Subplots in Python

How to make subplots in with Plotly's Python graphing library. Examples of stacked, custom-sized, gridded, and annotated subplots.

Subplots and Plotly Express

Plotly Express (/python/plotly-express/) is the easy-to-use, high-level interface to Plotly, which operates on a variety of types of data (/python/px-arguments/) and produces easy-to-style figures (/python/styling-plotly-express/).

Plotly Express does not support arbitrary subplot capabilities, instead it supports faceting by a given data dimension (/python/facet-plots/), and it also supports marginal charts to display distribution information (/python/marginal-plots/).

This page documents the usage of the lower-level `plotly.subplots` module and the `make_subplots` function it exposes to construct figures with arbitrary subplots. **Plotly Express faceting uses** `make_subplots` **internally** so adding traces to Plotly Express facets works just as documented here, with `fig.add_trace(..., row=<R>, col=<C>)`.

Simple Subplot

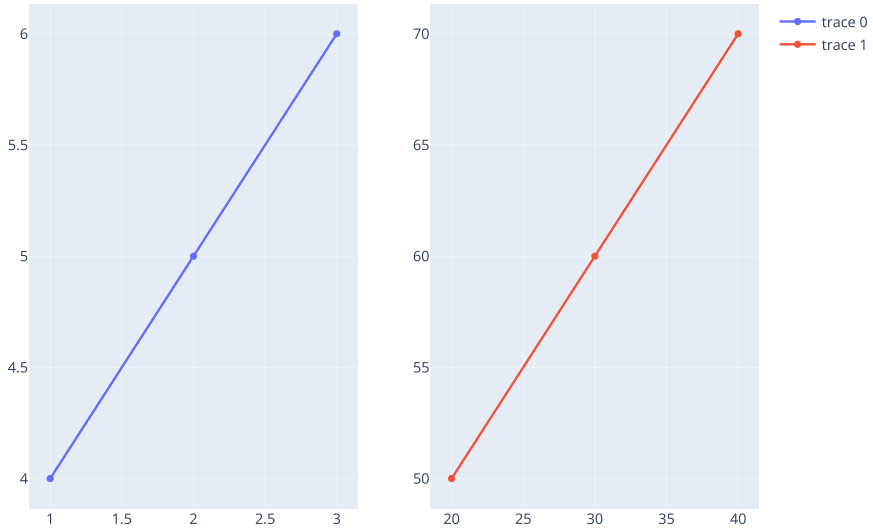
Figures with subplots are created using the `make_subplots` function from the `plotly.subplots` module.

Here is an example of creating a figure that includes two `scatter` traces which are side-by-side since there are 2 columns and 1 row in the subplot layout.

```
from plotly.subplots import make_subplots
import plotly.graph_objects as go

fig = make_subplots(rows=1, cols=2)

fig.add_trace(
    go.Scatter(x=[1, 2, 3], y=[4, 5, 6]),
    row=1, col=1
)

fig.add_trace(
    go.Scatter(x=[20, 30, 40], y=[50, 60, 70]),
    row=1, col=2
)

fig.update_layout(height=600, width=800, title_text="Side By Side Subplots")
fig.show()
```

and Row

Side By Side Subplots



Titles

l API)

xis (low-

/

Stacked Subplots

Here is an example of creating a figure with subplots that are stacked on top of each other since there are 3 rows and 1 column in the subplot layout.

and Row

```
from plotly.subplots import make_subplots
import plotly.graph_objects as go

fig = make_subplots(rows=3, cols=1)

fig.add_trace(go.Scatter(
    x=[3, 4, 5],
    y=[1000, 1100, 1200],
), row=1, col=1)

fig.add_trace(go.Scatter(
    x=[2, 3, 4],
    y=[100, 110, 120],
), row=2, col=1)

fig.add_trace(go.Scatter(
    x=[0, 1, 2],
    y=[10, 11, 12]
), row=3, col=1)


fig.update_layout(height=600, width=600, title_text="Stacked Subplots")
fig.show()
```

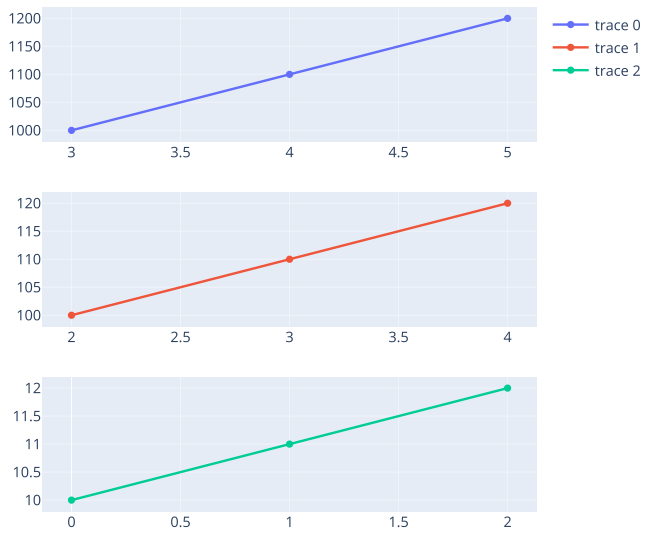
Titles

I API)

xis (low-

/

Stacked Subplots



Multiple Subplots

Here is an example of creating a 2 x 2 subplot grid and populating each subplot with a single scatter trace.

```
import plotly.graph_objects as go
from plotly.subplots import make_subplots

fig = make_subplots(rows=2, cols=2, start_cell="bottom-left")

fig.add_trace(go.Scatter(x=[1, 2, 3], y=[4, 5, 6]),
              row=1, col=1)

fig.add_trace(go.Scatter(x=[20, 30, 40], y=[50, 60, 70]),
              row=1, col=2)

fig.add_trace(go.Scatter(x=[300, 400, 500], y=[600, 700, 800]),
              row=2, col=1)

fig.add_trace(go.Scatter(x=[4000, 5000, 6000], y=[7000, 8000, 9000]),
              row=2, col=2)

fig.show()
```

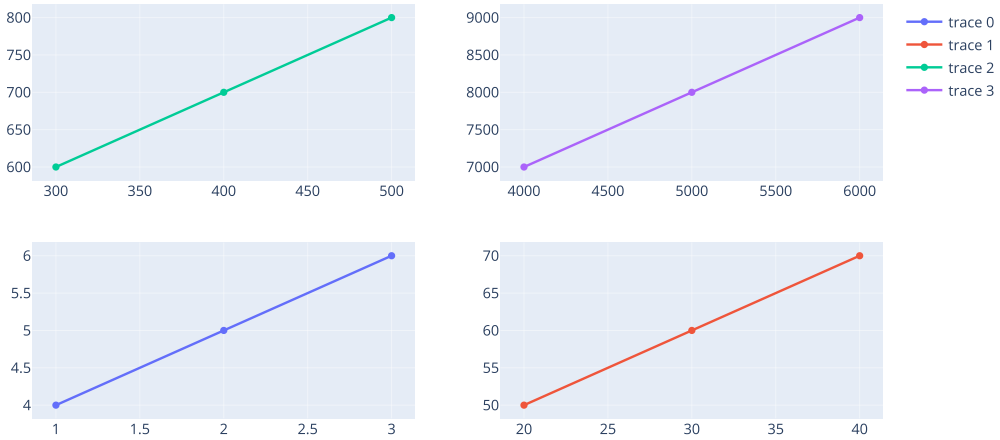
and Row

Titles

l API)

xis (low-

/



Multiple Subplots with Titles

The subplot_titles argument to make_subplots can be used to position text annotations as titles for each subplot.

