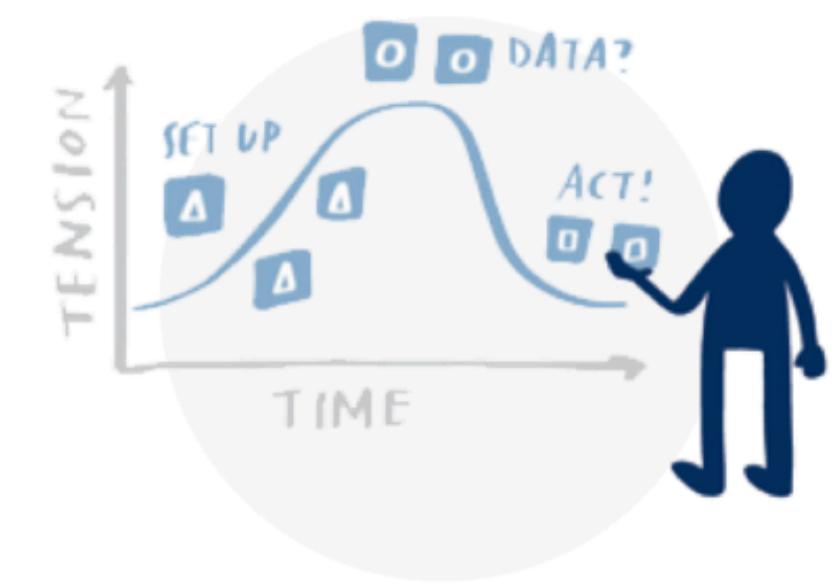
choose an
effective visual



**eliminate
clutter**



**focus
attention**



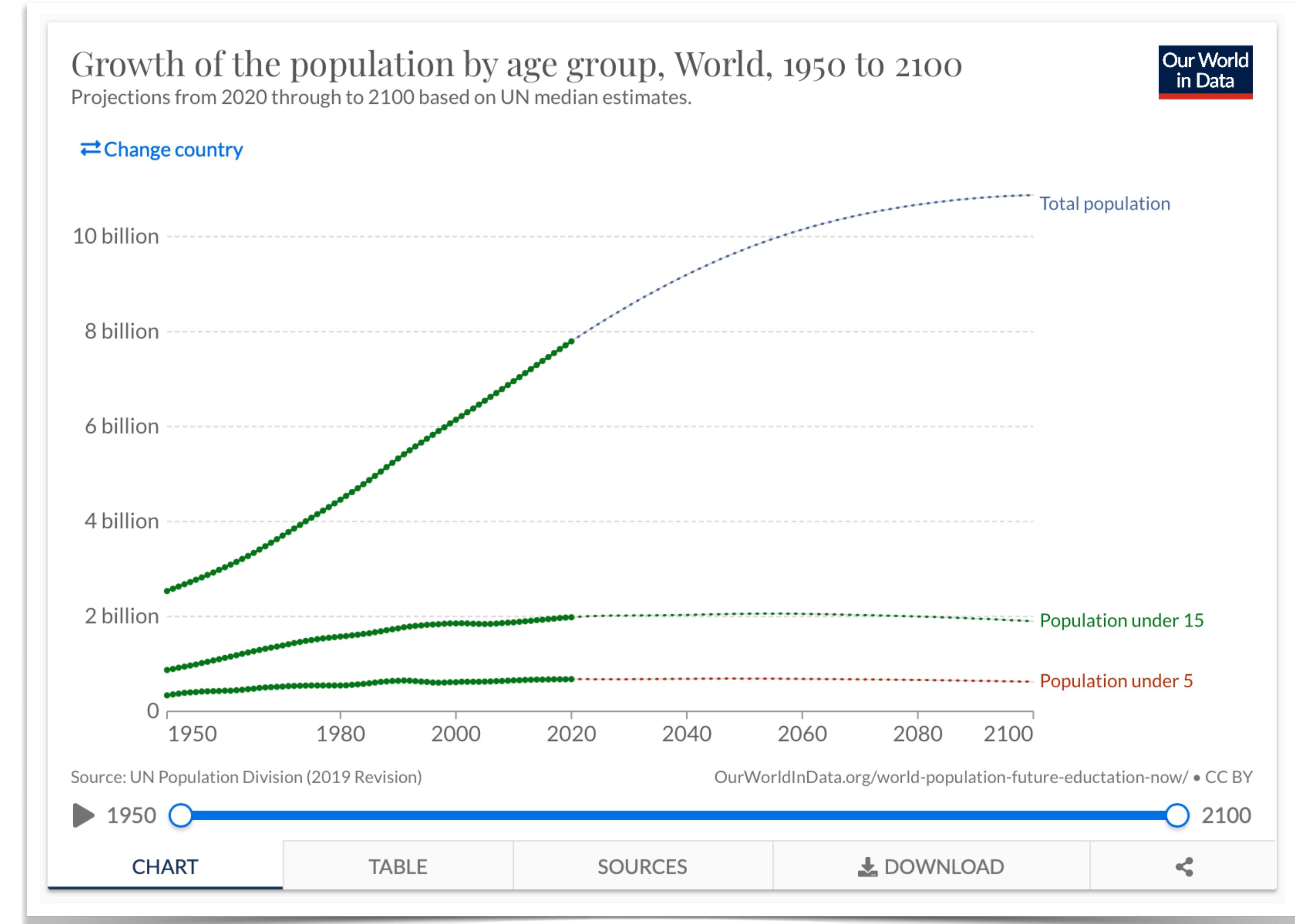
**tell a
story**

Data stories

- New York Times
- Nature/Lancet articles
- Wealth to scale
- Medium articles

Visualisation

- Static
- Dynamic
- Interactive



Which visualisation to use?

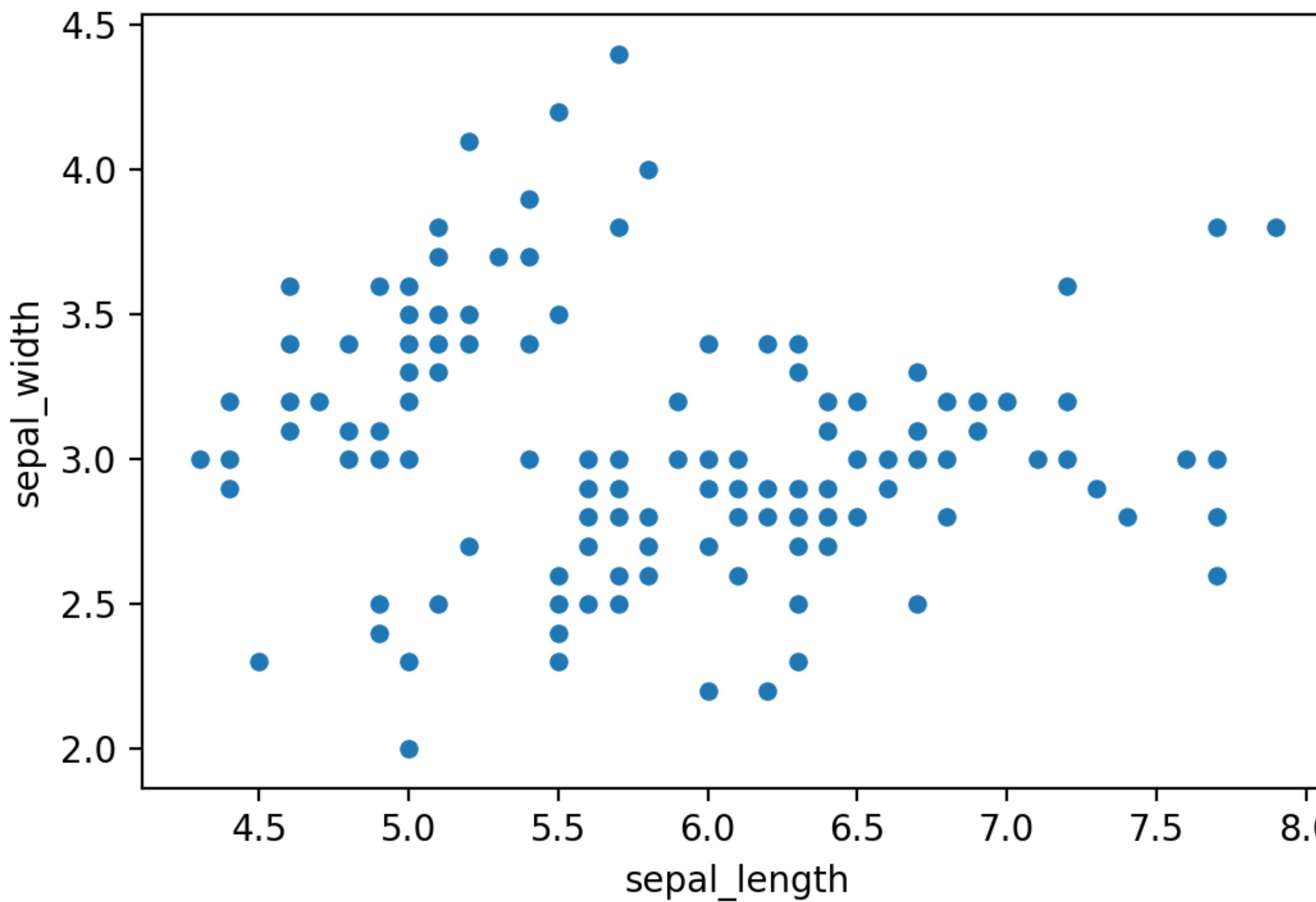
- Story
- Audience
- Size of the data
- Data types
- Functionality and relationship between data elements



