2D Histograms in Python

How to make 2D Histograms in Python with Plotly.

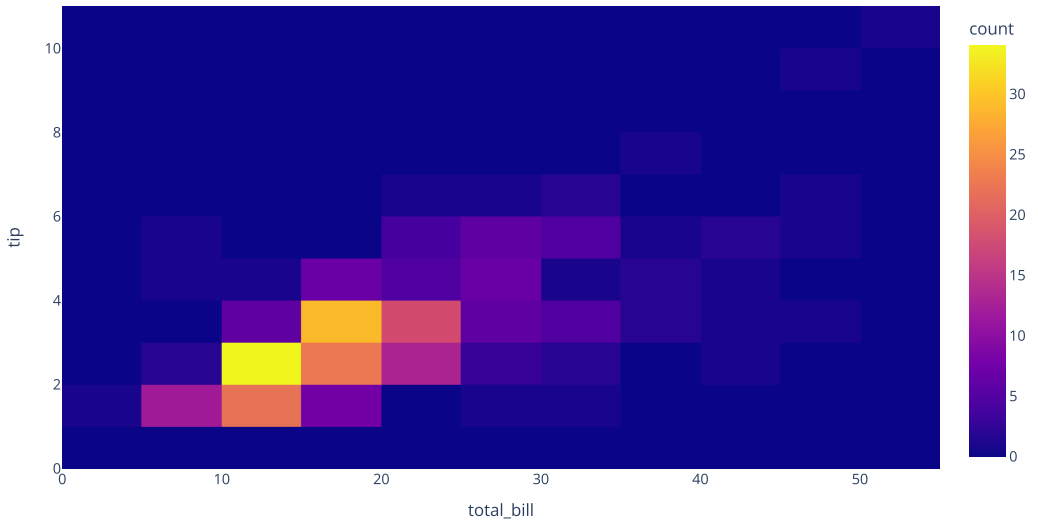
2D Histograms or Density Heatmaps

A 2D histogram, also known as a density heatmap, is the 2-dimensional generalization of a histogram (/python/histograms/) which resembles a heatmap (/python/heatmaps/) but is computed by grouping a set of points specified by their x and y coordinates into bins, and applying an aggregation function such as count or sum (if z is provided) to compute the color of the tile representing the bin. This kind of visualization (and the related 2D histogram contour, or density contour (https://plotly.com/python/2d-histogram-contour/)) is often used to manage over-plotting, or situations where showing large data sets as scatter plots (/python/line-and-scatter/) would result in points overlapping each other and hiding patterns. For data sets of more than a few thousand points, a better approach than the ones listed here would be to use Plotly with Datashader (/python/datashader/) to precompute the aggregations before displaying the data with Plotly.

Density Heatmaps with 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/). The Plotly Express function density_heatmap() can be used to produce density heatmaps.

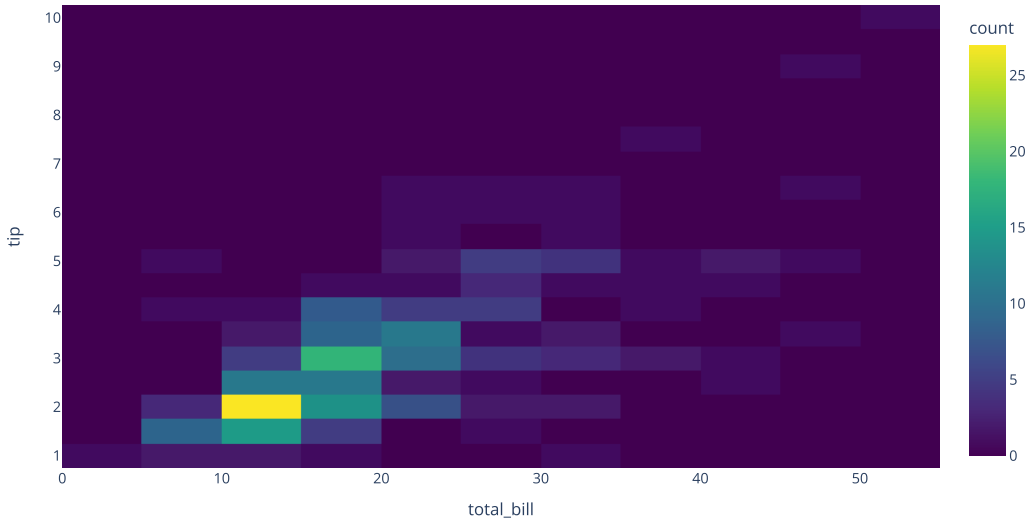
```
import plotly.express as px
df = px.data.tips()

fig = px.density_heatmap(df, x="total_bill", y="tip")
fig.show()
```



The number of bins can be controlled with nbinsx and nbinsy and the color scale (/python/colorscales/) with color_continuous_scale.