Here is an example of adding subplot titles to a 2 x 2 subplot grid of scatter traces.

```
from plotly.subplots import make_subplots
import plotly.graph_objects as go

fig = make_subplots(
    rows=2, cols=2,
    subplot_titles=("Plot 1", "Plot 2", "Plot 3", "Plot 4"))

fig.add_trace(go.Scatter(x=[1, 2, 3], y=[4, 5, 6]),
              row=1, col=1)

fig.add_trace(go.Scatter(x=[20, 30, 40], y=[50, 60, 70]),
              row=1, col=2)

fig.add_trace(go.Scatter(x=[300, 400, 500], y=[600, 700, 800]),
              row=2, col=1)

fig.add_trace(go.Scatter(x=[4000, 5000, 6000], y=[7000, 8000, 9000]),
              row=2, col=2)

fig.update_layout(height=500, width=700,
                  title_text="Multiple Subplots with Titles")

fig.show()
```

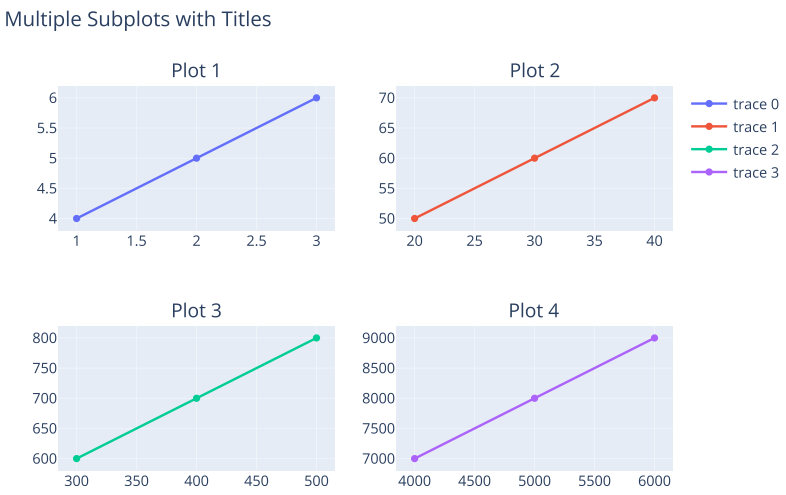
and Row

Titles

I API)

xis (low-

/



Subplots with Annotations

```
from plotly.subplots import make_subplots
import plotly.graph_objects as go

fig = make_subplots(rows=1, cols=2)

fig.add_trace(
    go.Scatter(
        x=[1, 2, 3],
        y=[4, 5, 6],
        mode="markers+text",
        text=["Text A", "Text B", "Text C"],
        textposition="bottom center"
    ),
    row=1, col=1
)

fig.add_trace(
    go.Scatter(
        x=[20, 30, 40],
        y=[50, 60, 70],
        mode="markers+text",
        text=["Text D", "Text E", "Text F"],
        textposition="bottom center"
    ),
    row=1, col=2
)

fig.update_layout(height=600, width=800, title_text="Subplots with Annotations")

fig.show()
```

and Row

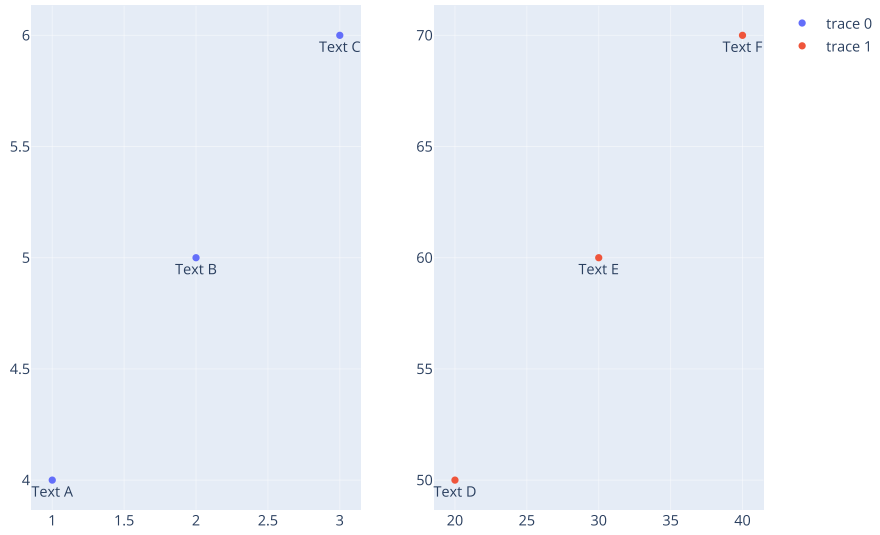
Titles

l API)

xis (low-

/

Subplots with Annotations



Customize Subplot Column Widths and Row Heights

The column_widths argument to make_subplots can be used to customize the relative widths of the columns in a subplot grid. It should be set to a list of numbers with a length that matches the cols argument. These number will be normalized, so that they sum to 1, and used to compute the relative widths of the subplot grid columns. The row_heights argument serves the same purpose for controlling the relative heights of rows in the subplot grid.