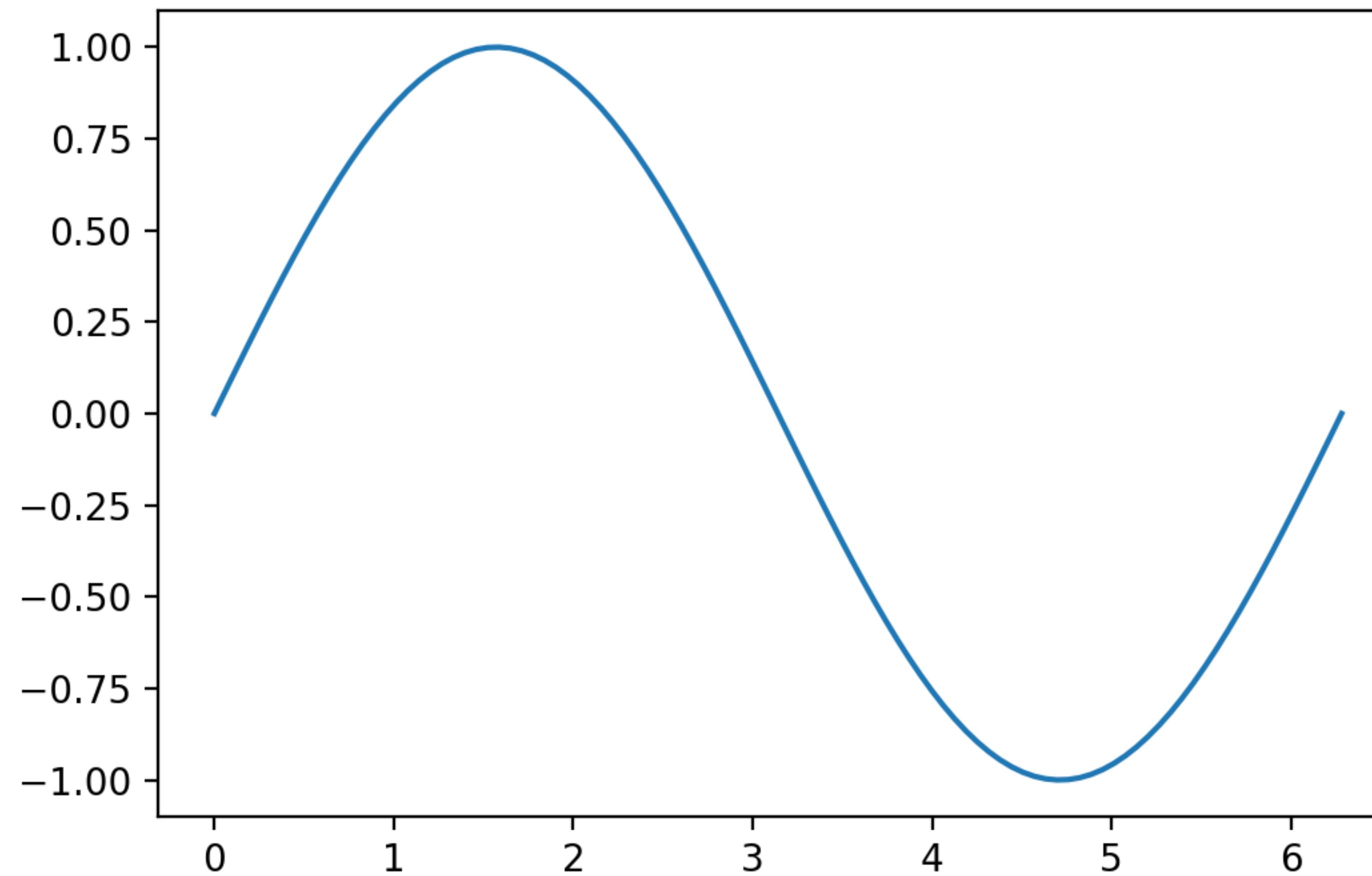
Data visualisation catalog

Basic plots

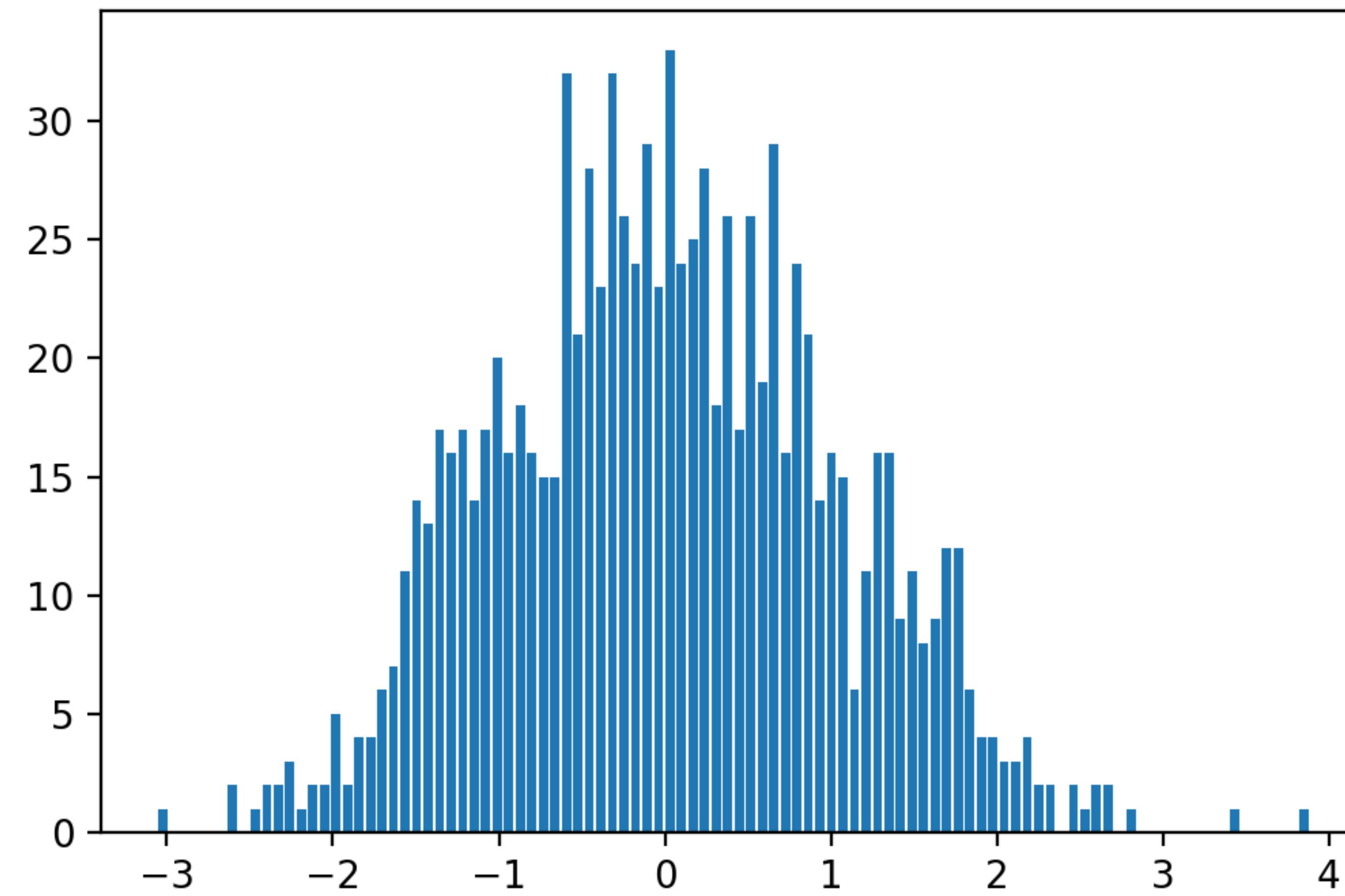
Scatter plot



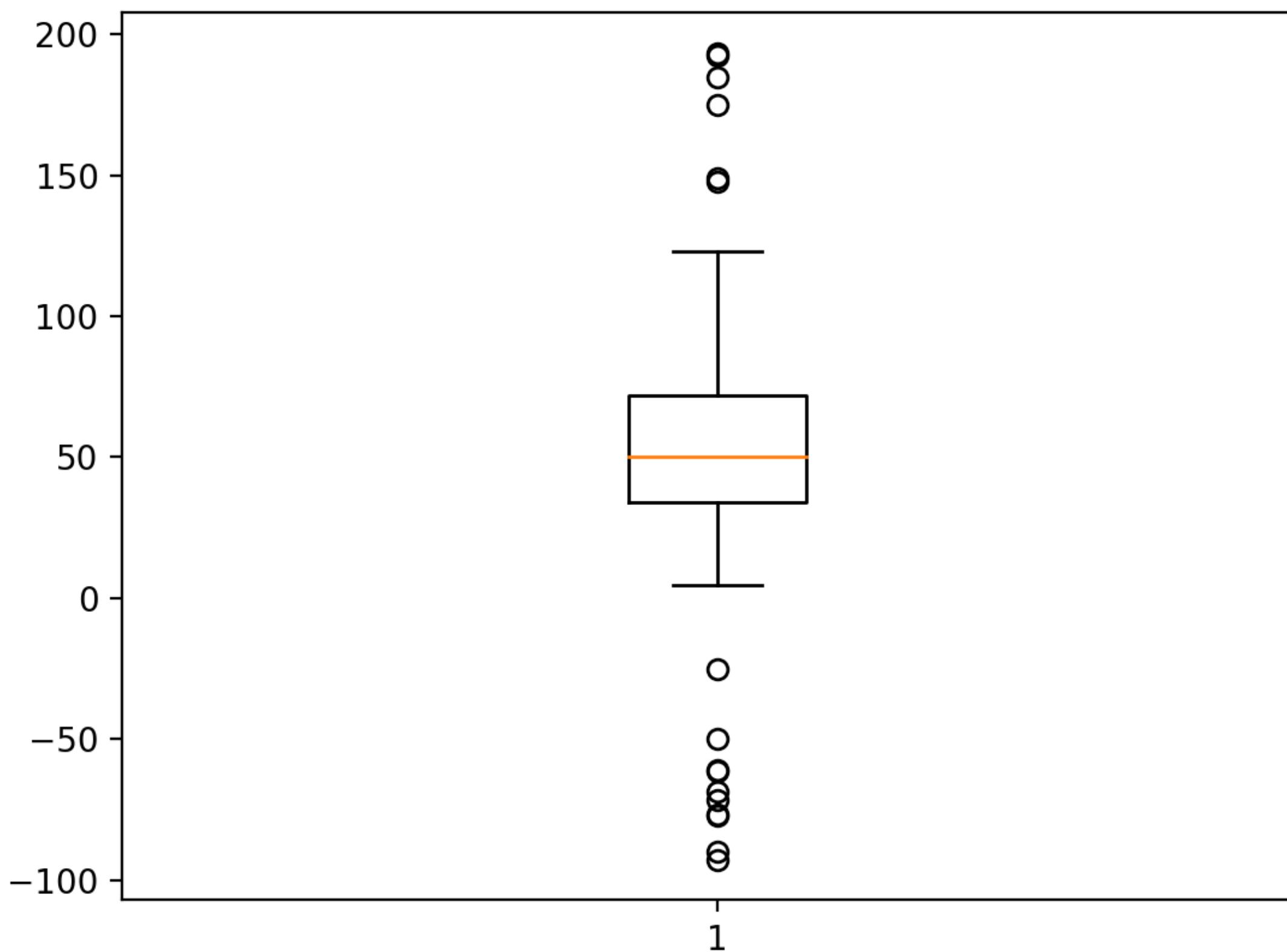
Line plot



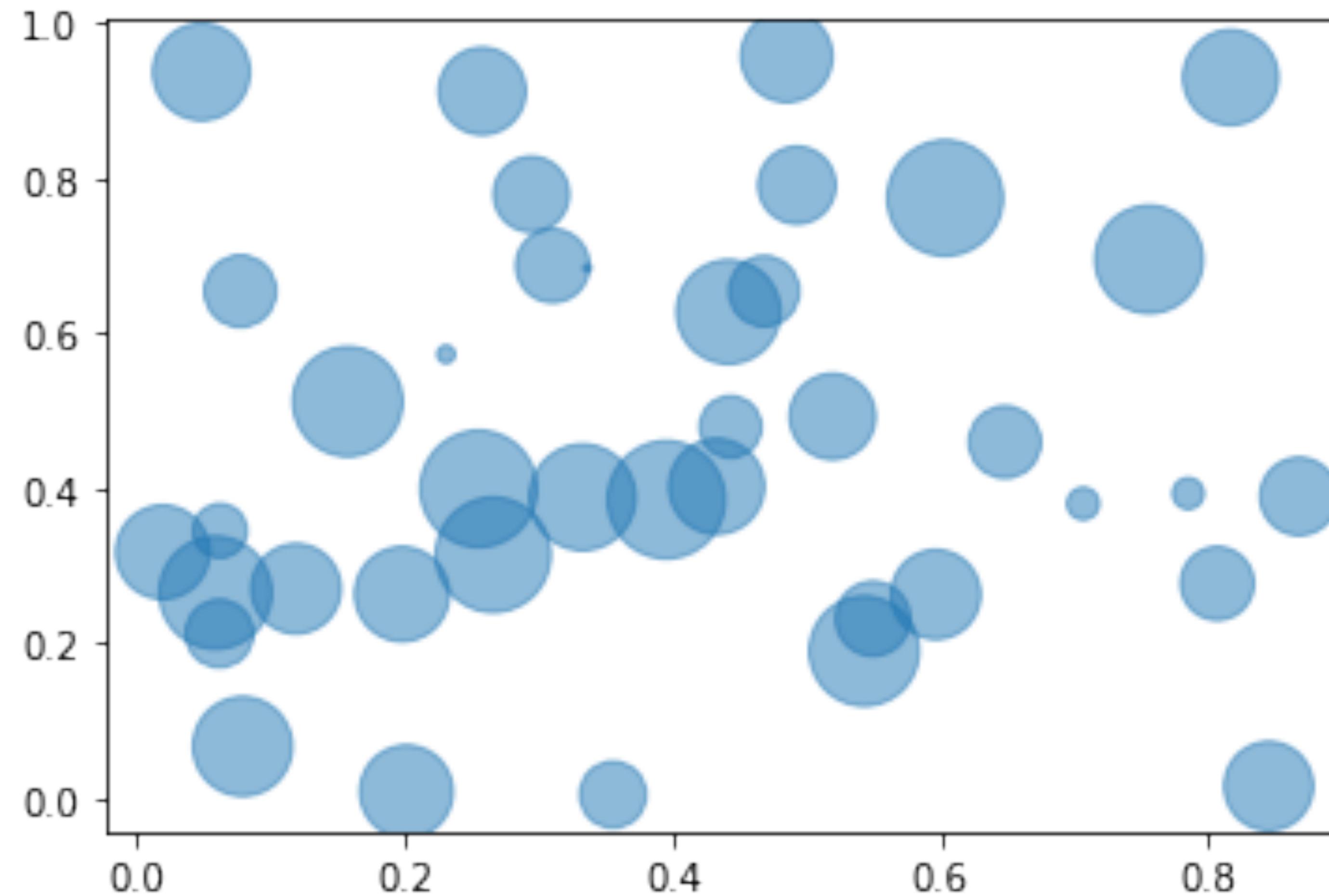
Histogram



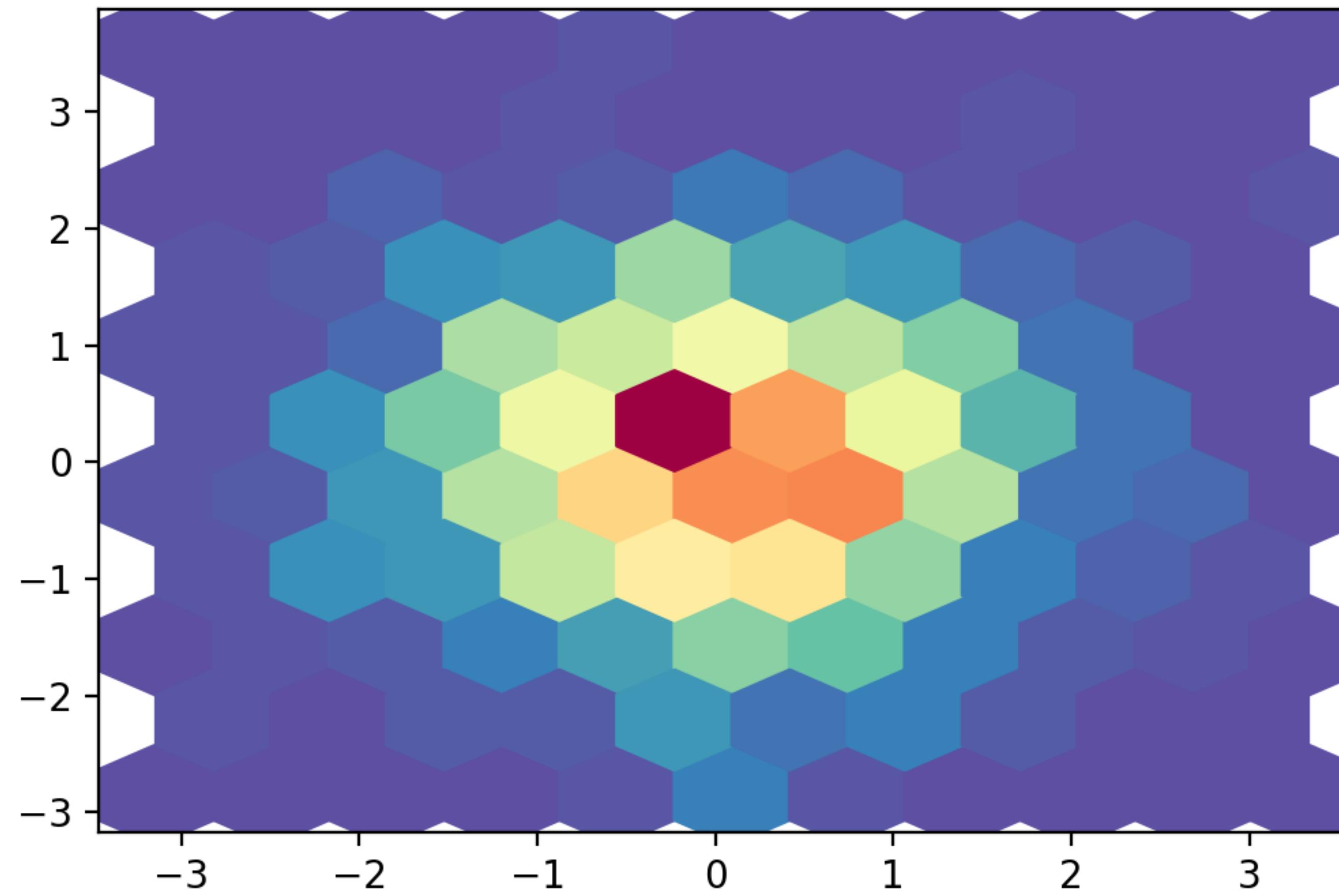
Boxplot



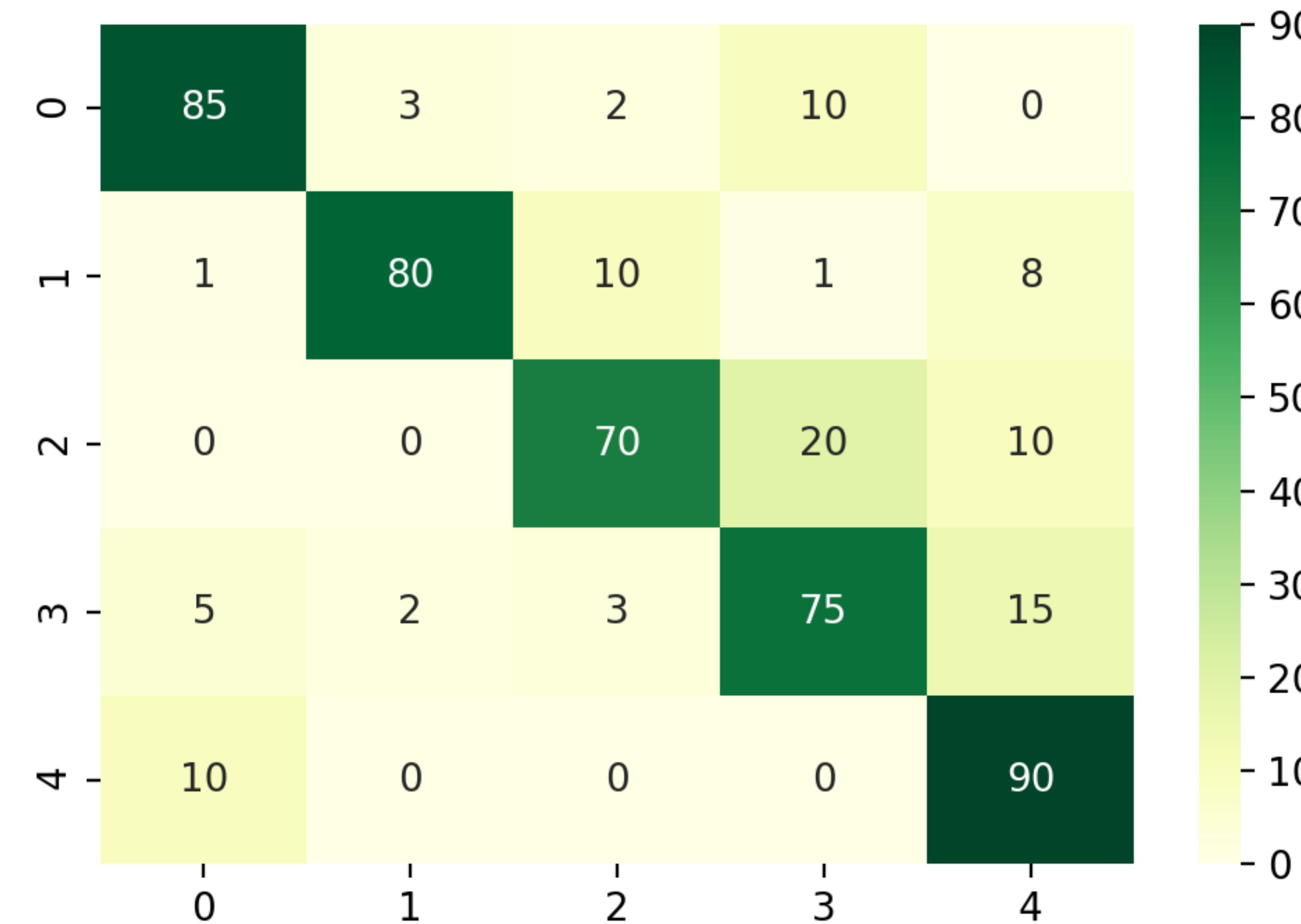
Bubble chart



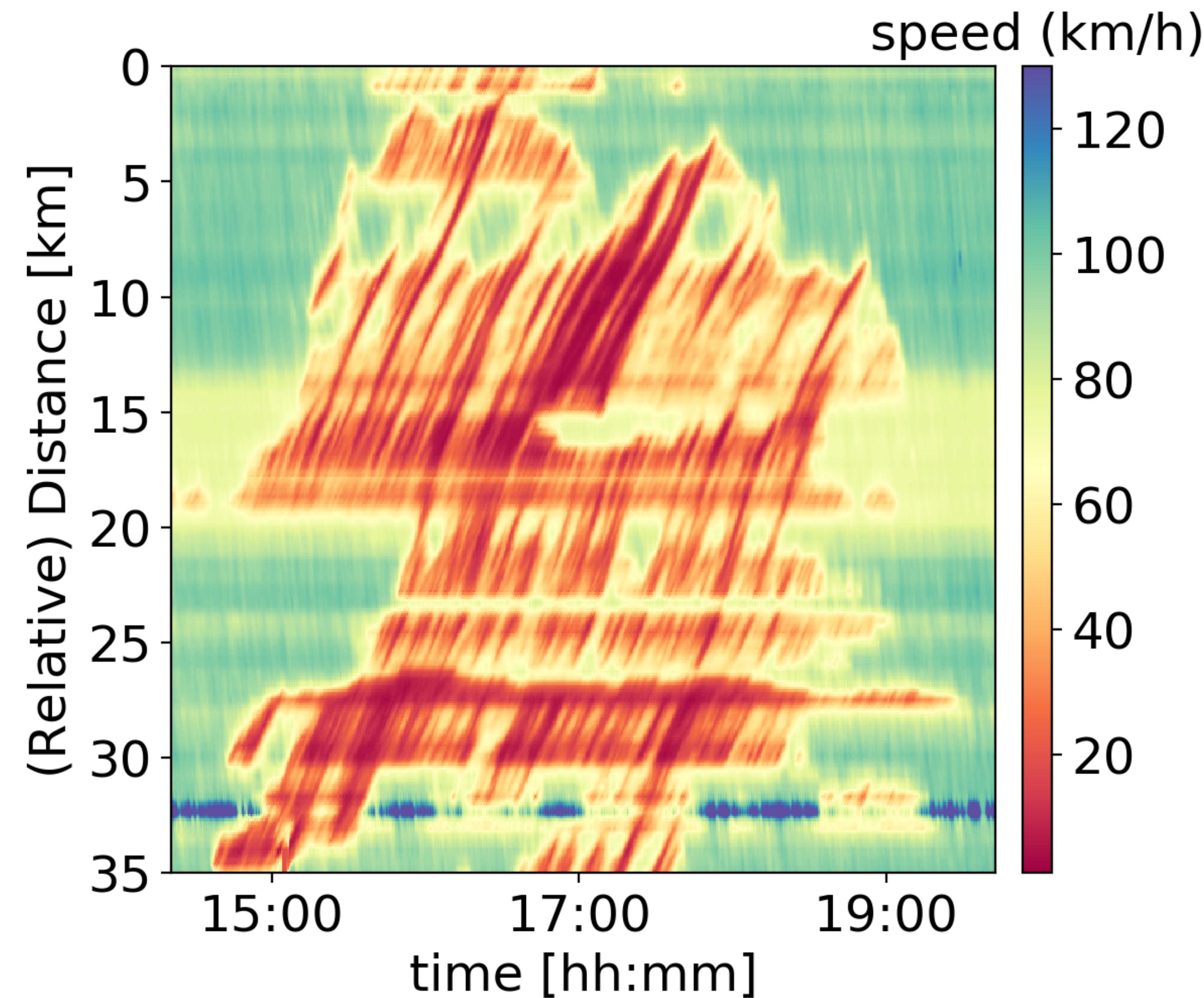
Hexbin



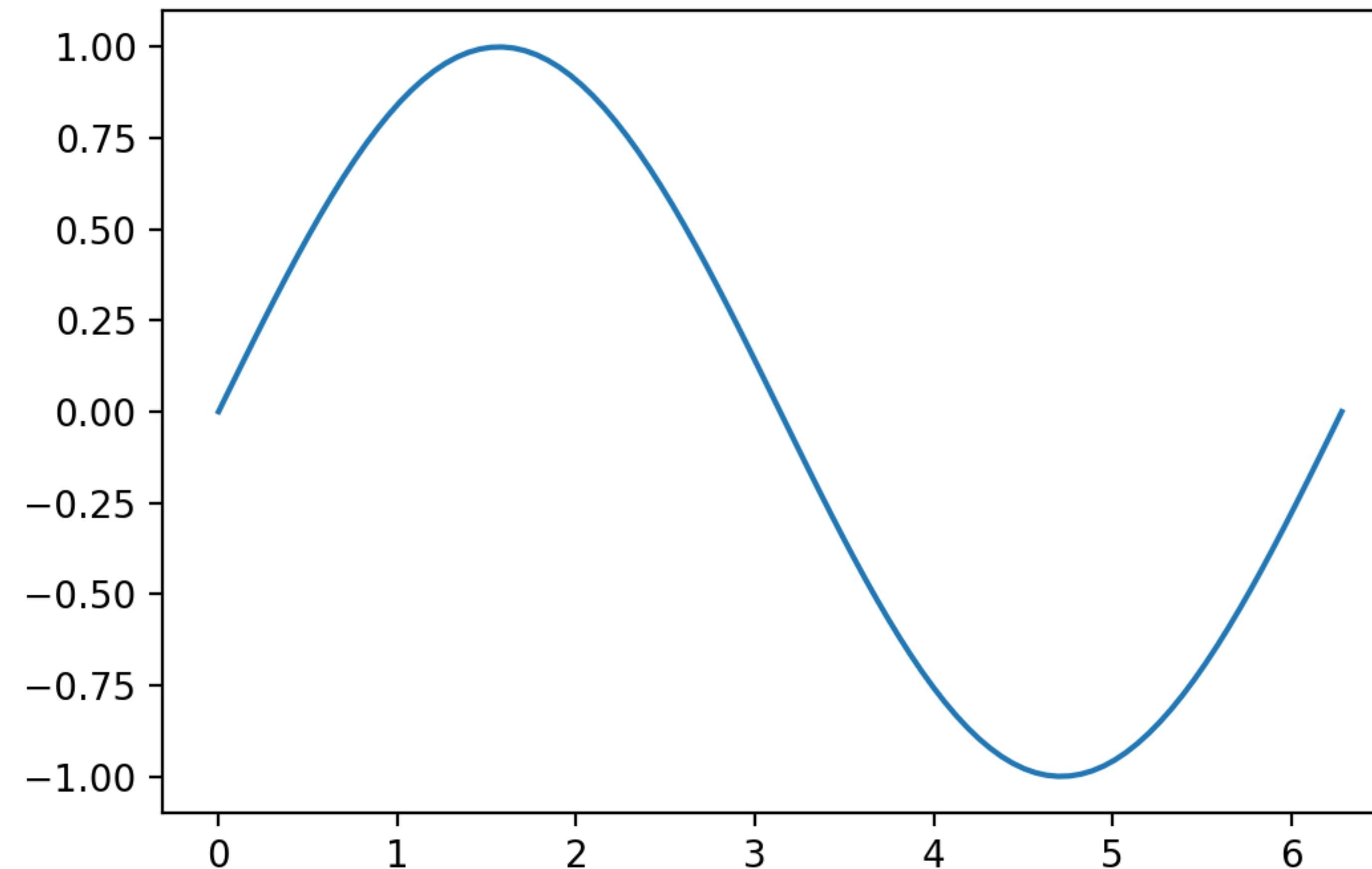
Heatmap