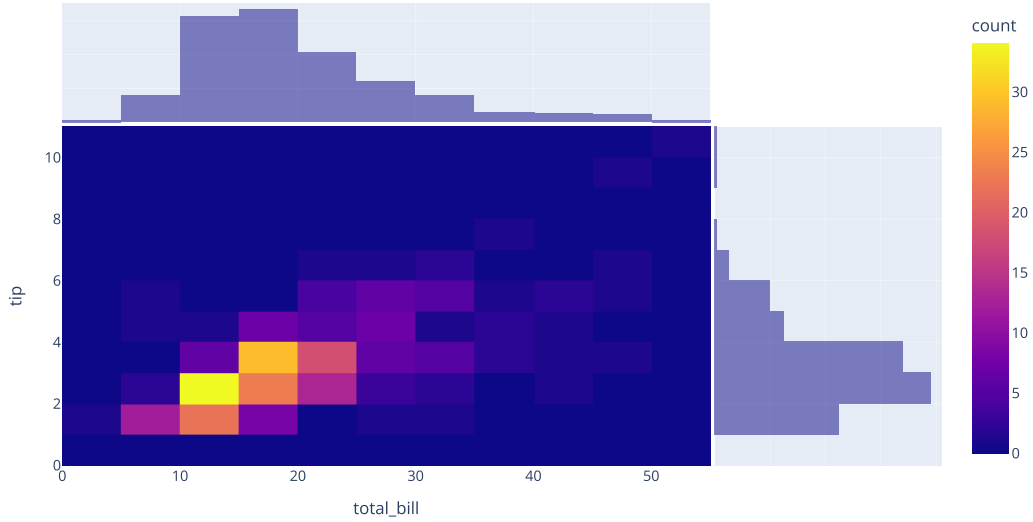
```
import plotly.express as px
df = px.data.tips()

fig = px.density_heatmap(df, x="total_bill", y="tip", nbinsx=20, nbinsy=20, color_continuous_scale="Viridis")
fig.show()
```

s
ss
ount
ptions
stograms
er Chart



Marginal plots can be added to visualize the 1-dimensional distributions of the two variables. Here we use a marginal histogram (/python/histograms/). Other allowable values are violin, box and rug.

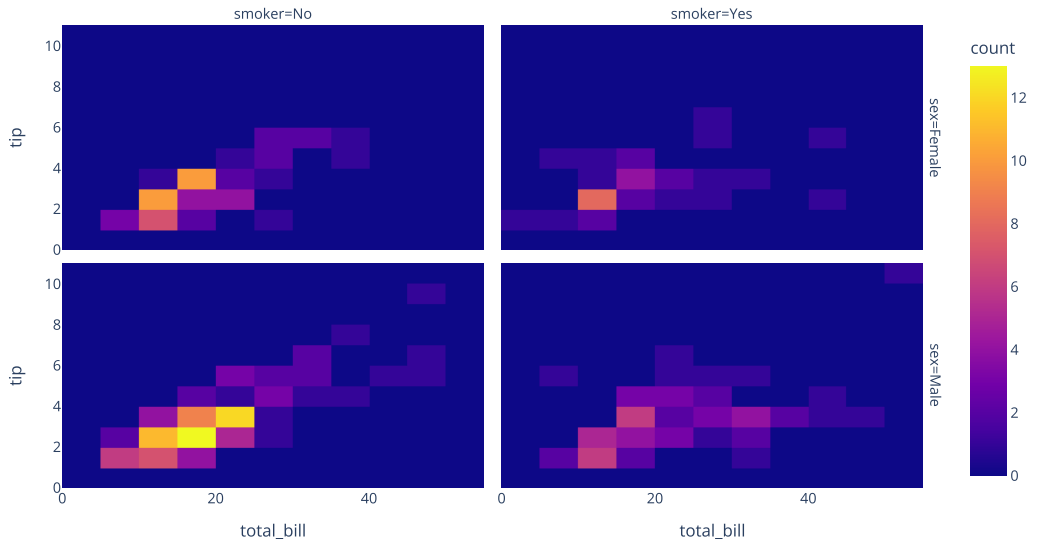
```
import plotly.express as px
df = px.data.tips()

fig = px.density_heatmap(df, x="total_bill", y="tip", marginal_x="histogram", marginal_y="histogram")
fig.show()
```