Here is an example of creating a figure with two scatter traces in side-by-side subplots. The left subplot is set to be wider than the right one.

```
import plotly.graph_objects as go
from plotly.subplots import make_subplots

fig = make_subplots(rows=1, cols=2, column_widths=[0.7, 0.3])

fig.add_trace(go.Scatter(x=[1, 2, 3], y=[4, 5, 6]),
              row=1, col=1)

fig.add_trace(go.Scatter(x=[20, 30, 40], y=[50, 60, 70]),
              row=1, col=2)

fig.show()
```

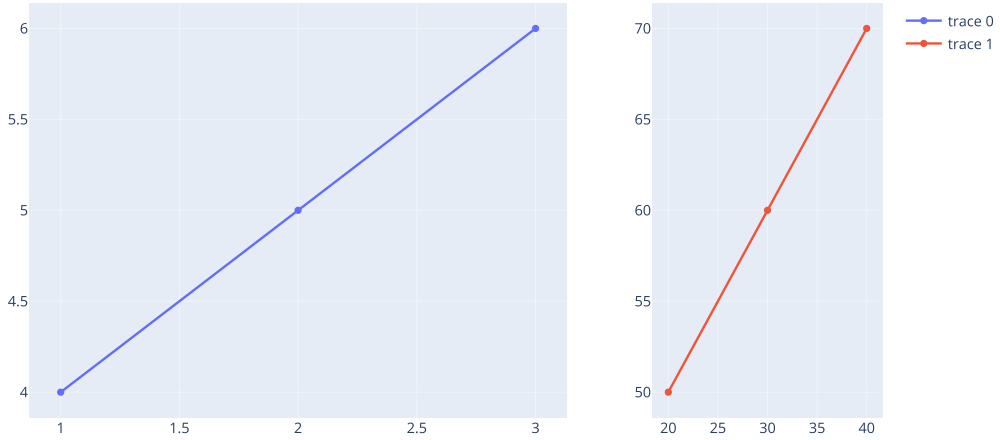
and Row

Titles

l API)

xis (low-

/



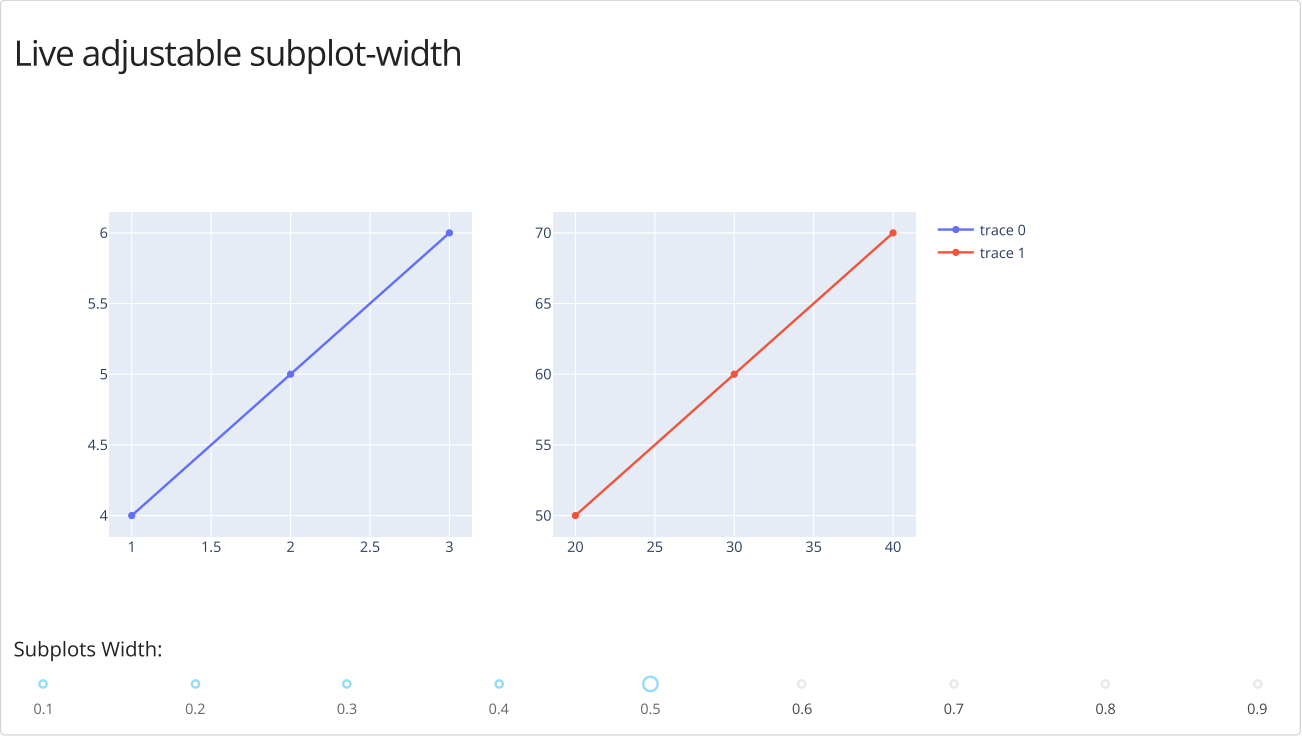
Subplots in Dash

Dash (https://plotly.com/dash/) is the best way to build analytical apps in Python using Plotly figures. To run the app below, run pip install dash, click "Download" to get the code and run python app.py.

Get started with the official Dash docs (https://dash.plotly.com/installation) and **learn how to effortlessly** style (https://plotly.com/dash/design-kit/) **&** deploy (https://plotly.com/dash/app-manager/) **apps like this with** Dash Enterprise (https://plotly.com/dash/)**.**

```
from dash import Dash, dcc, html, Input, Output
from plotly.subplots import make_subplots
import plotly.graph_objects as go

app = Dash(__name__)


app.layout = html.Div([
    html.H4('Live adjustable subplot-width'),
    dcc.Graph(id="graph"),
    html.P("Subplots Width:"),
    dcc.Slider(
        id='slider-width', min=.1, max=.9,
        value=0.5, step=0.1)
])


@app.callback(
    Output("graph", "figure"),
    Input("slider-width", "value"))
def customize_width(left_width):
    fig = make_subplots(rows=1, cols=2,
        column_widths=[left_width, 1 - left_width])

    fig.add_trace(row=1, col=1,
```