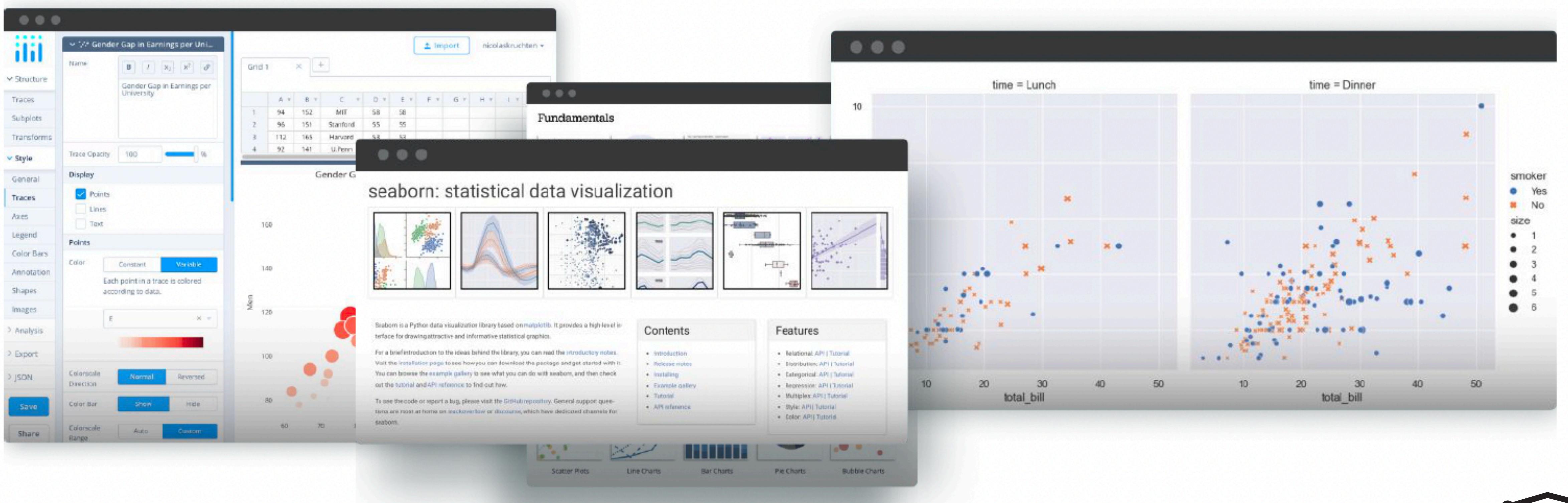
Image



What is wrong with this plot?



Which Python library to use?



Data visualisation with python

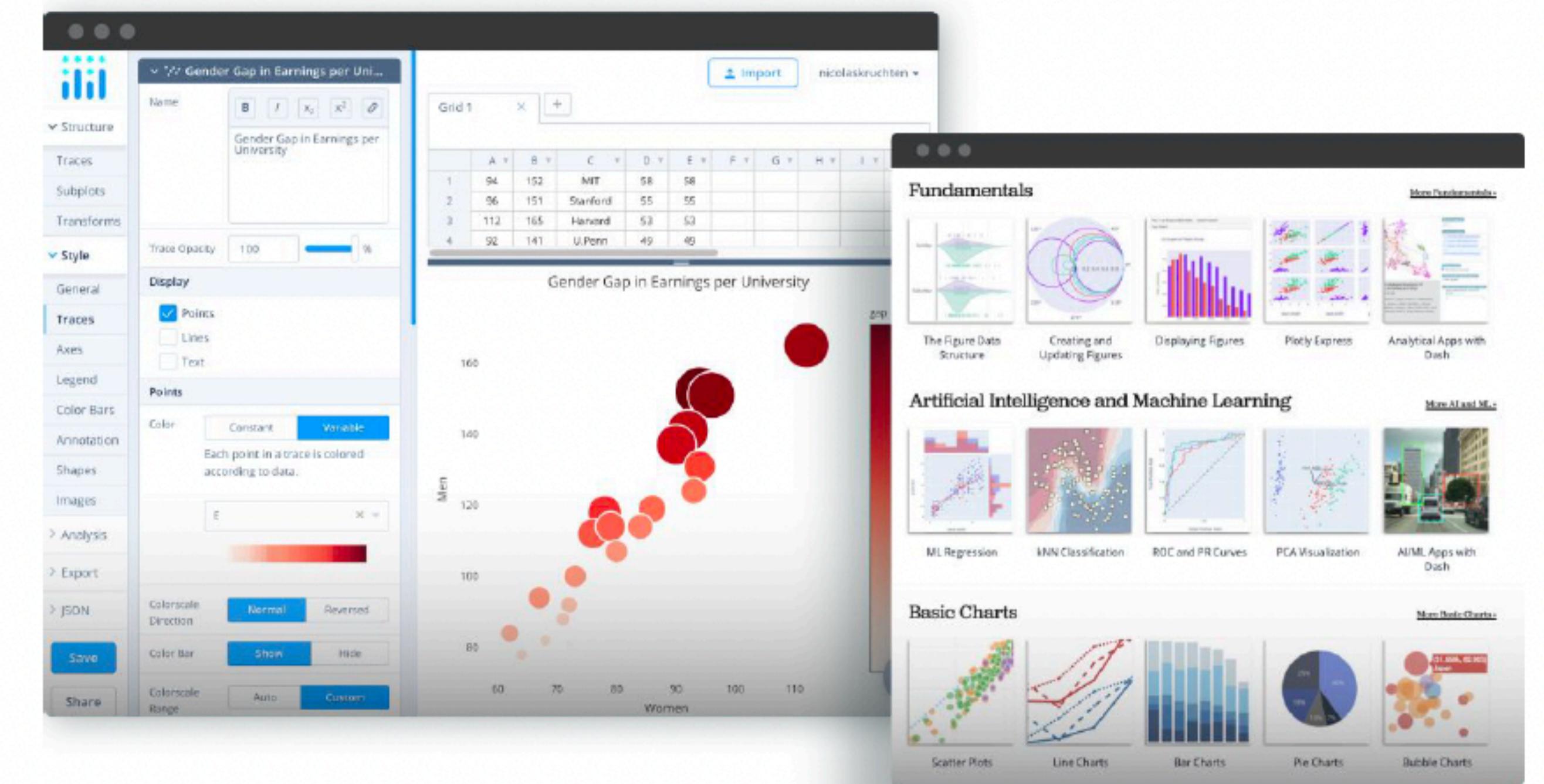
- Matplotlib
- Seaborn
- Plotly
- etc

Matplotlib

Python Libraries

- **Matplotlib**

- It is the first library for Python data visualization. Most of the other libraries took Matplotlib as their base. Some libraries exist only to extend the functionality of Matplotlib and work together with less code.
