Range Slider and Selector in Python

:ors

Now you can implement range sliders and selectors in your Plotly graphs purely with python!

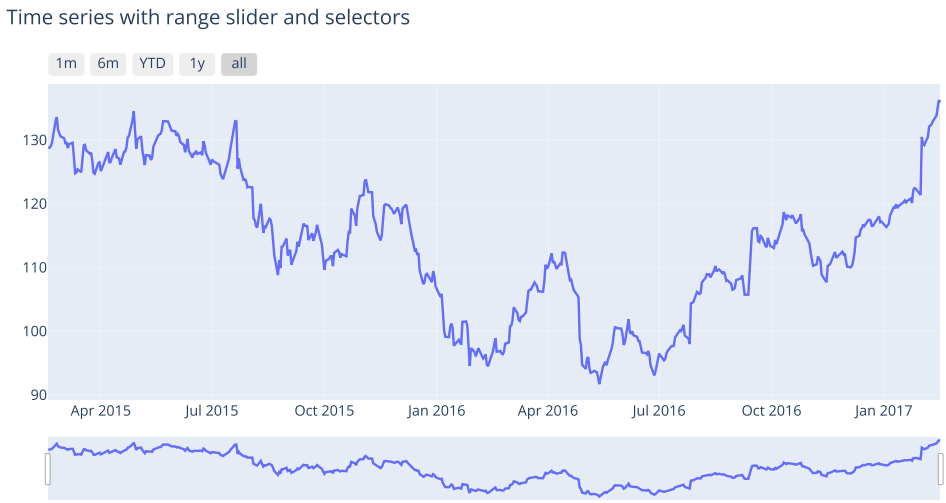
> Plotly Studio: Transform any dataset into an interactive data application in minutes with AI. Sign up for early access now. (https://plotly.com/studio/?utm_medium=graphing_libraries&utm_campaign=studio_early_access&utm_content=sidebar)

Basic Range Slider and Range Selectors

:ors

```
import plotly.graph_objects as go

import pandas as pd

# Load data
df = pd.read_csv(
    "https://raw.githubusercontent.com/plotly/datasets/master/finance-charts-apple.csv")
df.columns = [col.replace("AAPL.", "") for col in df.columns]

# Create figure
fig = go.Figure()

fig.add_trace(
    go.Scatter(x=list(df.Date), y=list(df.High)))

# Set title
fig.update_layout(
    title_text="Time series with range slider and selectors"
)

# Add range slider
fig.update_layout(
    xaxis=dict(
        rangeselector=dict(
            buttons=list([
                dict(count=1,
                     label="1m",
                     step="month",
                     stepmode="backward"),
                dict(count=6,
                     label="6m",
                     step="month",
                     stepmode="backward"),
                dict(count=1,
                     label="YTD",
                     step="year",
                     stepmode="todate"),
                dict(count=1,
                     label="1y",
                     step="year",
                     stepmode="backward"),
                dict(step="all")
            ])
        ),
        rangeslider=dict(
            visible=True
        ),
        type="date"
    )
)

fig.show()
```

