Week (12) - "Interactive Data Visualization in Jupyter Notebooks "

Khulood Alkhudaidi

It is very common to use Jupyter notebook for data exploration where the code, output, text, and viz are combined. To get the best out of the data we add interactivity to the visualizations.

This paper is more for developers than scientists. It walks the reader through different techniques, namely Matplotlib, visualization toolkits, and HTML embedding, and determines the pros and cons of each one, along with step by step explanation with code snippets included for each method.

- First, Matplotlib: The author stated that even though Matplotlib can be used directly in Jupyter notebook to create static, animated, and interactive viz, it does not support a wide variety of interactions. To fix that, they suggested using ipywidgets, a library that allows embedding HTML form in Jupyter notebook to allow more interactions.

```
%matplotlib notebook
import matplotlib.pyplot as plt

fig, ax = plt.subplots(); # Creating an empty chart
plt.xlim([0, 10]); plt.ylim([0, 10]) # Setting X and Y axis limits
def onclick(event): # Callback function
    ax.scatter(event.xdata, event.ydata, color='steelblue') # Draw a point on
 ↪top of the user click position.

cid = fig.canvas.mpl_connect('button_press_event', onclick) # Callback setup
```

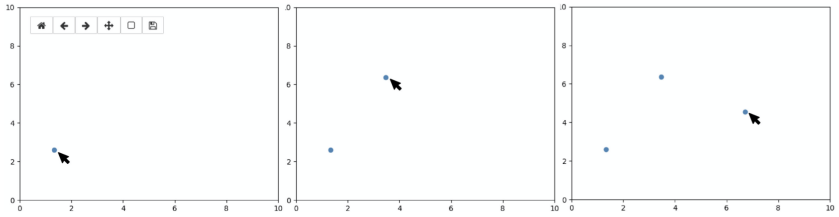

FIGURE 1. Interactive Matplotlib chart, where the user can click on the canvas in order to add a point at that position. The interactive chart also enables pan and zoom operations by default.

- Second, Visualization toolkits: These toolkits are built based on web technologies. The author stated that Plotly and Bokeh facilitate the use of Python and Jupyter notebooks. They allow interactivity and many user inputs through callbacks. Altair, however, has a limitation in communicating the viz with Python, so the result cannot be used as an input to other computations.

```
import altair as alt
from vega_datasets import data

df = data.iris()

alt.Chart(df).mark_circle().encode(
    x='petalLength',
    y='petalWidth',
    color='species',
    tooltip=['sepalLength', 'sepalWidth', 'petalLength', 'petalWidth', 'species']
).interactive()
```

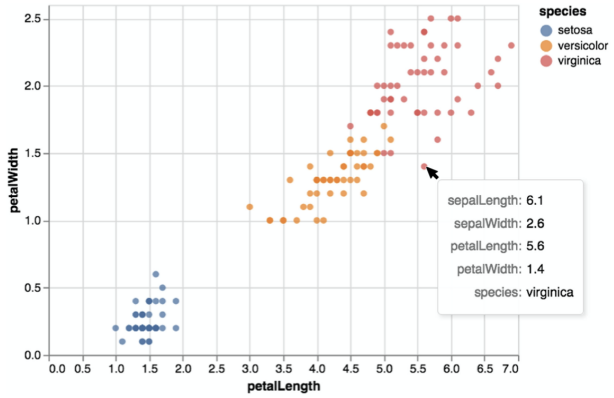

FIGURE 2. Interactive Altair scatter plot of the Iris dataset. The chart displays a tooltip with flower information on mouse hover. The library also enables pan and zoom.

- Third, Embedding HTML: Since the capabilities of the available libraries may not offer the degree of customization needed, the author suggested that developers can make use of HTML, CSS, JS in Jupyter Notebook to have more control over the viz.

```
from IPython.display import display, HTML
import pandas as pd

with open("./BaseballVisualizer/build/baseballvisualizer.js", "r") as f:
    bundled_code = f.read()

play = pd.read_csv("./BaseballVisualizer/play_annotated.csv")
data = {'tracking': play.to_json(orient="records")}

html = """
<html>
<body>
<div id="container"/>
<script type="application/javascript">
    {bundled_code}
    baseballvisualizer.renderBaseballAnnotator("#container", {data});
</script>
</body>
</html>
""".format(bundled_code=bundled_code, data=data)

display(HTML(html))
```

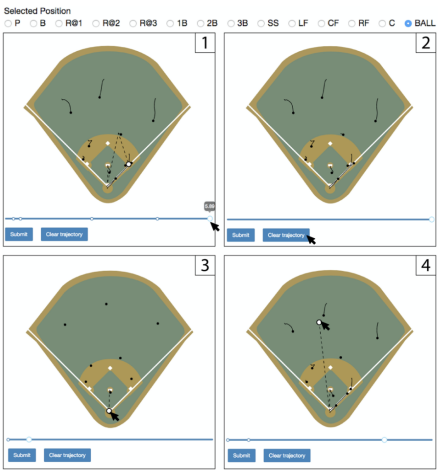

FIGURE 3 Custom JavaScript visualization of baseball plays. The user can (1) animate the play using the slider. Select a position to edit (in the picture, the BALL is selected) and (2) clear the trajectory. Annotate the positions of the ball when it is thrown (3) and hit to the center field (4).

Many other domain-specific libraries for Jupyter notebook interactive visualizations are mentioned in the paper without going deep into the implementation—for example, 3D scientific visualization in Jupyter notebook.

Although the paper is not intensive, it offers valuable techniques that can be useful in many domains.

Sources:

1. J. Piazentin Ono, J. Freire and C. T. Silva, "Interactive Data Visualization in Jupyter Notebooks," in *Computing in Science & Engineering*, vol. 23, no. 2, pp. 99-106, 1 March-April 2021, doi: 10.1109/MCSE.2021.3052619.