- Versatile
- Easy to see the property of the information
- Aesthetics need to be improved
- Low-level and less user-friendly

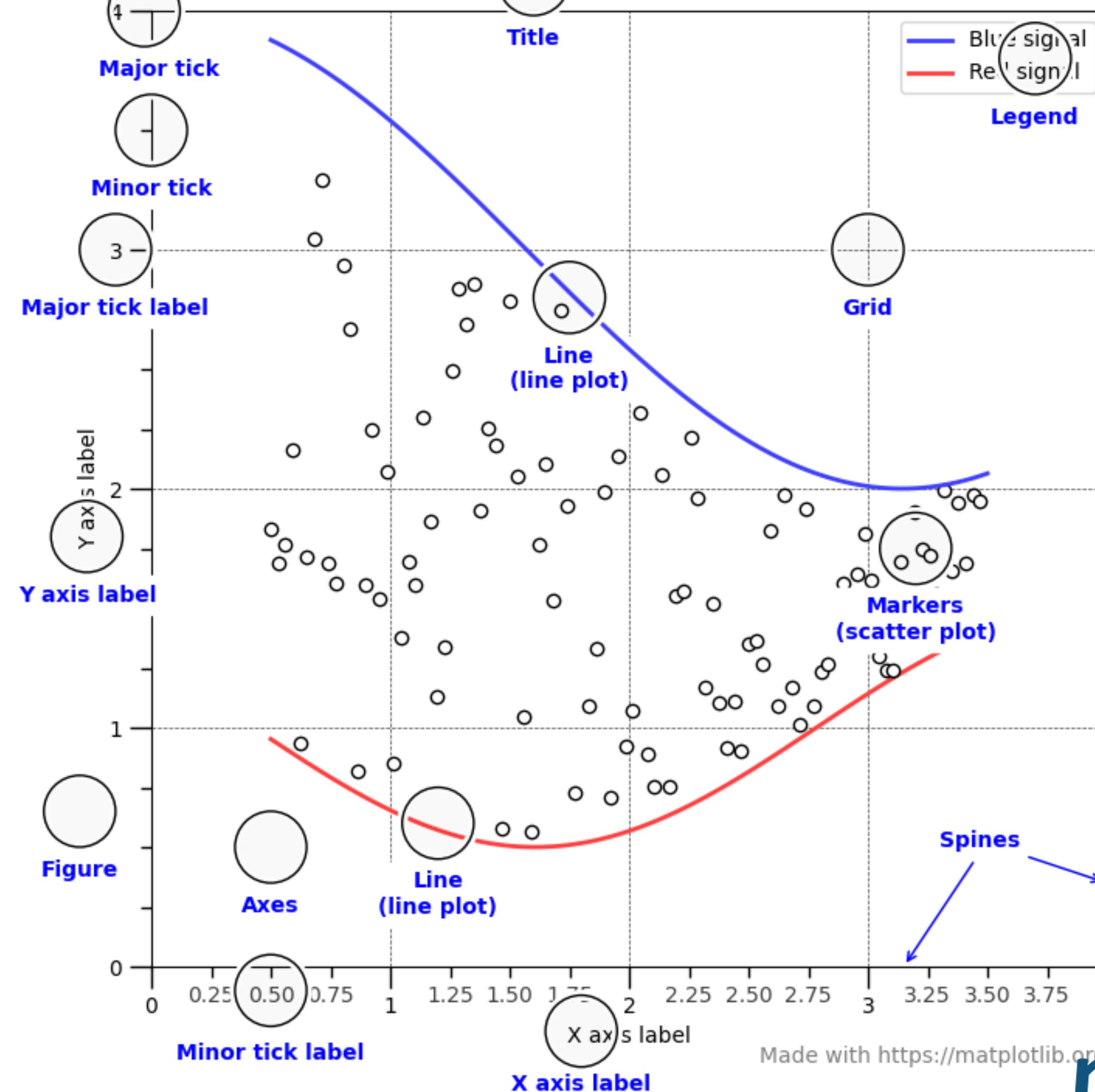


Matplotlib

- The most fundamental visualisation library in Python
- The base for other ‘advanced’, ‘more abstract’ visualisation libraries
- Make plots quickly
- Pros:
 - Full controls to low-level configurations of plots
- Cons:
 - Take time and efforts to generate ‘good’ plots