:ors



Range Slider with Vertically Stacked Subplots

:ors

```
import plotly.graph_objects as go

# Create figure
fig = go.Figure()

# Add traces
fig.add_trace(go.Scatter(
    x=["2013-01-15", "2013-01-29", "2013-02-26", "2013-04-19", "2013-07-02",
       "2013-08-27",
       "2013-10-22", "2014-01-20", "2014-05-05", "2014-07-01", "2015-02-09",
       "2015-04-13",
       "2015-05-13", "2015-06-08", "2015-08-05", "2016-02-25"],
    y=["8", "3", "2", "10", "5", "5", "6", "8", "3", "3", "7", "5", "10", "10", "9",
       "14"],
    name="var0",
    text=["8", "3", "2", "10", "5", "5", "6", "8", "3", "3", "7", "5", "10", "10", "9",
          "14"],
    yaxis="y",
))

fig.add_trace(go.Scatter(
    x=["2015-04-13", "2015-05-13", "2015-06-08", "2015-08-05", "2016-02-25"],
    y=["53.0", "69.0", "89.0", "41.0", "41.0"],
    name="var1",
    text=["53.0", "69.0", "89.0", "41.0", "41.0"],
    yaxis="y2",
))

fig.add_trace(go.Scatter(
    x=["2013-01-29", "2013-02-26", "2013-04-19", "2013-07-02", "2013-08-27",
       "2013-10-22",
       "2014-01-20", "2014-04-09", "2014-05-05", "2014-07-01", "2014-09-30",
       "2015-02-09",
       "2015-04-13", "2015-06-08", "2016-02-25"],
    y=["9.6", "4.6", "2.7", "8.3", "18", "7.3", "3", "7.5", "1.0", "0.5", "2.8", "9.2",
       "13", "5.8", "6.9"],
    name="var2",
    text=["9.6", "4.6", "2.7", "8.3", "18", "7.3", "3", "7.5", "1.0", "0.5", "2.8",
          "9.2",
          "13", "5.8", "6.9"],
    yaxis="y3",
))

fig.add_trace(go.Scatter(
    x=["2013-01-29", "2013-02-26", "2013-04-19", "2013-07-02", "2013-08-27",
       "2013-10-22",
       "2014-01-20", "2014-04-09", "2014-05-05", "2014-07-01", "2014-09-30",
       "2015-02-09",
       "2015-04-13", "2015-06-08", "2016-02-25"],
    y=["6.9", "7.5", "7.3", "7.3", "6.9", "7.1", "8", "7.8", "7.4", "7.9", "7.9", "7.6",
       "7.2", "7.2", "8.0"],
    name="var3",
    text=["6.9", "7.5", "7.3", "7.3", "6.9", "7.1", "8", "7.8", "7.4", "7.9", "7.9",
          "7.6",
          "7.2", "7.2", "8.0"],
    yaxis="y4",
))

fig.add_trace(go.Scatter(
    x=["2013-02-26", "2013-07-02", "2013-09-26", "2013-10-22", "2013-12-04",
       "2014-01-02",
       "2014-01-20", "2014-05-05", "2014-07-01", "2015-02-09", "2015-05-05"],
    y=["290", "1078", "263", "407", "660", "740", "33", "374", "95", "734", "3000"],
    name="var4",
    text=["290", "1078", "263", "407", "660", "740", "33", "374", "95", "734", "3000"],
    yaxis="y5",
))

# style all the traces
fig.update_traces(
    hoverinfo="name+x+text",
    linecolor=0.5},
    marker={"size": 8},
    mode="lines+markers",
    showlegend=False
)
```