can also be faceted (/python/facet-plots/):

```
import plotly.express as px
df = px.data.tips()

fig = px.density_heatmap(df, x="total_bill", y="tip", facet_row="sex", facet_col="smoker")
fig.show()
```

s
ss
ount
ptions
stograms
er Chart

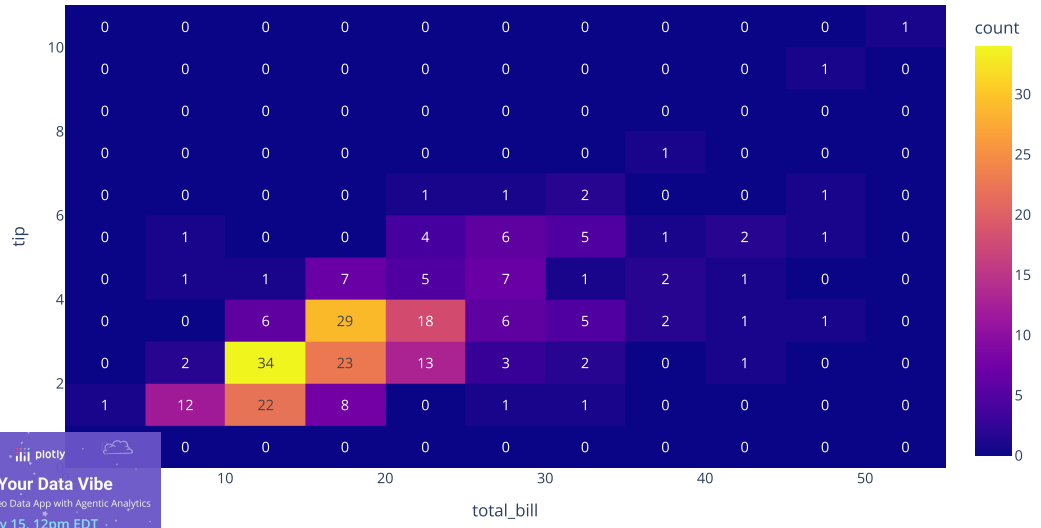


Displaying Text

New in v5.5

You can add the z values as text using the text_auto argument. Setting it to True will display the values on the bars, and setting it to a d3-format formatting string will control the output format.

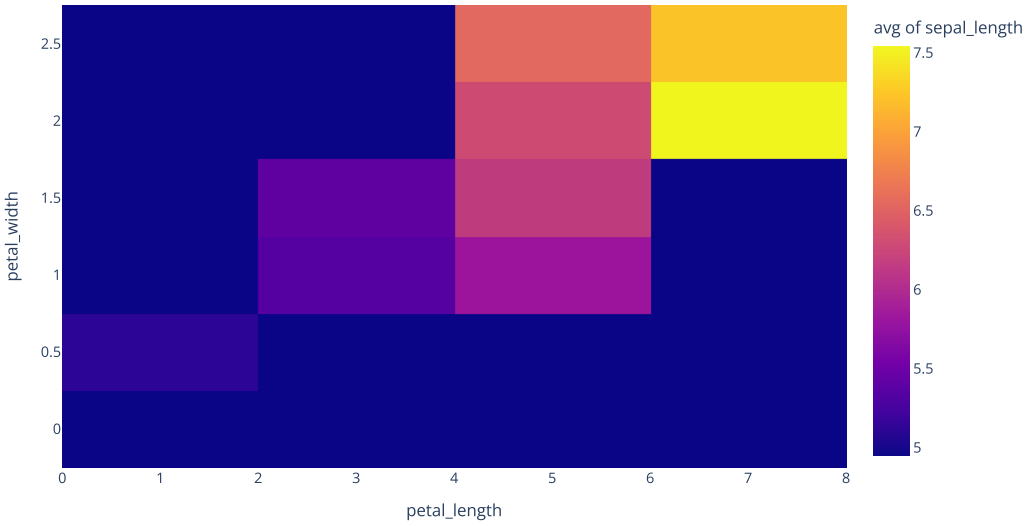
```
import plotly.express as px
df = px.data.tips()

fig = px.density_heatmap(df, x="total_bill", y="tip", text_auto=True)
fig.show()
```