Anatomy of a figure

Matplotlib Figure anatomy

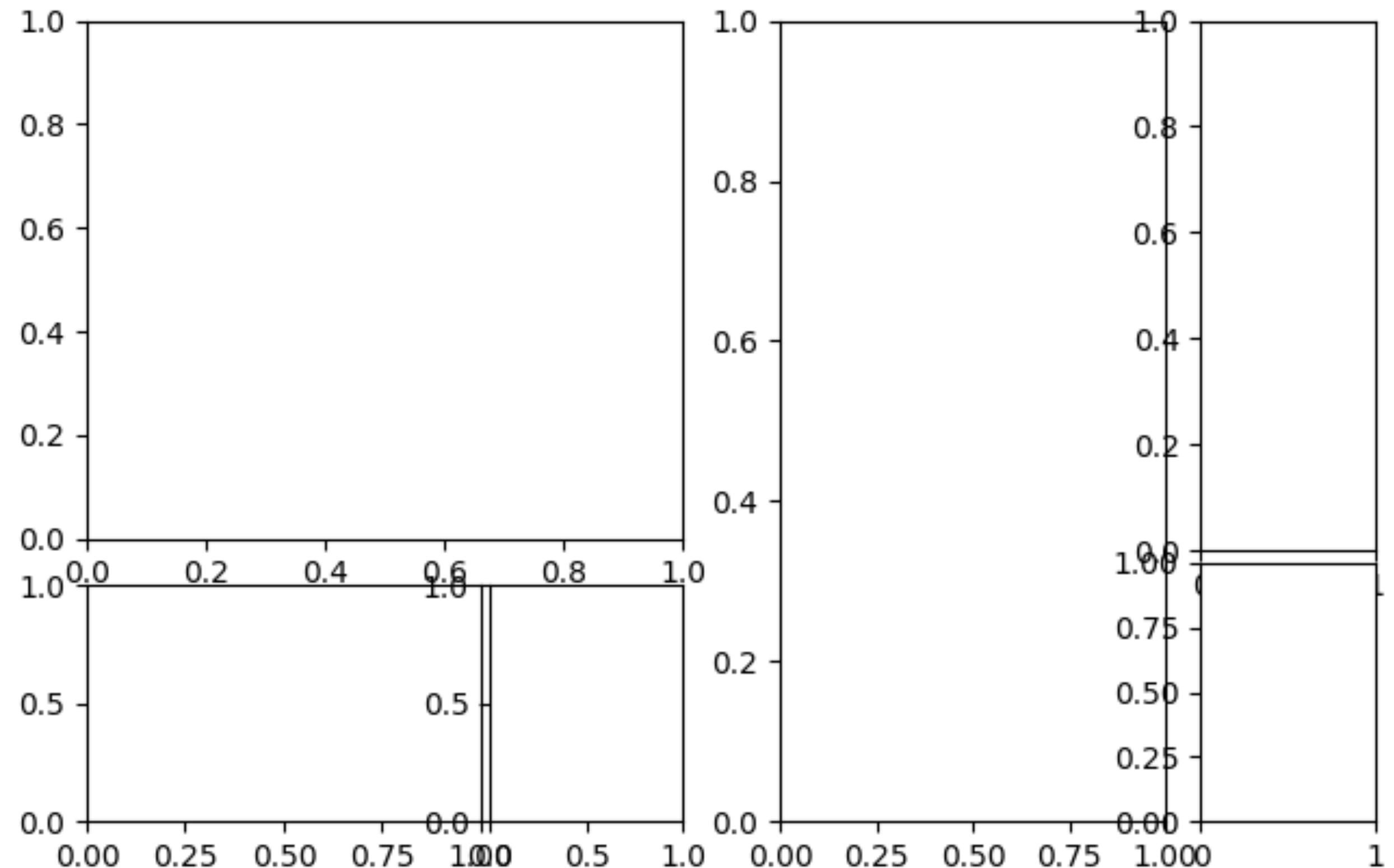


Made with <https://matplotlib.org>

matplotlib

Matplotlib Subplots

- Multiple *sub-plots* in a single figure
- Customisable layouts

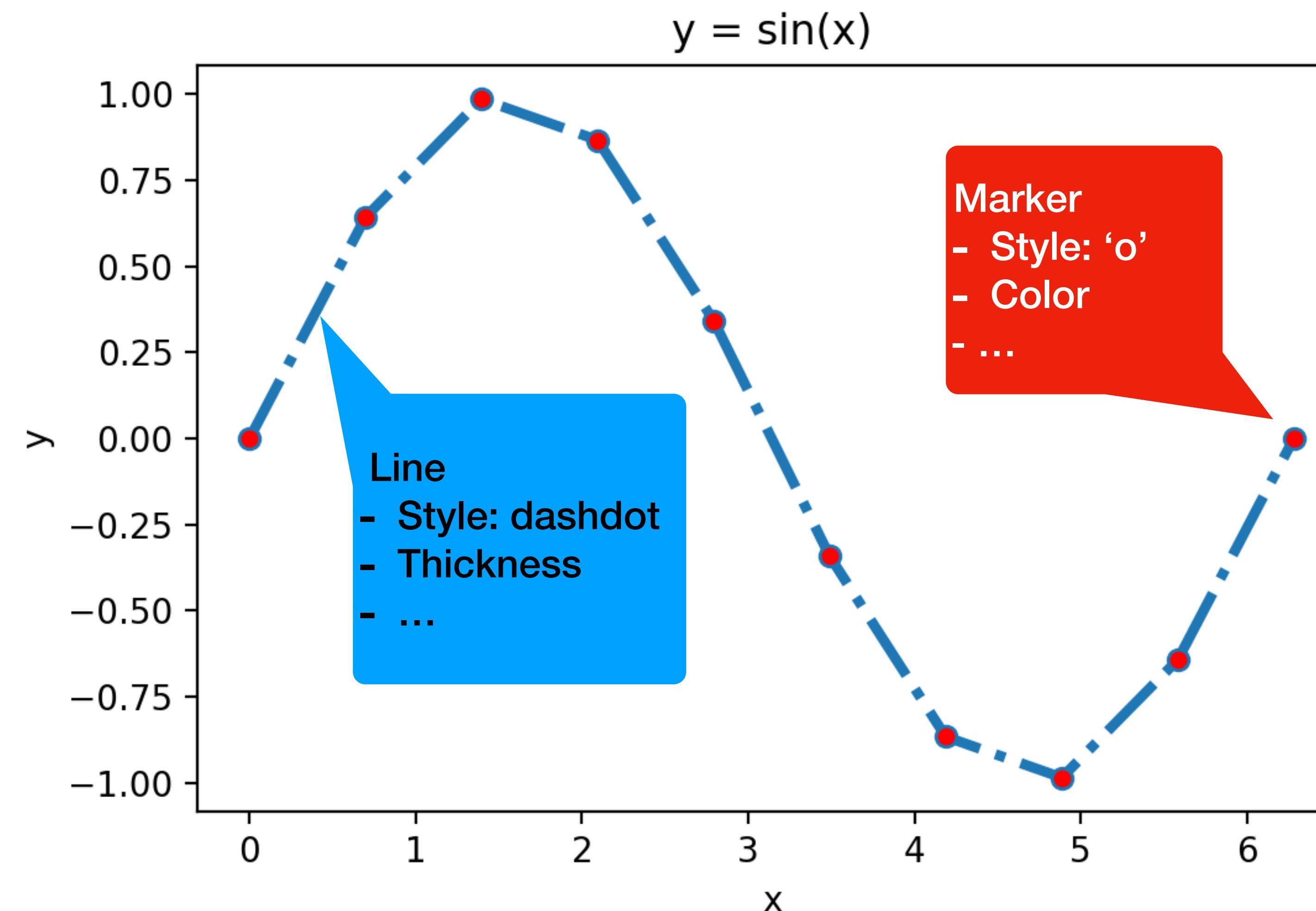


Line plot

Properties

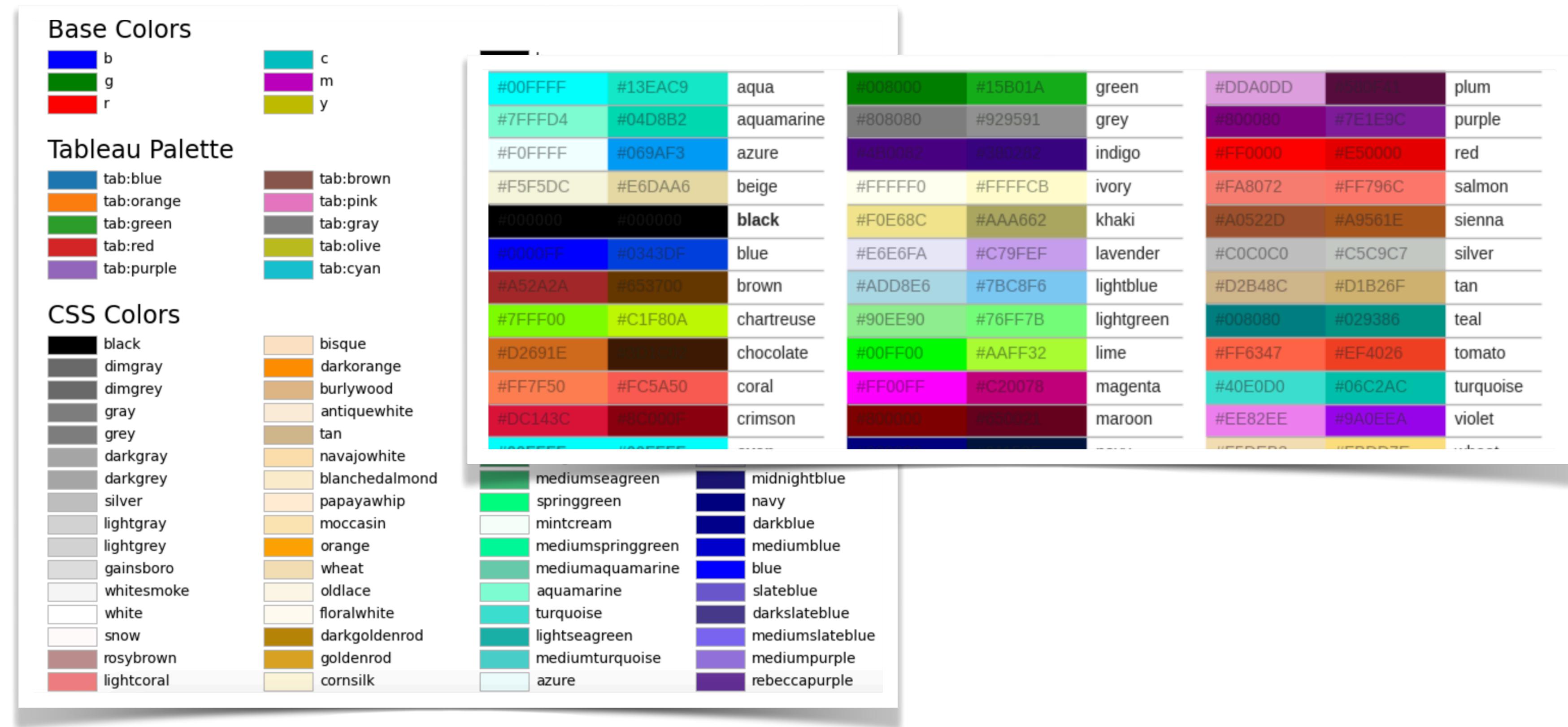
<code>figure</code>	<code>Figure</code>
<code>fillstyle</code>	<code>{'full', 'left', 'right', 'bottom', 'top', 'none'}</code>
<code>gid</code>	<code>str</code>
<code>in_layout</code>	<code>bool</code>
<code>label</code>	<code>object</code>
<code>linestyle</code> or <code>ls</code>	<code>{'-', '--', '-.', ':', '', (offset, on-off-seq), ...}</code>
<code>linewidth</code> or <code>lw</code>	<code>float</code>
<code>marker</code>	<code>marker style string, Path or MarkerStyle</code>
<code>markeredgecolor</code> or <code>mec</code>	<code>color</code>
<code>markeredgewidth</code> or <code>mew</code>	<code>float</code>
<code>markerfacecolor</code> or <code>mfc</code>	<code>color</code>
<code>markerfacecoloralt</code> or <code>mfcalt</code>	<code>color</code>
<code>markersize</code> or <code>ms</code>	<code>float</code>
<code>markevery</code>	<code>None or int or (int, int) or slice or list[int] or float or (float, float) or list[bool]</code>
<code>path_effects</code>	<code>AbstractPathEffect</code>
<code>picker</code>	<code>float or callable[[Artist, Event], tuple[bool, dict]]</code>
<code>pickradius</code>	<code>float</code>
<code>rasterized</code>	<code>bool</code>
<code>sketch_params</code>	<code>(scale: float, length: float, randomness: float)</code>
<code>snap</code>	<code>bool or None</code>
<code>solid_capstyle</code>	<code>CapStyle</code> or <code>{'butt', 'projecting', 'round'}</code>
<code>solid_joinstyle</code>	<code>JoinStyle</code> or <code>{'miter', 'round', 'bevel'}</code>