DOWNLOAD

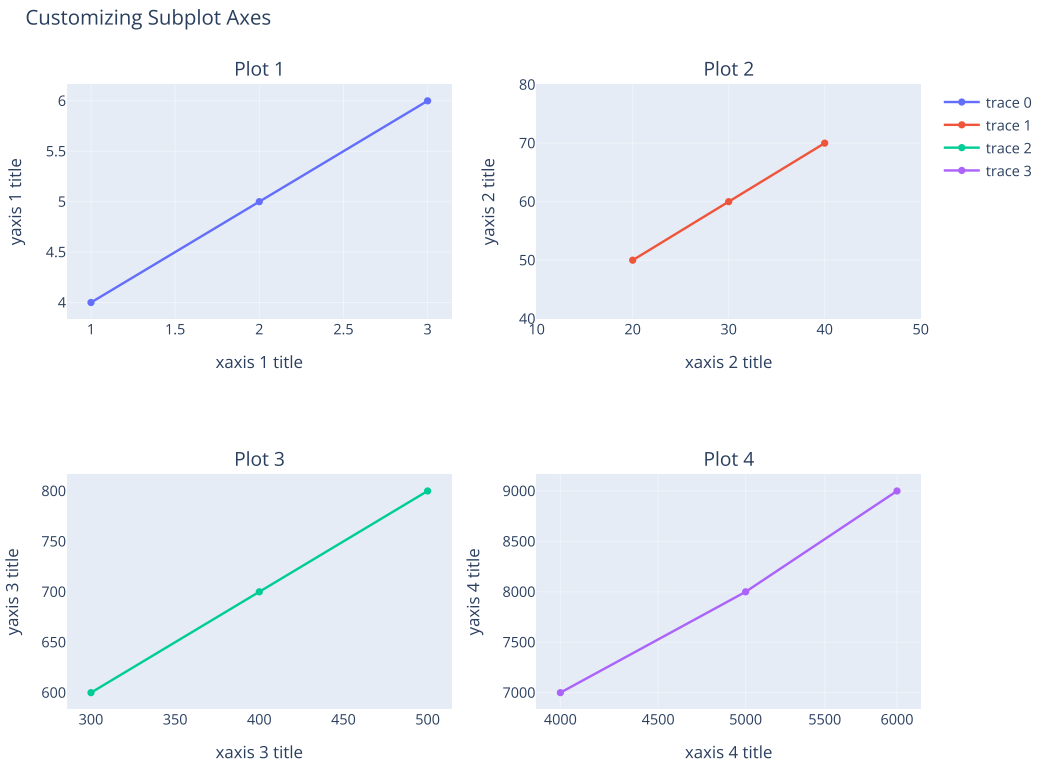


Customizing Subplot Axes

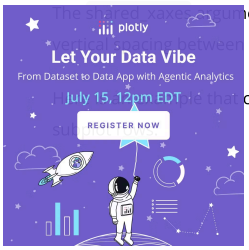
After a figure with subplots is created using the `make_subplots` function, its axis properties (title, font, range, grid style, etc.) can be customized using the `update_xaxes` and `update_yaxes` graph object figure methods. By default, these methods apply to all of the x axes or y axes in the figure. The row and col arguments can be used to control which axes are targeted by the update.

Here is an example that creates a figure with a 2 x 2 subplot grid, populates each subplot with a scatter trace, and then updates the x and y axis titles for each subplot individually:

```
from plotly.subplots import make_subplots
import plotly.graph_objects as go

# Initialize figure with subplots
fig = make_subplots(
    rows=2, cols=2, subplot_titles=("Plot 1", "Plot 2", "Plot 3", "Plot 4")
)

# Add traces
fig.add_trace(go.Scatter(x=[1, 2, 3], y=[4, 5, 6]), row=1, col=1)
fig.add_trace(go.Scatter(x=[20, 30, 40], y=[50, 60, 70]), row=1, col=2)
fig.add_trace(go.Scatter(x=[300, 400, 500], y=[600, 700, 800]), row=2, col=1)
fig.add_trace(go.Scatter(x=[4000, 5000, 6000], y=[7000, 8000, 9000]), row=2, col=2)

# Update xaxis properties
fig.update_xaxes(title_text="xaxis 1 title", row=1, col=1)
fig.update_xaxes(title_text="xaxis 2 title", range=[10, 50], row=1, col=2)
fig.update_xaxes(title_text="xaxis 3 title", showgrid=False, row=2, col=1)
fig.update_xaxes(title_text="xaxis 4 title", type="log", row=2, col=2)

# Update yaxis properties
fig.update_yaxes(title_text="yaxis 1 title", row=1, col=1)
fig.update_yaxes(title_text="yaxis 2 title", range=[40, 80], row=1, col=2)
fig.update_yaxes(title_text="yaxis 3 title", showgrid=False, row=2, col=1)
fig.update_yaxes(title_text="yaxis 4 title", row=2, col=2)

# Update title and height
fig.update_layout(title_text="Customizing Subplot Axes", height=700)

fig.show()
```



Subplots with Shared X-Axes



The shared_xaxes argument to make_subplots can be used to link the x axes of subplots in the resulting figure. The vertical_spacing argument is used to control the vertical spacing between rows in the subplot grid.

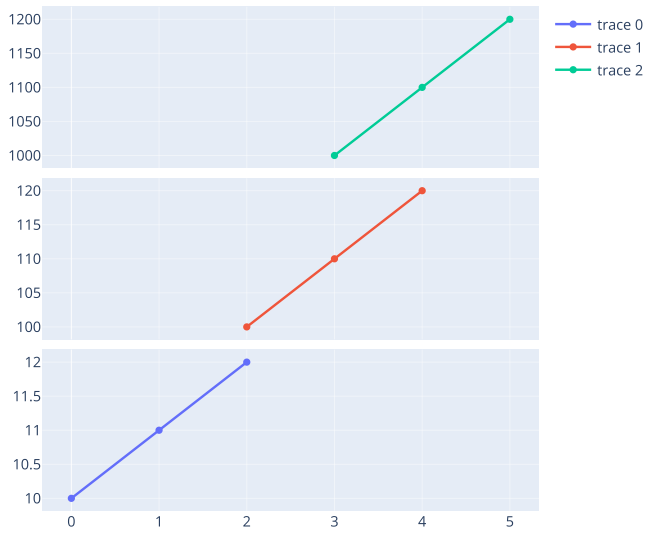
This example makes a subplot that creates a figure with 3 vertically stacked subplots with linked x axes. A small vertical spacing value is used to reduce the spacing between

```
from plotly.subplots import make_subplots
import plotly.graph_objects as go

fig = make_subplots(rows=3, cols=1,
                    shared_xaxes=True,
                    vertical_spacing=0.02)

fig.add_trace(go.Scatter(x=[0, 1, 2], y=[10, 11, 12]),
              row=3, col=1)

fig.add_trace(go.Scatter(x=[2, 3, 4], y=[100, 110, 120]),
              row=2, col=1)

fig.add_trace(go.Scatter(x=[3, 4, 5], y=[1000, 1100, 1200]),
              row=1, col=1)

fig.update_layout(height=600, width=600,
                  title_text="Stacked Subplots with Shared X-Axes")
fig.show()
```

and Row

Stacked Subplots with Shared X-Axes

Titles



l API)

xis (low-

/

Subplots with Shared Y-Axes

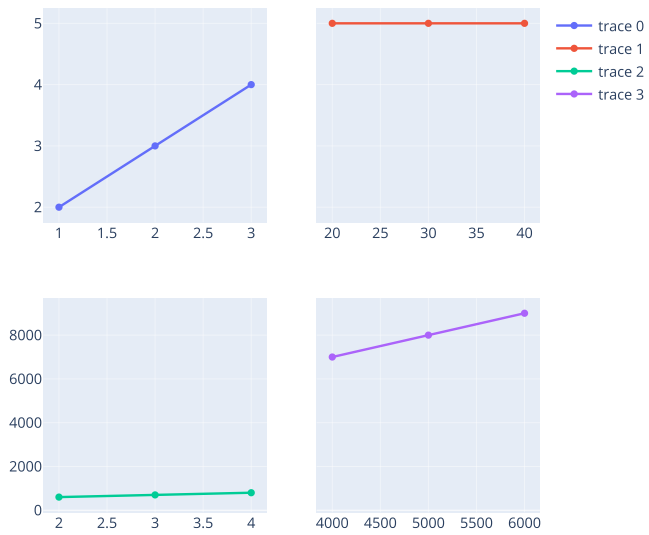
The `shared_yaxes` argument to `make_subplots` can be used to link the y axes of subplots in the resulting figure.