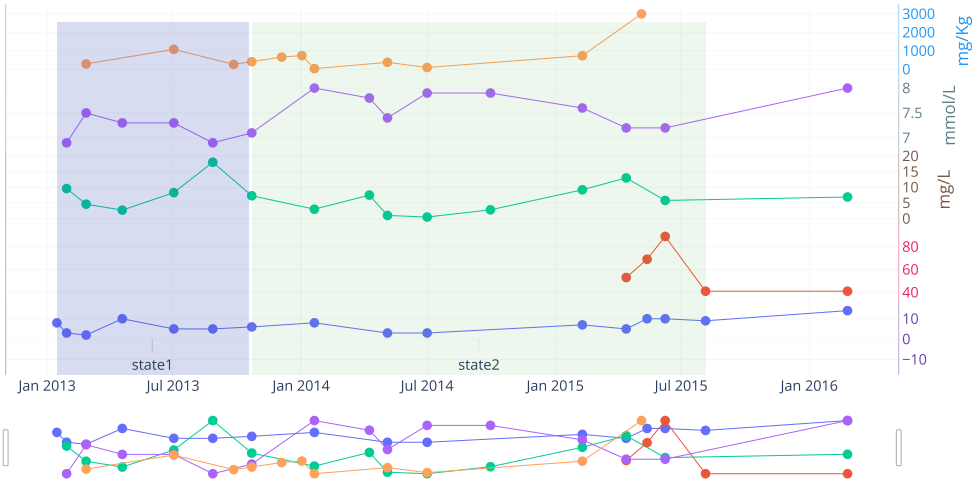
:ors

```
fig.update_layout(
    annotations=[
        dict(
            x="2013-06-01",
            y=0,
            arrowcolor="rgba(63, 81, 181, 0.2)",
            arrowsize=0.3,
            ax=0,
            ay=30,
            text="state1",
            xref="x",
            yanchor="bottom",
            yref="y"
        ),
        dict(
            x="2014-09-13",
            y=0,
            arrowcolor="rgba(76, 175, 80, 0.1)",
            arrowsize=0.3,
            ax=0,
            ay=30,
            text="state2",
            xref="x",
            yanchor="bottom",
            yref="y"
        )
    ],
)

# Add shapes
fig.update_layout(
    shapes=[
        dict(
            fillcolor="rgba(63, 81, 181, 0.2)",
            line={"width": 0},
            type="rect",
            x0="2013-01-15",
            x1="2013-10-17",
            xref="x",
            y0=0,
            y1=0.95,
            yref="paper"
        ),
        dict(
            fillcolor="rgba(76, 175, 80, 0.1)",
            line={"width": 0},
            type="rect",
            x0="2013-10-22",
            x1="2015-08-05",
            xref="x",
            y0=0,
            y1=0.95,
            yref="paper"
        )
    ]
)

# Update axes
fig.update_layout(
    xaxis=dict(
        autorange=True,
        range=["2012-10-31 18:36:37.3129", "2016-05-10 05:23:22.6871"],
        rangeslider=dict(
            autorange=True,
            range=["2012-10-31 18:36:37.3129", "2016-05-10 05:23:22.6871"]
        ),
        type="date"
    ),
    yaxis=dict(
        anchor="x",
        autorange=True,
        0.2],
        #673ab7",
        ,
        0858369099, 28.4406294707],
        ue,
        ",
        color": "#673ab7"},
        uto",
```

```
        title=dict(
            font=dict(
                color="#673ab7"
            )
        ),
        type="linear",
        zeroline=False
    ),
    yaxis2=dict(
        anchor="x",
        autorange=True,
        domain=[0.2, 0.4],
        linecolor="#E91E63",
        mirror=True,
        range=[29.3787777032, 100.621222297],
        showline=True,
        side="right",
        tickfont={"color": "#E91E63"},
        tickmode="auto",
        ticks="",
        title=dict(
            font=dict(
                color="#E91E63"
            )
        ),
        type="linear",
        zeroline=False
    ),
    yaxis3=dict(
        anchor="x",
        autorange=True,
        domain=[0.4, 0.6],
        linecolor="#795548",
        mirror=True,
        range=[-3.73690396239, 22.2369039624],
        showline=True,
        side="right",
        tickfont={"color": "#795548"},
        tickmode="auto",
        ticks="",
        title=dict(
            text="mg/L",
            font=dict(
                color="#795548"
            )
        ),
        type="linear",
        zeroline=False
    ),
    yaxis4=dict(
        anchor="x",
        autorange=True,
        domain=[0.6, 0.8],
        linecolor="#607d8b",
        mirror=True,
        range=[6.63368032236, 8.26631967764],
        showline=True,
        side="right",
        tickfont={"color": "#607d8b"},
        tickmode="auto",
        ticks="",
        title=dict(
            text="mmol/L",
            font=dict(
                color="#607d8b"
            )
        ),
        type="linear",
        zeroline=False
    ),
    yaxis5=dict(
                                    rue,
                                    , 1],
                                    #2196F3",
                                    ,
                                    .336803224, 3718.33680322],
                                    ue,
                                    ",
                                    color": "#2196F3"},
```

:ors

```
        tickmode="auto",
        ticks="",
        title=dict(
            text="mg/Kg",
            font=dict(
                color="#2196F3"
            )
        ),
        type="linear",
        zeroline=False
    )
)

# Update layout
fig.update_layout(
    dragmode="zoom",
    hovermode="x",
    legend=dict(traceorder="reversed"),
    height=600,
    template="plotly_white",
    margin=dict(
        t=100,
        b=100
    ),
)

fig.show()
```



Reference

See https://plotly.com/python/reference/layout/xaxis/#layout-xaxis-rangeselector (https://plotly.com/python/reference/layout/xaxis/#layout-xaxis-rangeselector) and https://plotly.com/python/reference/layout/xaxis/#layout-xaxis-rangeslider (https://plotly.com/python/reference/layout/xaxis/#layout-xaxis-rangeslider) for more information and attribute options!

What About Dash?

Dash (https://dash.plot.ly/) is an open-source framework for building analytical applications, with no Javascript required, and it is tightly integrated with the Plotly graphing library.

Learn about how to install Dash at https://dash.plot.ly/installation (https://dash.plot.ly/installation).

Everywhere in this page that you see fig.show(), you can display the same figure in a Dash application by passing it to the figure argument of the Graph component (https://dash.plot.ly/dash-core-components/graph) from the built-in dash_core_components package like this:

```
import plotly.graph_objects as go # or plotly.express as px
fig = go.Figure() # or any Plotly Express function e.g. px.bar(...)
# fig.add_trace( ... )
# fig.update_layout( ... )

from dash import Dash, dcc, html

app = Dash()
app.layout = html.Div([
    dcc.Graph(figure=fig)
])

app.run(debug=True, use_reloader=False)  # Turn off reloader if inside Jupyter
```