Line plot Properties



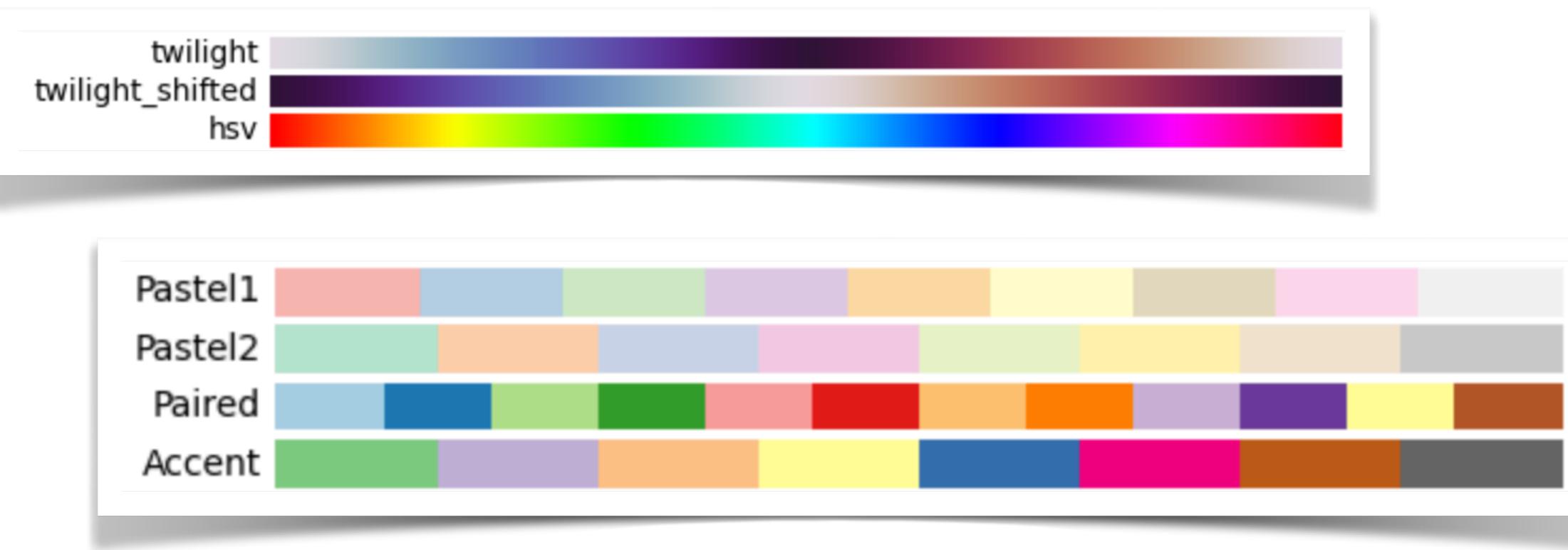
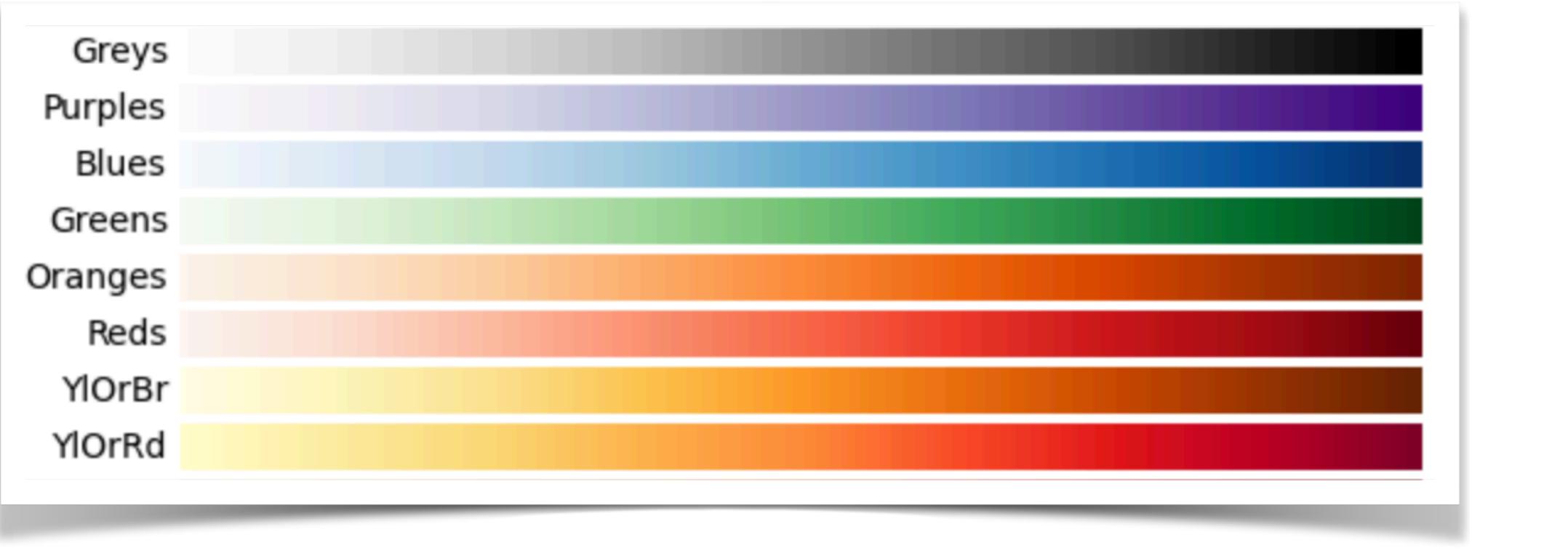
Colors

- Named colors
 - Hex color codes
 - RGB tuples
 - $(0.1, 0.2, 0.5)$



Colormap

- Sequential
- Diverging
- Cyclic
- Qualitative

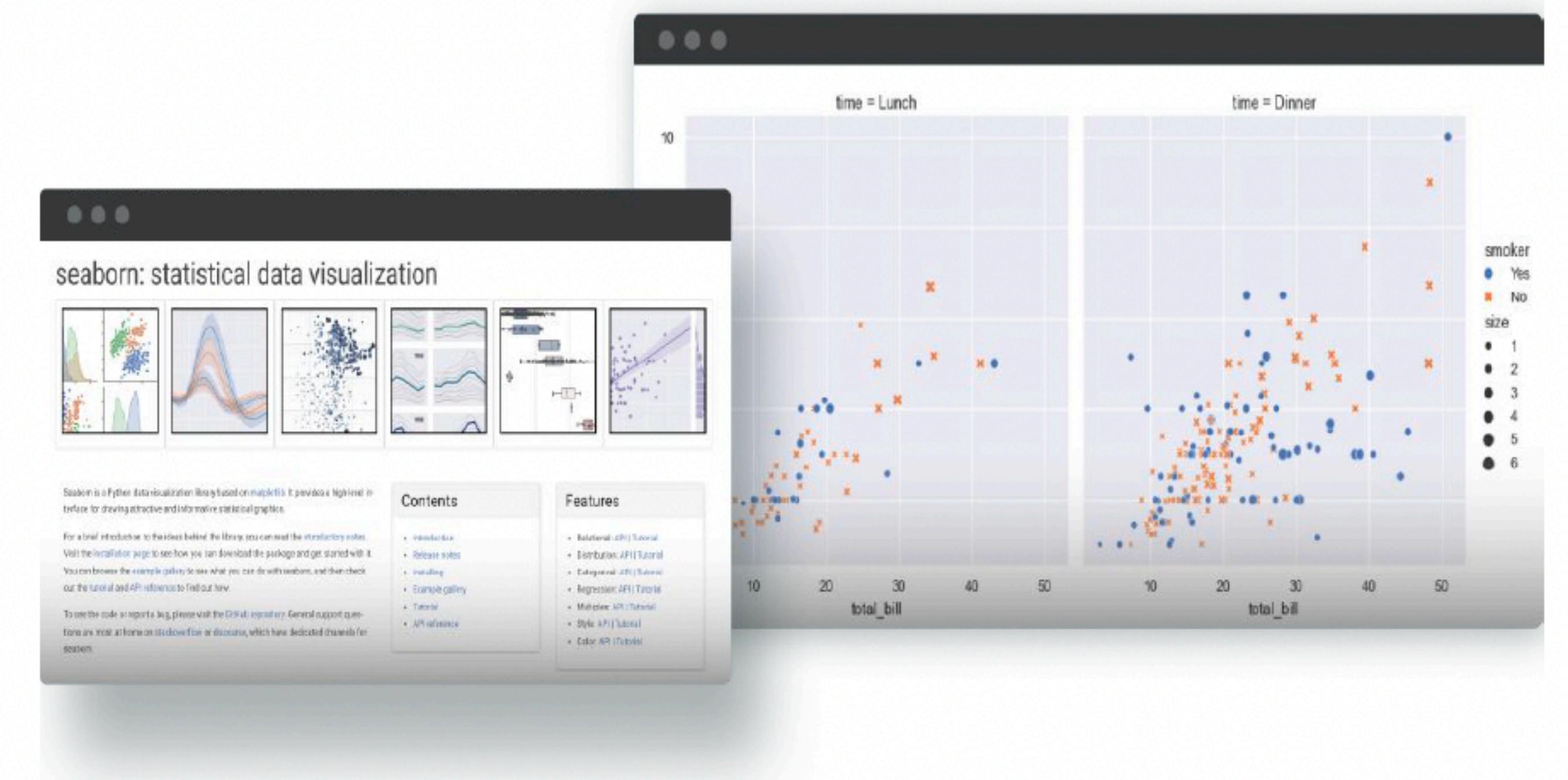


Seaborn

Python Libraries

- **Seaborn**

- Seaborn is a library for visualizing data arrays based on a Matplotlib python plot package.
- Very easy to create individual graphs and heat maps.
- Beautifully presents processed data.
- Less Code
- Less Extensive Collection
- Better aesthetics



Seaborn

- Statistical data visualisation
- Based on matplotlib
- High-level interface

The screenshot shows the official Seaborn website homepage. At the top, there is a navigation bar with the Seaborn logo, version 0.11.2, and links for Gallery, Tutorial, API, Site, and Page. A search bar is also present. Below the navigation, the title "seaborn: statistical data visualization" is displayed. Underneath the title, there is a row of six small images showcasing different types of plots generated by Seaborn, including density plots, scatter plots, and box plots. To the left of these images, a text block provides an overview of what Seaborn is and how to get started. To the right, there are two sidebar boxes: one titled "Contents" listing links to introductory notes, release notes, installing, example gallery, tutorial, and API reference; and another titled "Features" listing categories like Relational, Distribution, Categorical, Regression, Multiples, Style, and Color, each with a link to the API and Tutorial.