Here is an example that creates a figure with a 2 x 2 subplot grid, where the y axes of each row are linked.

```
from plotly.subplots import make_subplots
import plotly.graph_objects as go

fig = make_subplots(rows=2, cols=2, shared_yaxes=True)

fig.add_trace(go.Scatter(x=[1, 2, 3], y=[2, 3, 4]),
              row=1, col=1)

fig.add_trace(go.Scatter(x=[20, 30, 40], y=[5, 5, 5]),
              row=1, col=2)

fig.add_trace(go.Scatter(x=[2, 3, 4], y=[600, 700, 800]),
              row=2, col=1)

fig.add_trace(go.Scatter(x=[4000, 5000, 6000], y=[7000, 8000, 9000]),
              row=2, col=2)

fig.update_layout(height=600, width=600,
                  title_text="Multiple Subplots with Shared Y-Axes")
fig.show()
```

and Row

Multiple Subplots with Shared Y-Axes



Titles

l API)

xis (low-

/

Subplots with Shared Colorscale

To share colorscale information in multiple subplots, you can use coloraxis (https://plotly.com/javascript/reference/scatter/#scatter-marker-line-coloraxis).

```
from plotly.subplots import make_subplots
import plotly.graph_objects as go

fig = make_subplots(rows=1, cols=2, shared_yaxes=True)

fig.add_trace(go.Bar(x=[1, 2, 3], y=[4, 5, 6],
                     marker=dict(color=[4, 5, 6], coloraxis="coloraxis")),
              1, 1)

fig.add_trace(go.Bar(x=[1, 2, 3], y=[2, 3, 5],
                     marker=dict(color=[2, 3, 5], coloraxis="coloraxis")),
              1, 2)

fig.update_layout(coloraxis=dict(colorscale='Bluered_r'), showlegend=False)
fig.show()
```

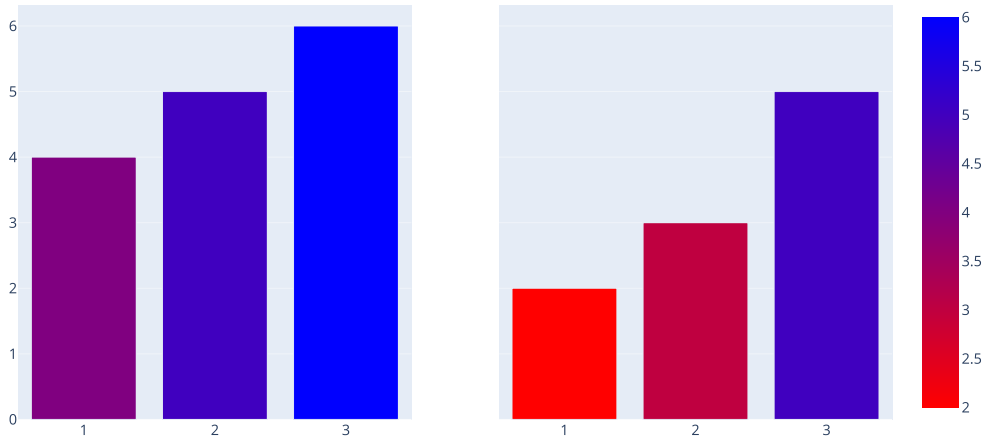
and Row

Titles

l API)

xis (low-

/



Custom Sized Subplot with Subplot Titles

The `specs` argument to `make_subplots` is used to configure per-subplot options. `specs` must be a 2-dimension list with dimensions that match those provided as the `rows` and `cols` arguments. The elements of `specs` may either be `None`, indicating no subplot should be initialized starting with this grid cell, or a dictionary containing subplot options. The `colspan` subplot option specifies the number of grid columns that the subplot starting in the given cell should occupy. If unspecified, `colspan` defaults to 1.

Here is an example that creates a 2 by 2 subplot grid containing 3 subplots. The subplot `specs` element for position (2, 1) has a `colspan` value of 2, causing it to span the full figure width. The subplot `specs` element for position (2, 2) is `None` because no subplot begins at this location in the grid.

```
from plotly.subplots import make_subplots
import plotly.graph_objects as go

fig = make_subplots(
    rows=2, cols=2,
    specs=[[{}, {}],
           [{"colspan": 2}, None]],
    subplot_titles=("First Subplot","Second Subplot", "Third Subplot"))

fig.add_trace(go.Scatter(x=[1, 2], y=[1, 2]),
                 row=1, col=1)

fig.add_trace(go.Scatter(x=[1, 2], y=[1, 2]),
                 row=1, col=2)
fig.add_trace(go.Scatter(x=[1, 2, 3], y=[2, 1, 2]),
                 row=2, col=1)

fig.update_layout(showlegend=False, title_text="Specs with Subplot Title")
fig.show()
```

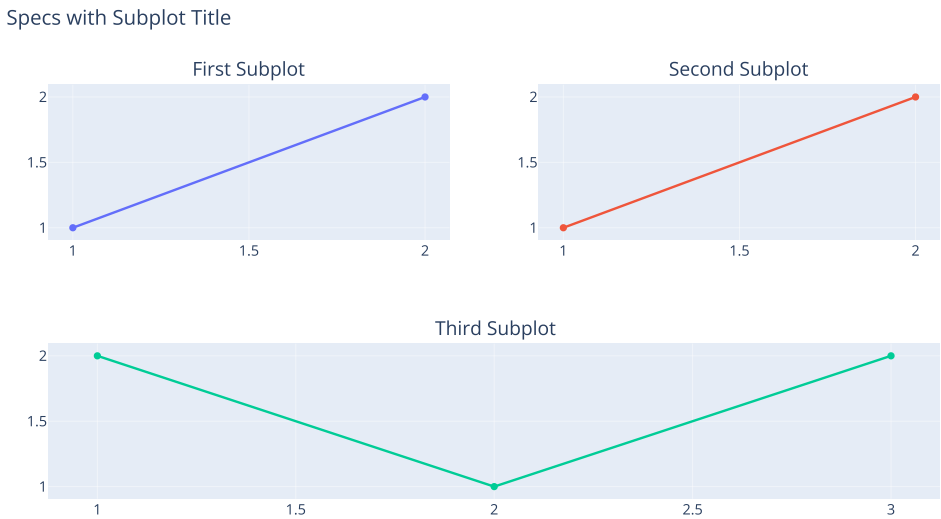
and Row

Titles

I API)

xis (low-

/



Multiple Custom Sized Subplots

If the print_grid argument to make_subplots is set to True, then a text representation of the subplot grid will be printed.