Other aggregation functions than count

By passing in a z value and a histfunc, density heatmaps can perform basic aggregation operations. Here we show average Sepal Length grouped by Petal Length and Petal Width for the Iris dataset.

```
import plotly.express as px
df = px.data.iris()

fig = px.density_heatmap(df, x="petal_length", y="petal_width", z="sepal_length", histfunc="avg")
fig.show()
```

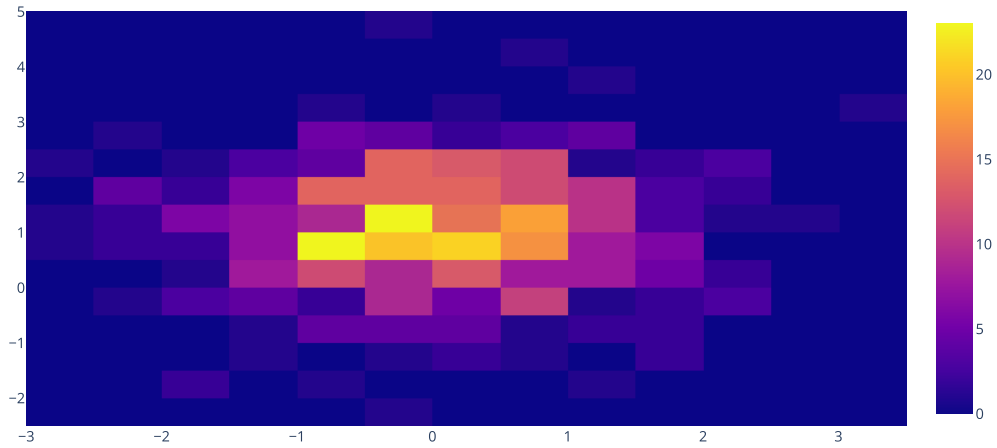


2D Histograms with Graph Objects

To build this kind of figure using graph objects (/python/graph-objects/) without using Plotly Express, we can use the go.Histogram2d class.

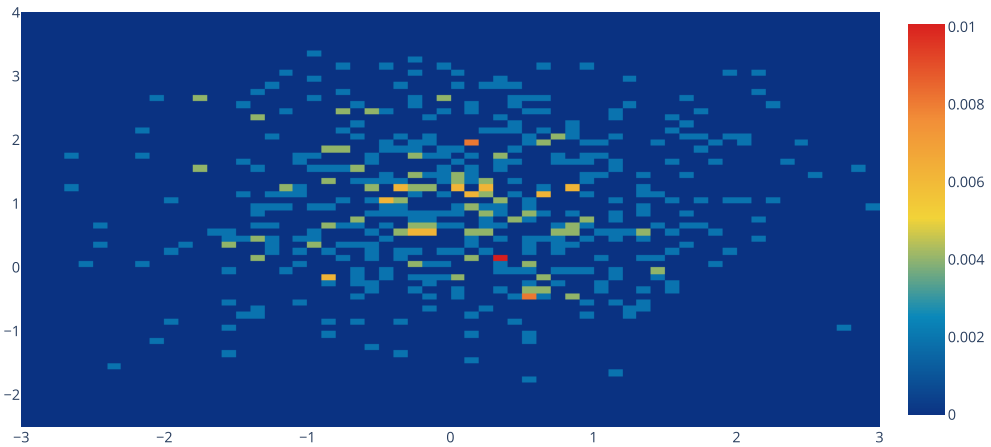
2D Histogram of a Bivariate Normal Distribution

```
import plotly.graph_objects as go

import numpy as np
np.random.seed(1)

x = np.random.randn(500)
y = np.random.randn(500)+1

fig = go.Figure(go.Histogram2d(
        x=x,
        y=y
    ))
fig.show()
```