Seaborn is a Python data visualization library based on [matplotlib](#). It provides a high-level interface for drawing attractive and informative statistical graphics.

For a brief introduction to the ideas behind the library, you can read the [introductory notes](#) or the [paper](#). Visit the [installation page](#) to see how you can download the package and get started with it. You can browse the [example gallery](#) to see some of the things that you can do with seaborn, and then check out the [tutorial](#) or [API reference](#) to find out how.

To see the code or report a bug, please visit the [GitHub repository](#). General support questions are most at home on [stackoverflow](#) or [discourse](#), which have dedicated channels for seaborn.

Contents

- [Introduction](#)
- [Release notes](#)
- [Installing](#)
- [Example gallery](#)
- [Tutorial](#)
- [API reference](#)

Features

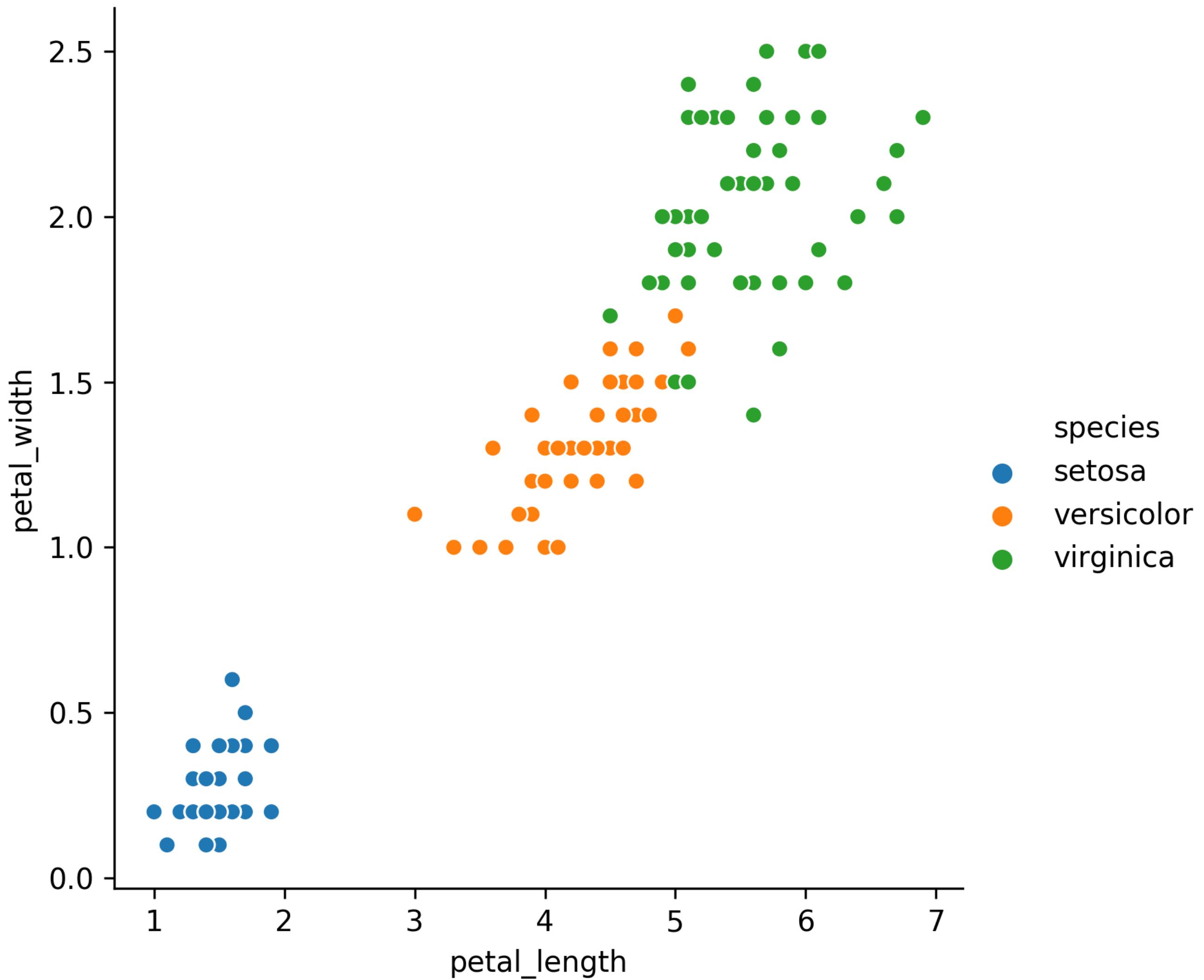
- Relational: [API](#) | [Tutorial](#)
- Distribution: [API](#) | [Tutorial](#)
- Categorical: [API](#) | [Tutorial](#)
- Regression: [API](#) | [Tutorial](#)
- Multiples: [API](#) | [Tutorial](#)
- Style: [API](#) | [Tutorial](#)
- Color: [API](#) | [Tutorial](#)



Seaborn

Statistical relationships

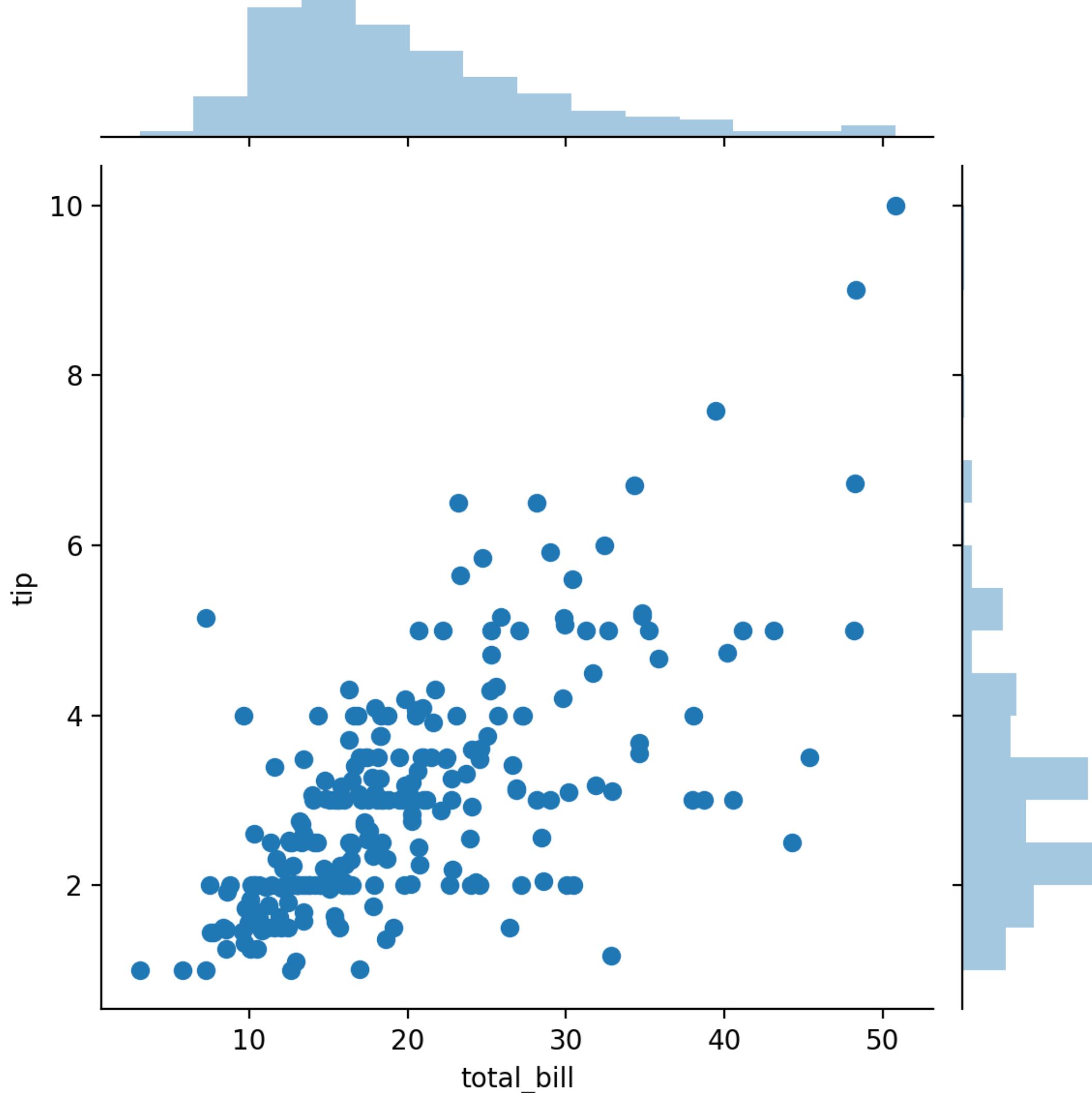
- Scatter plot
- Line plot + confidence-interval
- Joint plot