Here is an example that uses the rowspan and colspan subplot options to create a custom subplot layout with subplots of mixed sizes. The print_grid argument is set to True so that the subplot grid is printed to the screen.

```
from plotly.subplots import make_subplots
import plotly.graph_objects as go

fig = make_subplots(
    rows=5, cols=2,
    specs=[[{}, {"rowspan": 2}],
           [{}, None],
           [{"rowspan": 2, "colspan": 2}, None],
           [None, None],
           [{}, {}]],
    print_grid=True)

fig.add_trace(go.Scatter(x=[1, 2], y=[1, 2], name="(1,1)"), row=1, col=1)
fig.add_trace(go.Scatter(x=[1, 2], y=[1, 2], name="(1,2)"), row=1, col=2)
fig.add_trace(go.Scatter(x=[1, 2], y=[1, 2], name="(2,1)"), row=2, col=1)
fig.add_trace(go.Scatter(x=[1, 2], y=[1, 2], name="(3,1)"), row=3, col=1)
fig.add_trace(go.Scatter(x=[1, 2], y=[1, 2], name="(5,1)"), row=5, col=1)
fig.add_trace(go.Scatter(x=[1, 2], y=[1, 2], name="(5,2)"), row=5, col=2)

fig.update_layout(height=600, width=600, title_text="specs examples")
fig.show()
```

and Row

```
This is the format of your plot grid:
[ (1,1) x,y  ] [ (1,2) x2,y2 ]
[ (2,1) x3,y3 ] [    :    ]
[ (3,1) x4,y4        -    ]
[    :          :    ]
[ (5,1) x5,y5 ] [ (5,2) x6,y6 ]
```

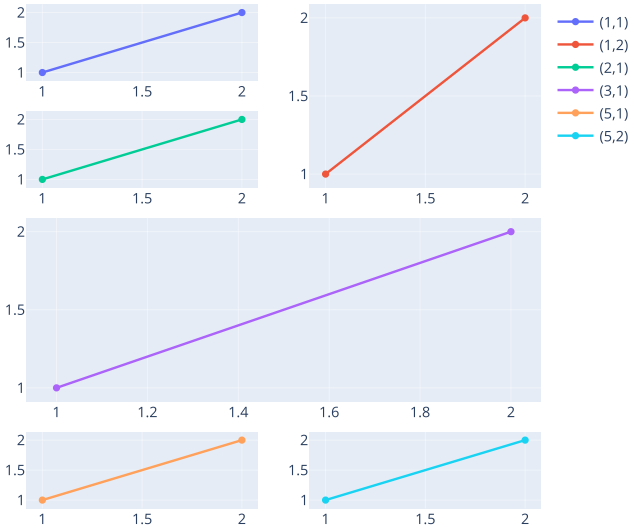
Titles

I API)

xis (low-

/

specs examples



Subplots Types

By default, the make_subplots function assumes that the traces that will be added to all subplots are 2-dimensional cartesian traces (e.g. scatter, bar, histogram, violin, etc.). Traces with other subplot types (e.g. scatterpolar, scattergeo, parcoords, etc.) are supported by specifying the type subplot option in the specs argument to make_subplots.

Here are the possible values for the type option:

- "xy": 2D Cartesian subplot type for scatter, bar, etc. This is the default if no type is specified.
- "scene": 3D Cartesian subplot for scatter3d, cone, etc.
- "polar": Polar subplot for scatterpolar, barpolar, etc.
- "ternary": Ternary subplot for scatterternary.
- "mapbox": Mapbox subplot for scattermapbox.
- "domain": Subplot type for traces that are individually positioned. pie, parcoords, parcats, etc.
- trace type: A trace type name (e.g. "bar", "scattergeo", "carpet", "mesh", etc.) which will be used to determine the appropriate subplot type for that trace.

Here is an example that creates and populates a 2 x 2 subplot grid containing 4 different subplot types.

and Row

Titles

l API)

xis (low-

/

```
from plotly.subplots import make_subplots
import plotly.graph_objects as go

fig = make_subplots(
    rows=2, cols=2,
    specs=[[{"type": "xy"}, {"type": "polar"}],
           [{"type": "domain"}, {"type": "scene"}]],
)

fig.add_trace(go.Bar(y=[2, 3, 1]),
              row=1, col=1)

fig.add_trace(go.Barpolar(theta=[0, 45, 90], r=[2, 3, 1]),
              row=1, col=2)

fig.add_trace(go.Pie(values=[2, 3, 1]),
              row=2, col=1)

fig.add_trace(go.Scatter3d(x=[2, 3, 1], y=[0, 0, 0],
                           z=[0.5, 1, 2], mode="lines"),
              row=2, col=2)

fig.update_layout(height=700, showlegend=False)

fig.show()
```

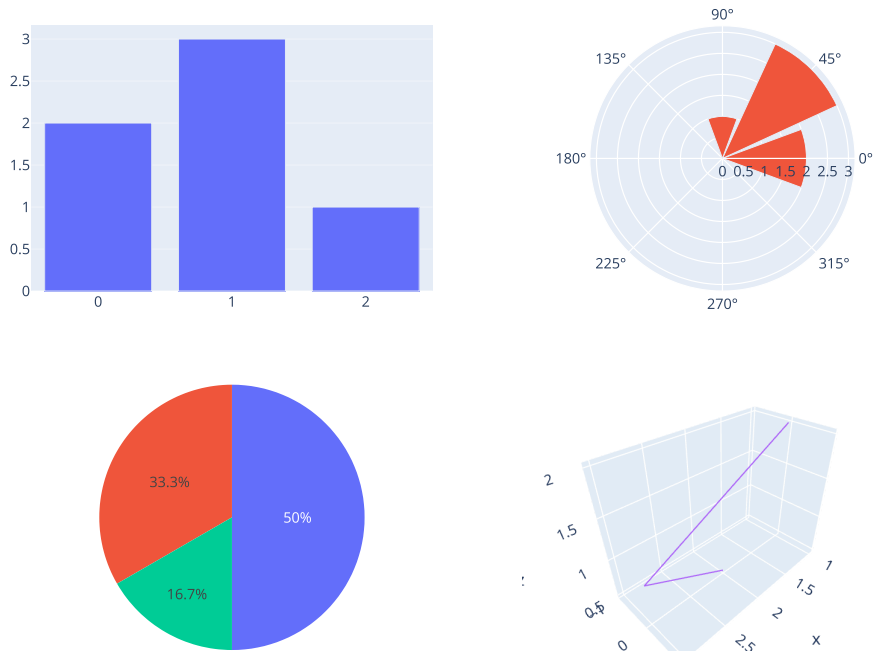
and Row

Titles

l API)

xis (low-

/



As an alternative to providing the name of a subplot type (e.g. "xy", "polar", "domain", "scene", etc), the type option may also be set to a string containing the name of a trace type (e.g. "bar", "barpolar", "pie", "scatter3d", etc.), which will be used to determine the subplot type that is compatible with that trace.