2D Histogram Binning and Styling Options

```
import plotly.graph_objects as go

import numpy as np

x = np.random.randn(500)
y = np.random.randn(500)+1

fig = go.Figure(go.Histogram2d(x=x, y=y, histnorm='probability',
        autobinx=False,
        xbins=dict(start=-3, end=3, size=0.1),
        autobiny=False,
        ybins=dict(start=-2.5, end=4, size=0.1),
        colorscale=[[0, 'rgb(12,51,131)'], [0.25, 'rgb(10,136,186)'], [0.5, 'rgb(242,211,56)'], [0.75, 'rgb(242,143,56)'], [1, 'rgb(217,30,30)']]
    ))
fig.show()
```

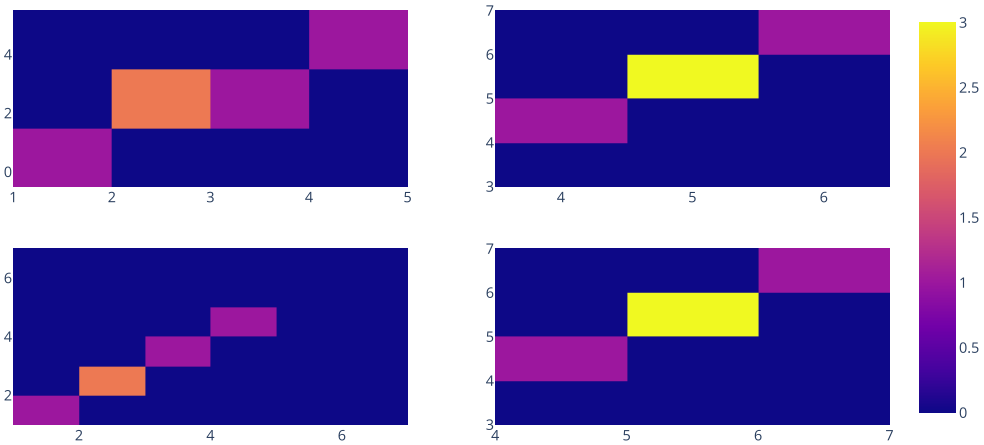


Sharing bin settings between 2D Histograms

This example shows how to use bingroup (https://plotly.com/python/reference/histogram/#histogram-bingroup) attribute to have a compatible bin settings for both histograms. To define start, end and size value of x-axis and y-axis separately, set ybins (https://plotly.com/python/reference/histogram2dcontour/#histogram2dcontour-ybins) and xbins.

s
ss
ount
ptions
stograms
er Chart

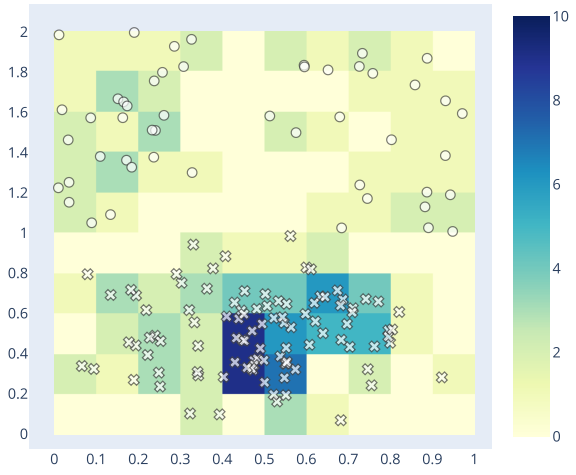
```
import plotly.graph_objects as go
from plotly.subplots import make_subplots

fig = make_subplots(2,2)
fig.add_trace(go.Histogram2d(
    x = [ 1, 2, 2, 3, 4 ],
    y = [ 1, 2, 2, 3, 4 ],
    coloraxis = "coloraxis",
    xbins = {'start':1, 'size':1}), 1,1)
fig.add_trace(go.Histogram2d(
    x = [ 4, 5, 5, 5, 6 ],
    y = [ 4, 5, 5, 5, 6 ],
    coloraxis = "coloraxis",
    ybins = {'start': 3, 'size': 1}),1,2)
fig.add_trace(go.Histogram2d(
    x = [ 1, 2, 2, 3, 4 ],
    y = [ 1, 2, 2, 3, 4 ],
    bingroup = 1,
    coloraxis = "coloraxis",
    xbins = {'start':1, 'size':1}), 2,1)
fig.add_trace(go.Histogram2d(
    x = [ 4, 5, 5, 5, 6 ],
    y = [ 4, 5, 5, 5, 6 ],
    bingroup = 1,
    coloraxis = "coloraxis",
    ybins = {'start': 3, 'size': 1}),2,2)
fig.show()
```