Seaborn

Statistical relationships

- Scatter plot
- Line plot + confidence-interval
- Joint plot

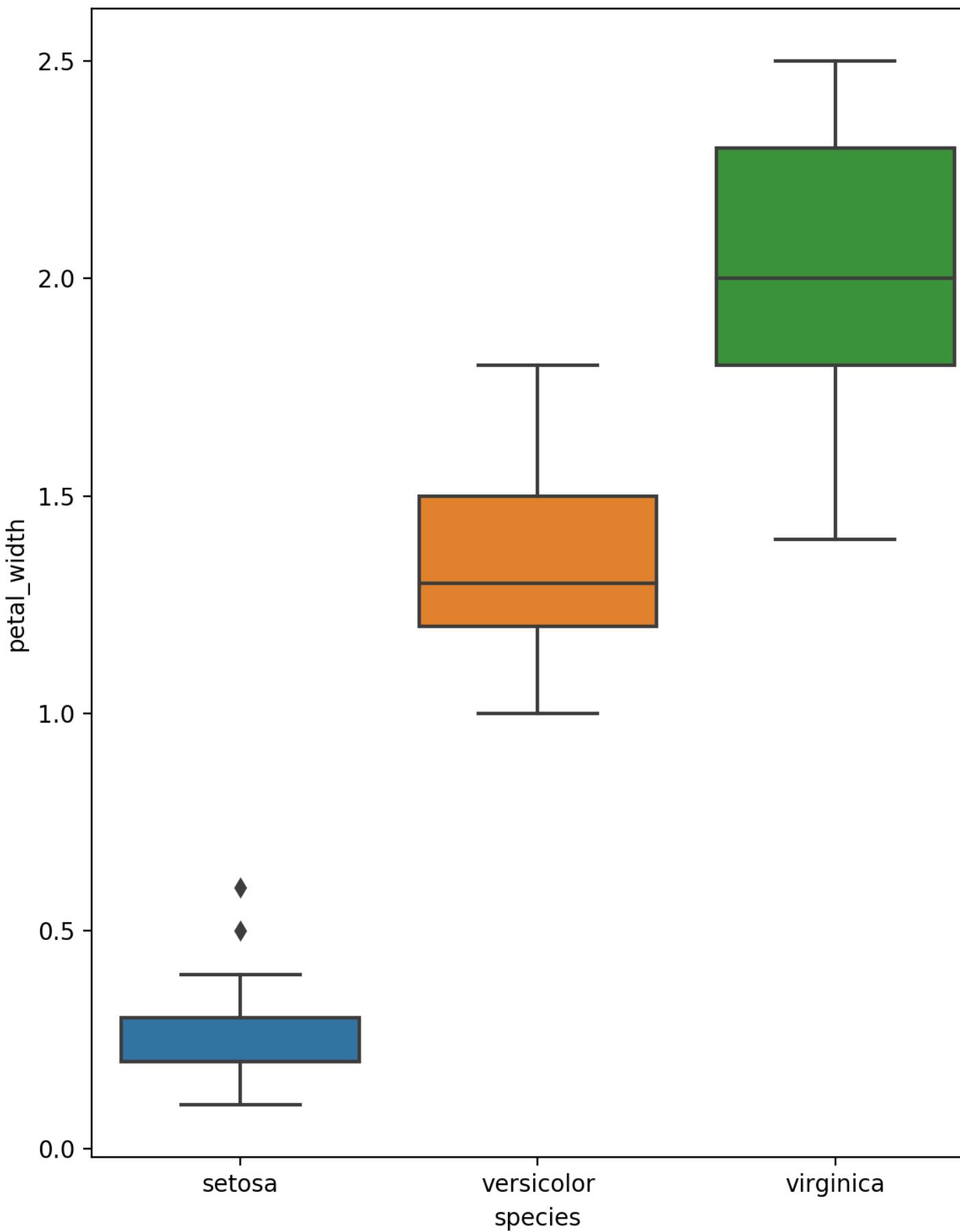


seaborn

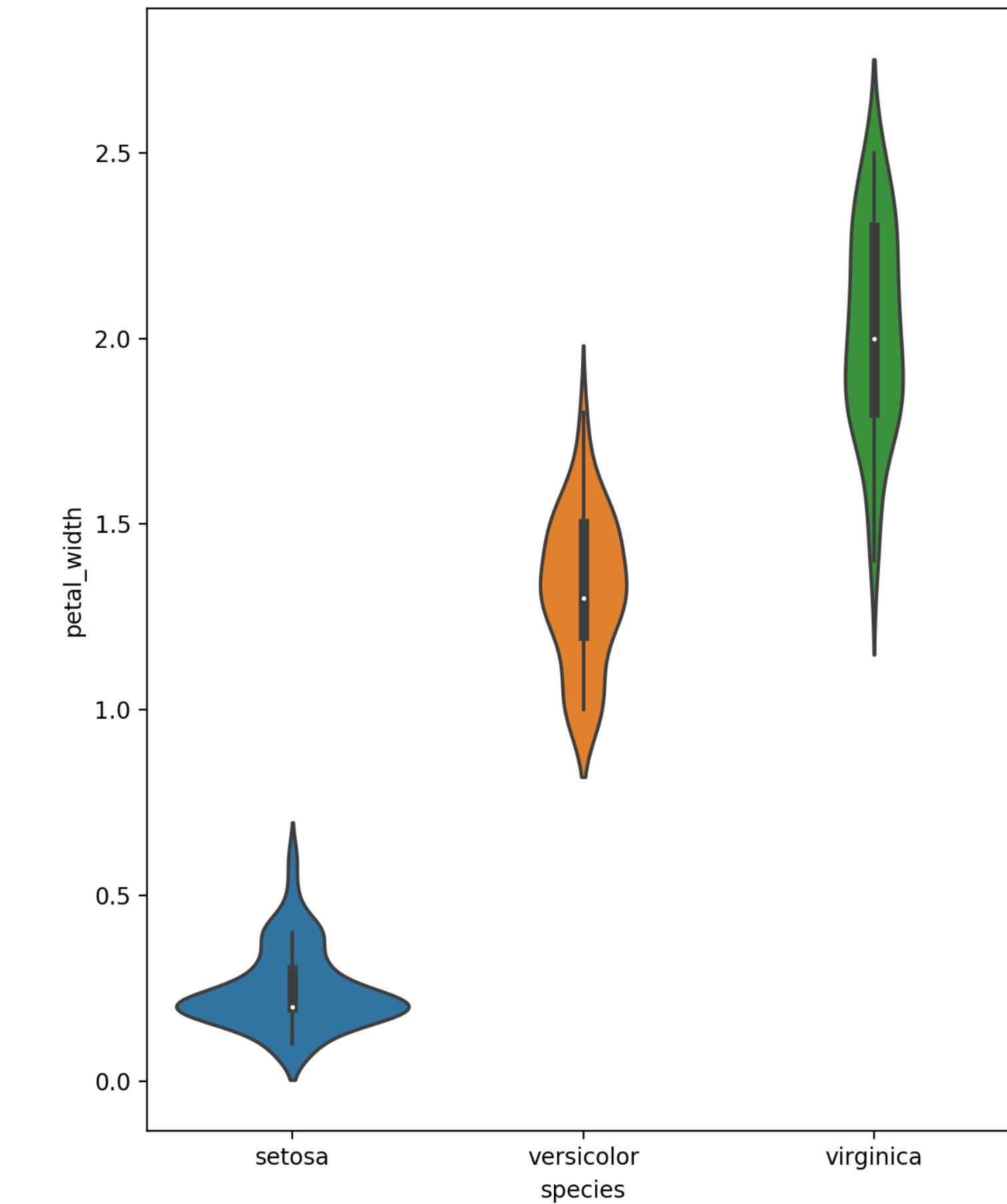
Seaborn

Categorial data

Boxplot



Violinplot

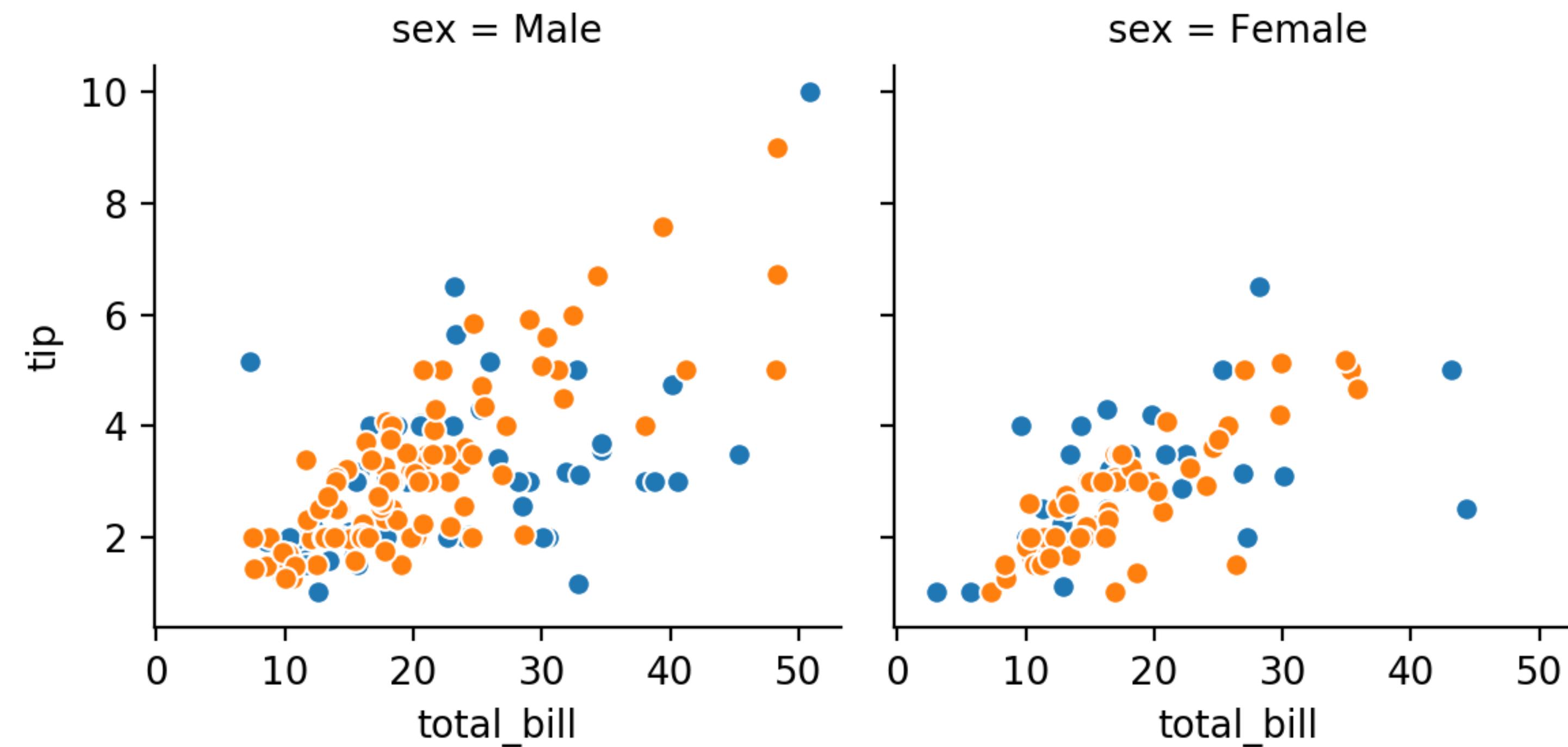


seaborn

Seaborn

Multi-plot grids

- Repeat the same plot on different subsets of dataset



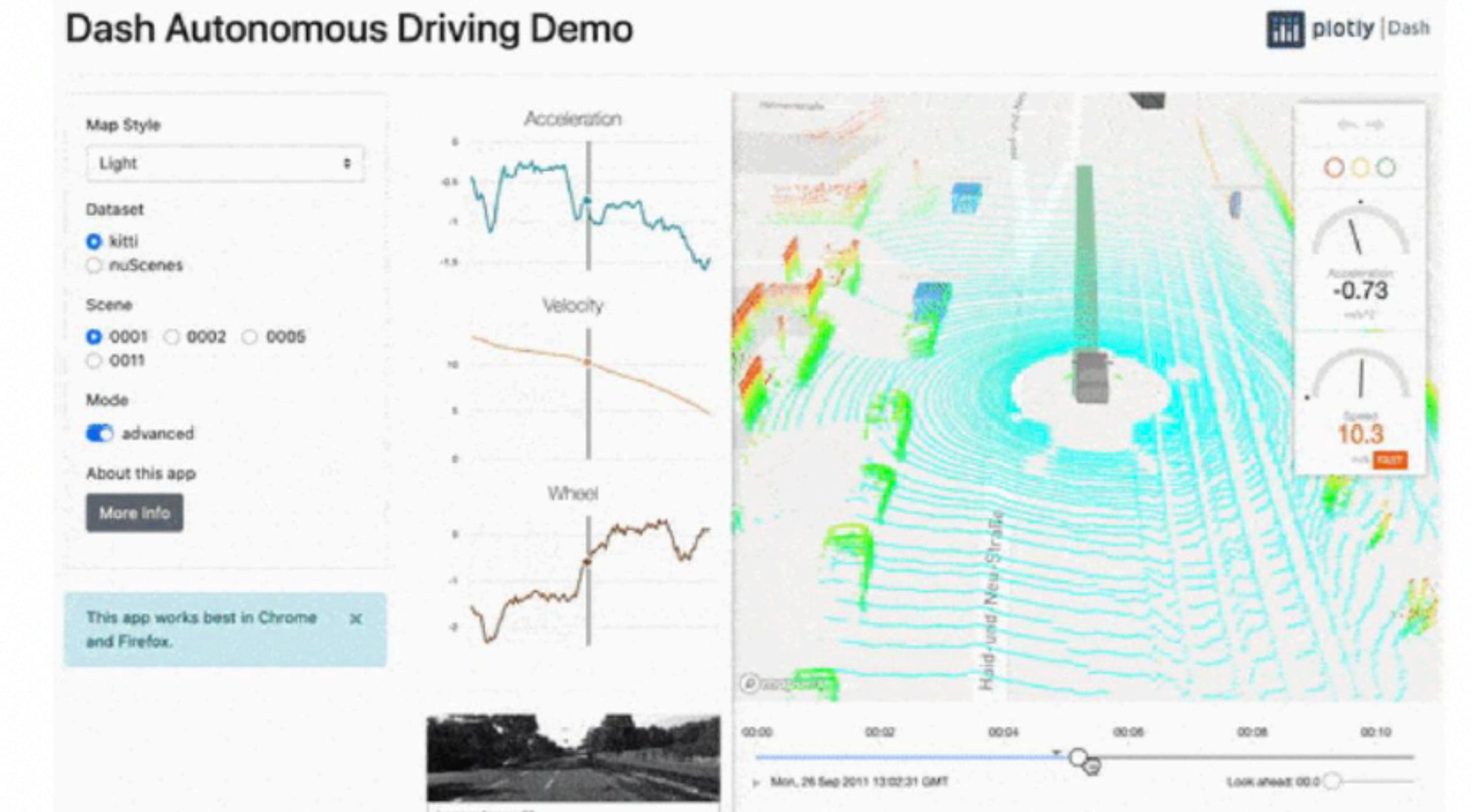
seaborn

Python Libraries

- **Plotly**

- Simple to create interactive plots
- Easily create plots that are usually difficult to develop.
- Plotly is the perfect tool for creating interactive plots with just a few lines of code.

Dash Autonomous Driving Demo



Exam

- 4 types of questions - True/False, multiple choice, multi-select, coding
- 50 points in total
- Digital exam, nothing is allowed other than scrap paper for calculations (if needed)
- 3 hours