Here is the example above, modified to specify the subplot types using trace type names.

```
from plotly.subplots import make_subplots
import plotly.graph_objects as go

fig = make_subplots(
    rows=2, cols=2,
    specs=[[{"type": "bar"}, {"type": "barpolar"}],
           [{"type": "pie"}, {"type": "scatter3d"}]],
)

fig.add_trace(go.Bar(y=[2, 3, 1]),
              row=1, col=1)

fig.add_trace(go.Barpolar(theta=[0, 45, 90], r=[2, 3, 1]),
              row=1, col=2)

fig.add_trace(go.Pie(values=[2, 3, 1]),
              row=2, col=1)

fig.add_trace(go.Scatter3d(x=[2, 3, 1], y=[0, 0, 0],
                           z=[0.5, 1, 2], mode="lines"),
              row=2, col=2)

fig.update_layout(height=700, showlegend=False)

fig.show()
```

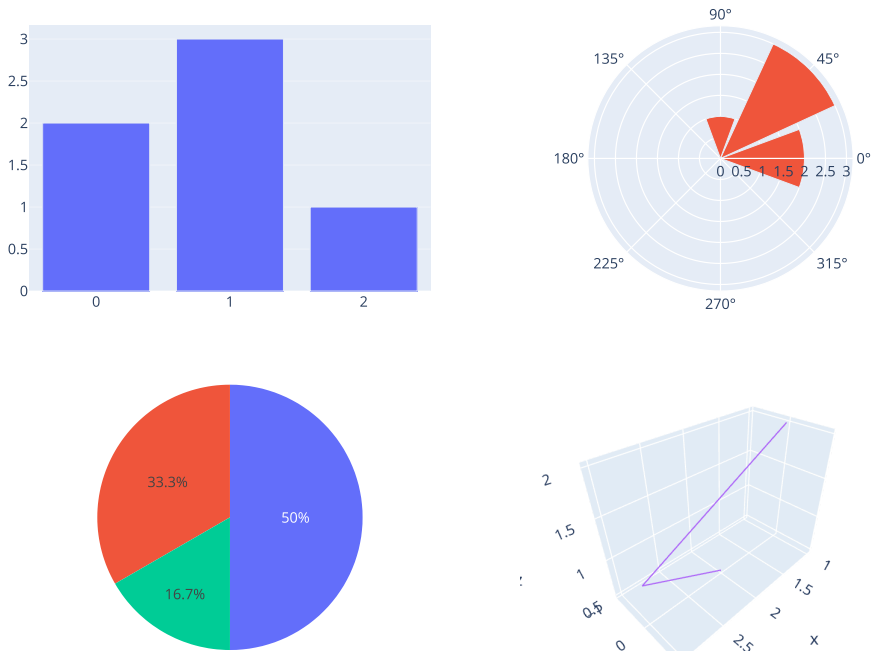
and Row

Titles

l API)

xis (low-

/



Side by Side Subplot (low-level API)

and Row

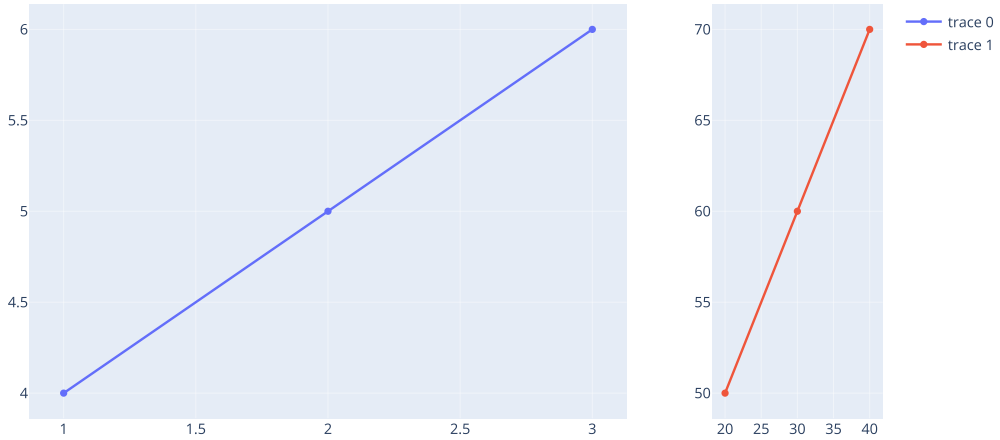
Titles

I API)

xis (low-

/

```
import plotly.graph_objects as go

trace1 = go.Scatter(
    x=[1, 2, 3],
    y=[4, 5, 6]
)
trace2 = go.Scatter(
    x=[20, 30, 40],
    y=[50, 60, 70],
    xaxis="x2",
    yaxis="y2"
)
data = [trace1, trace2]
layout = go.Layout(
    xaxis=dict(
        domain=[0, 0.7]
    ),
    xaxis2=dict(
        domain=[0.8, 1]
    ),
    yaxis2=dict(
        anchor="x2"
    )
)
fig = go.Figure(data=data, layout=layout)
fig.show()
```



