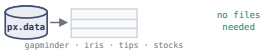
Plotly

A quick reference for Plotly's interactive charts you can zoom, pan, and hover, from express one-liners to composed figures and dashboards.

plotly.express one-liners

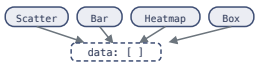
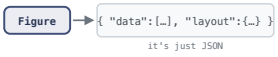
One function call, tidy DataFrame in, interactive Figure out.

PURPOSE	COMMAND	RESULT
Interactive scatter from a DataFrame.	`px.scatter(df, x="sepal_width", y="sepal_length", color="species")`	
Line / time series.	`px.line(df, x="date", y="GOOG")`	
Bar chart (aggregated).	`px.bar(df, x="day", y="total_bill", color="sex")`	
Distribution.	`px.histogram(df, x="total_bill", nbins=30)`	
Categorical spread.	`px.box(df, x="day", y="total_bill")`	
Built-in demo data.	`df = px.data.gapminder()`	

px.scatter / line / bar / histogram / box take a tidy DataFrame plus column names and return one interactive Figure

The Figure Object

Every figure is one object: a list of traces plus a layout.

PURPOSE	COMMAND	RESULT
Build a Figure by hand.	`fig = go.Figure(data=go.Scatter( x=[1,2,3], y=[4,5,6], mode="markers"))`	
A Figure has data + layout.	`fig.data · fig.layout`	
Every chart type is a trace.	`go.Scatter · go.Bar · go.Heatmap · go.Box`	
Inspect as a plain dict.	`fig.to_dict() · print(fig)`	
px returns the same object.	`type(px.scatter(df, x="a", y="b")) # go.Figure`	
Render it.	`fig.show()`	

px and go both return one go.Figure: a data list of traces plus a layout dict; it's just JSON you can inspect and edit

add_trace + update_layout

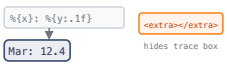
Layer more traces, then style the whole figure with update_ methods.

PURPOSE	COMMAND	RESULT
Start an empty figure.	`fig = go.Figure()`	
Add a trace (layer a series).	`fig.add_trace(go.Scatter(x=x, y=y, name="obs"))`	
Title and axis labels.	`fig.update_layout(title="Sales", xaxis_title="Month", yaxis_title="USD")`	
Restyle matching traces.	`fig.update_traces(marker_color="#0d6efd", selector=dict(type="bar"))`	
Tweak one axis.	`fig.update_xaxes(range=[0, 100], showgrid=False)`	
Drop a reference line.	`fig.add_hline(y=50, line_dash="dash", annotation_text="target")`	

add_trace layers series into data; the update_ family (update_layout / update_traces / update_xaxes) and add_hline restyle the figure

Hover, Zoom, Pan

The payoff over static plots: tooltips, zoom-box, pan, and the mode-bar.

PURPOSE	COMMAND	RESULT
Show extra fields on hover.	`px.scatter(df, x="sepal_width", y="sepal_length", hover_data=["petal_length", "petal_width"])`	
Bold a hover title.	`px.scatter(df, x="a", y="b", hover_name="country")`	
Align tooltips across a row.	`fig.update_layout(hovermode="x unified")`	
Fully custom tooltip text.	`fig.update_traces(hovertemplate= "%{x}: %{y:.1f}<extra></extra>")`	
Default drag = zoom; switch to pan.	`fig.update_layout(dragmode="pan")`	
Add a range slider.	`fig.update_xaxes(rangeslider_visible=True)`	

Every Plotly figure ships hover, drag-to-zoom, pan, and a mode-bar for free; tune them with hover_data/hovertemplate, hovermode, dragmode, and rangeslider

Color, Size & Animation Encodings

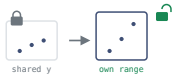
Map extra columns to color, marker size, symbol, and time.

PURPOSE	COMMAND	RESULT
Color by a category.	`px.scatter(df, x="a", y="b", color="species", color_discrete_map={"setosa": "#198754"})`	
Color by a number (gradient).	`px.scatter(df, x="a", y="b", color="petal_length", color_continuous_scale="Viridis")`	
Size by a value (bubbles).	`px.scatter(df, x="gdpPercap", y="lifeExp", size="pop", size_max=60)`	
Distinguish with shapes.	`px.scatter(df, x="a", y="b", symbol="species")`	
Animate over a frame variable.	`px.scatter(df, x="gdpPercap", y="lifeExp", size="pop", color="continent",animation_frame="year")`	
Lock axes so frames compare.	`px.scatter(df, ..., animation_frame="year", range_x=[100,1e5],range_y=[25,90],log_x=True)`	

color/size/symbol map columns to visual channels; animation_frame turns a column into a play slider, fixed range_x/range_y keep frames comparable

Facets & subplots

Small multiples with px facets; mixed grids with make_subplots.

PURPOSE	COMMAND	RESULT
Split into a row of panels.	`px.scatter(df, x="sepal_width", y="sepal_length", facet_col="species")`	
Grid by two variables.	`px.scatter(df, x="total_bill", y="tip", facet_row="sex", facet_col="time")`	
Wrap many facets.	`px.scatter(df, x="gdpPercap", y="lifeExp", facet_col="continent", facet_col_wrap=3)`	
Free each panel's axis.	`fig.update_yaxes(matches=None)`	
Mixed chart types in a grid.	`make_subplots(rows=2, cols=1, subplot_titles=("Price", "Volume"))`	
Place a trace in a cell.	`fig.add_trace(go.Bar(x=x, y=v), row=2, col=1)`	

px facets (facet_row / facet_col / facet_col_wrap) repeat one chart; make_subplots + add_trace(row=, col=) mix chart types in a grid

Build a Quick Dashboard

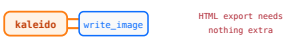
A theme, a notebook widget, and several figures on one HTML page.

PURPOSE	COMMAND	RESULT
Pick a global theme.	`import plotly.ioaspio` `pio.templates.default= "plotly_white"`	
Polish the layout.	`fig.update_layout(legend=dict(orientation="h"),` `margin=dict(l=40, r=20, t=40, b=40))`	
Add buttons / dropdowns.	`fig.update_layout(updatemenus=[...])`	
Live widget in a notebook.	`fw = go.FigureWidget(fig)`	
Stitch figures into one page.	`"".join(f.to_html(full_html=False,` `include_plotlyjs="cdn" if i==0 else False)…)`	
Want callbacks? Reach for Dash.	`# pip install dash` `app.layout = [dcc.Graph(figure=fig)]`	

set pio.templates.default, polish each layout, concat to_html fragments into one page; FigureWidget for live notebooks, Dash for real callbacks

Export to HTML and static image

Ship the live widget as HTML, or freeze a PNG/SVG/PDF.

PURPOSE	COMMAND	RESULT
Save a standalone interactive page.	`fig.write_html("chart.html")`	
Slim the page with a CDN script.	`fig.write_html("chart.html",` `include_plotlyjs="cdn")`	
Embed in an existing page.	`html = fig.to_html(full_html=False,` `include_plotlyjs=False)`	
Freeze a raster image.	`fig.write_image("chart.png",` `scale=2, width=900, height=500)`	
Vector / print output.	`fig.write_image("chart.svg" · "chart.pdf")`	
Needs the kaleido engine.	`pip install kaleido # static export`	

write_html ships a still-interactive page (no engine); write_image freezes a PNG/SVG/PDF and needs the kaleido backend