2D Histogram Overlaid with a Scatter Chart

```
import plotly.graph_objects as go

import numpy as np

x0 = np.random.randn(100)/5. + 0.5  # 5. enforces float division
y0 = np.random.randn(100)/5. + 0.5
x1 = np.random.rand(50)
y1 = np.random.rand(50) + 1.0

x = np.concatenate([x0, x1])
y = np.concatenate([y0, y1])

fig = go.Figure()

fig.add_trace(go.Scatter(
    x=x0,
    y=y0,
    mode='markers',
    showlegend=False,
    marker=dict(
        symbol='x',
        opacity=0.7,
        color='white',
        size=8,
        line=dict(width=1),
    )
))
fig.add_trace(go.Scatter(
    x=x1,
    y=y1,
    mode='markers',
    showlegend=False,
    marker=dict(
        symbol='circle',
        opacity=0.7,
        color='white',
        size=8,
        line=dict(width=1),
    )
))
fig.add_trace(go.Histogram2d(
    x=x,
    y=y,
    colorscale='YlGnBu',
    zmax=10,
    nbinsx=14,
    nbinsy=14,
    zauto=False,
))

fig.update_layout(
    xaxis=dict( ticks='', showgrid=False, zeroline=False, nticks=20 ),
    yaxis=dict( ticks='', showgrid=False, zeroline=False, nticks=20 ),
    autosize=False,
    height=550,
    width=550,
    hovermode='closest',

)

fig.show()
```

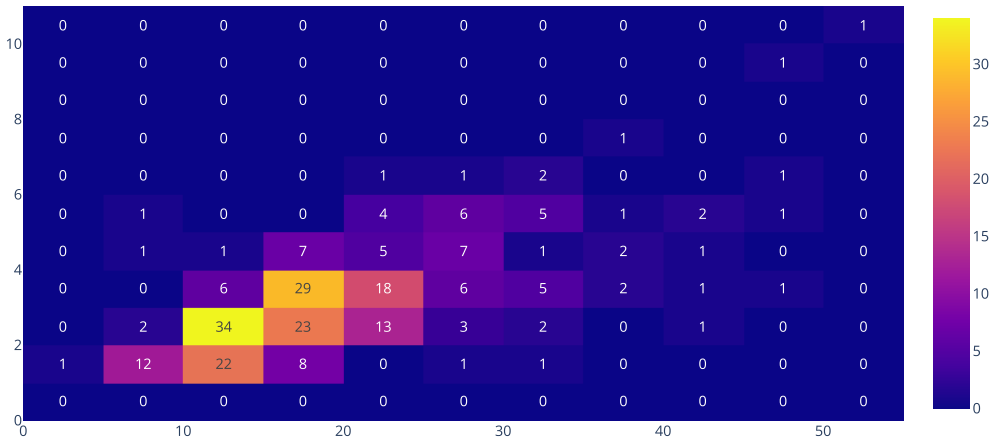
s
ss
ount
ptions
stograms
er Chart



Text on 2D Histogram Points

In this example we add text to 2D Histogram points. We use the values from the z attribute for the text.

```
import plotly.graph_objects as go
from plotly import data

df = data.tips()

fig = go.Figure(go.Histogram2d(
        x=df.total_bill,
        y=df.tip,
        texttemplate= "%{z}"
    ))

fig.show()
```



See https://plotly.com/python/reference/histogram2d/ (https://plotly.com/python/reference/histogram2d/) for more information and chart attribute options!

s
ss
ount
ptions
stograms
er Chart

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
```


JOIN OUR MAILING LIST

Sign up to stay in the loop with all things Plotly — from Dash Club to product updates, webinars, and more!

SUBSCRIBE (HTTPS://GO.PLOT.LY/SUBSCRIPTION)

Products

Dash (https://plotly.com/dash/)
Consulting and Training (https://plotly.com/consulting-and-oem/)

Pricing

Enterprise Pricing (https://plotly.com/get-pricing/)

About Us

Careers (https://plotly.com/careers)
Resources (https://plotly.com/resources/)
Blog (https://medium.com/@plotlygraphs)

Support

Community Support (https://community.plot.ly/)
Documentation (https://plotly.com/graphing-libraries)

Terms of Service (https://community.plotly.com/tos) Privacy Policy (https://plotly.com/privacy/)