Subplots with shared axes (low-level API)

and Row

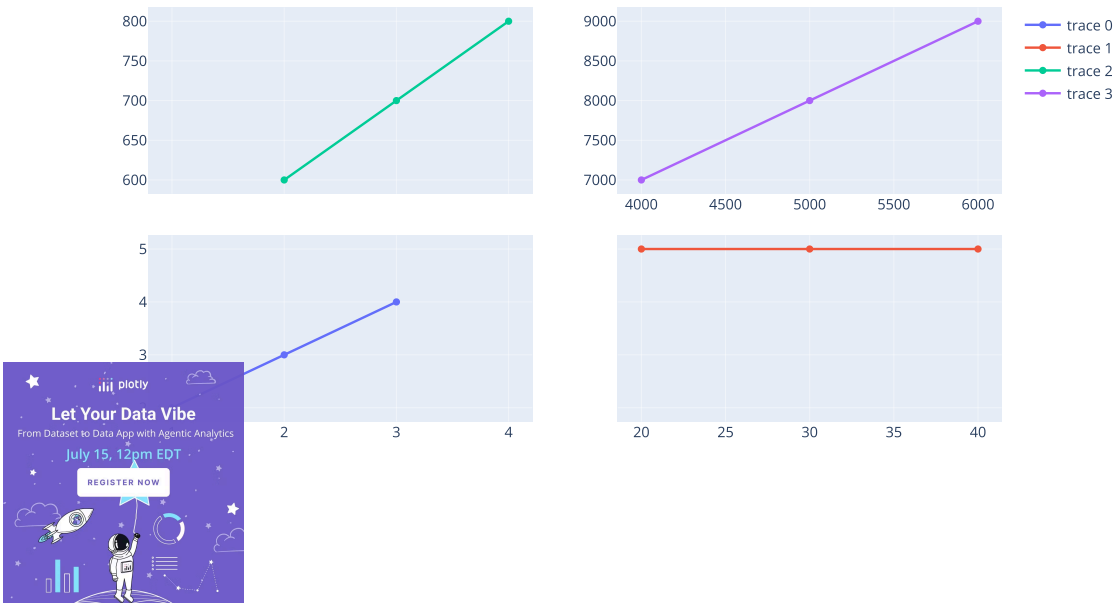
Titles

l API)

xis (low-

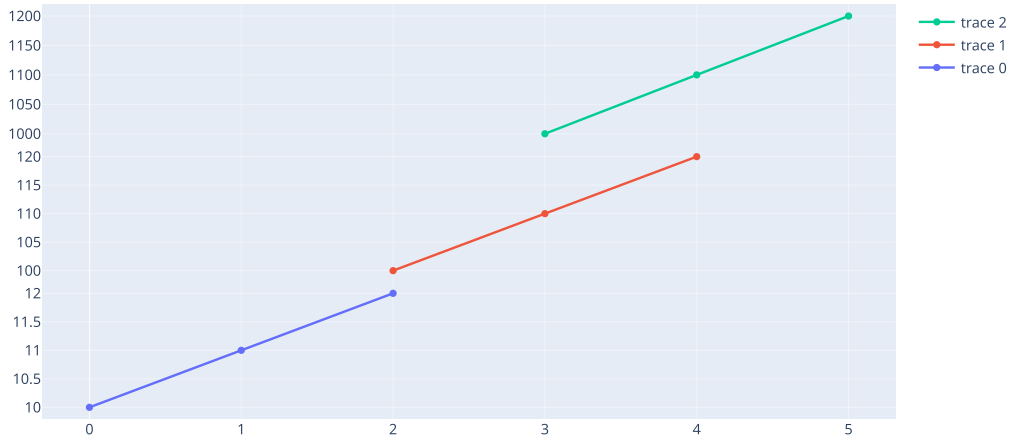
/

```
import plotly.graph_objects as go

trace1 = go.Scatter(
    x=[1, 2, 3],
    y=[2, 3, 4]
)
trace2 = go.Scatter(
    x=[20, 30, 40],
    y=[5, 5, 5],
    xaxis="x2",
    yaxis="y"
)
trace3 = go.Scatter(
    x=[2, 3, 4],
    y=[600, 700, 800],
    xaxis="x",
    yaxis="y3"
)
trace4 = go.Scatter(
    x=[4000, 5000, 6000],
    y=[7000, 8000, 9000],
    xaxis="x4",
    yaxis="y4"
)
data = [trace1, trace2, trace3, trace4]
layout = go.Layout(
    xaxis=dict(
        domain=[0, 0.45]
    ),
    yaxis=dict(
        domain=[0, 0.45]
    ),
    xaxis2=dict(
        domain=[0.55, 1]
    ),
    xaxis4=dict(
        domain=[0.55, 1],
        anchor="y4"
    ),
    yaxis3=dict(
        domain=[0.55, 1]
    ),
    yaxis4=dict(
        domain=[0.55, 1],
        anchor="x4"
    )
)
fig = go.Figure(data=data, layout=layout)
fig.show()
```



Stacked Subplots with a Shared X-Axis (low-level API)

```
import plotly.graph_objects as go

trace1 = go.Scatter(
    x=[0, 1, 2],
    y=[10, 11, 12]
)
trace2 = go.Scatter(
    x=[2, 3, 4],
    y=[100, 110, 120],
    yaxis="y2"
)
trace3 = go.Scatter(
    x=[3, 4, 5],
    y=[1000, 1100, 1200],
    yaxis="y3"
)
data = [trace1, trace2, trace3]
layout = go.Layout(
    yaxis=dict(
        domain=[0, 0.33]
    ),
    legend=dict(
        traceorder="reversed"
    ),
    yaxis2=dict(
        domain=[0.33, 0.66]
    ),
    yaxis3=dict(
        domain=[0.66, 1]
    )
)
fig = go.Figure(data=data, layout=layout)
fig.show()
```



Setting Subplots on a Figure Directly

new in 4.13

Subplots can be added to an already existing figure, provided it doesn't already have subplots. go.Figure.set_subplots accepts all the same arguments as make_subplots.

```
import plotly.graph_objects as go
fig = go.Figure().set_subplots(2, 3, horizontal_spacing=0.1)
```

is equivalent to:

```
from plotly.subplots import make_subplots
fig = make_subplots(2, 3, horizontal_spacing=0.1)
```

Reference

All of the x-axis properties are found here: https://plotly.com/python/reference/layout/xaxis/ (https://plotly.com/python/reference/layout/xaxis/) All of the y-axis properties are found here: https://plotly.com/python/reference/layout/yaxis/ (https://plotly.com/python/reference/layout/yaxis/)

and Row

```
from plotly.subplots import make_subplots
help(make_subplots)
```

Titles

l API)

xis (low-

/

```
Help on function make_subplots in module plotly.subplots:

make_subplots(rows=1, cols=1, shared_xaxes=False, shared_yaxes=False, start_cell='top-left', print_grid=False, horizontal_spacing=None, vertical_spacing=None, subplot_titles=None, column_widths=None, row_heights=None, specs=None, insets=None, column_titles=None, row_titles=None, x_title=None, y_title=None, figure=None, **kwargs) -> plotly.graph_objs._figure.Figure
    Return an instance of plotly.graph_objs.Figure with predefined subplots
    configured in 'layout'.

    Parameters
    ----------
    rows: int (default 1)
        Number of rows in the subplot grid. Must be greater than zero.
```

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
```

and Row

Titles

l API)

xis (low-

/