Violin Plots in Python

How to make violin plots in Python with Plotly.

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)

Violin Plot with Plotly Express

A violin plot (https://en.wikipedia.org/wiki/Violin_plot) is a statistical representation of numerical data. It is similar to a box plot (https://plotly.com/python/box-plots/), with the addition of a rotated kernel density (https://en.wikipedia.org/wiki/Kernel_density_estimation) plot on each side.

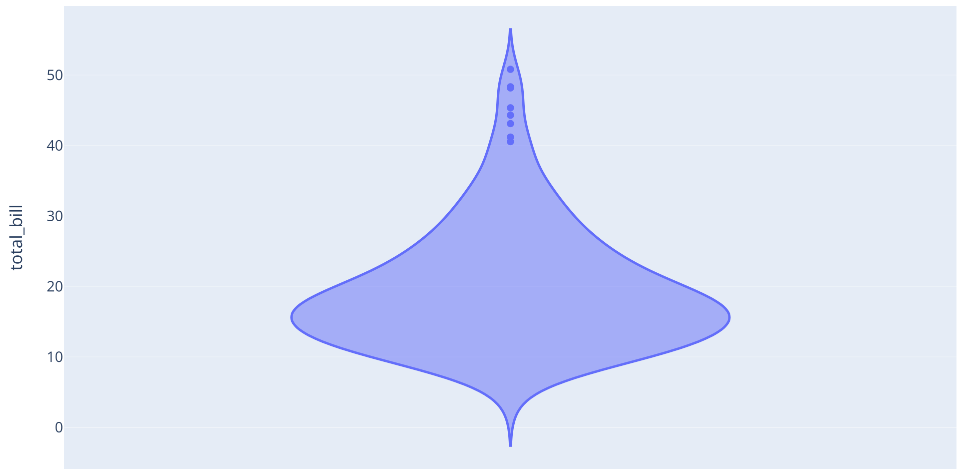
ting

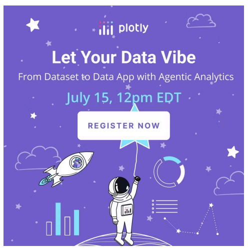
Alternatives to violin plots for visualizing distributions include histograms (https://plotly.com/python/histograms/), box plots (https://plotly.com/python/box-plots/), ECDF plots (https://plotly.com/python/ecdf-plots/) and strip charts (https://plotly.com/python/strip-charts/).

Basic Violin Plot 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/).

```
import plotly.express as px

df = px.data.tips()
fig = px.violin(df, y="total_bill")
fig.show()
```

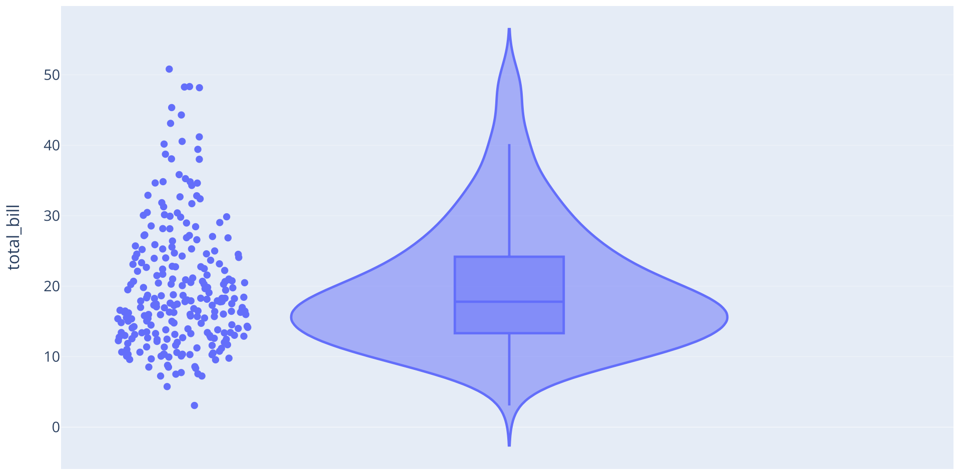




Violin plot with box and data points

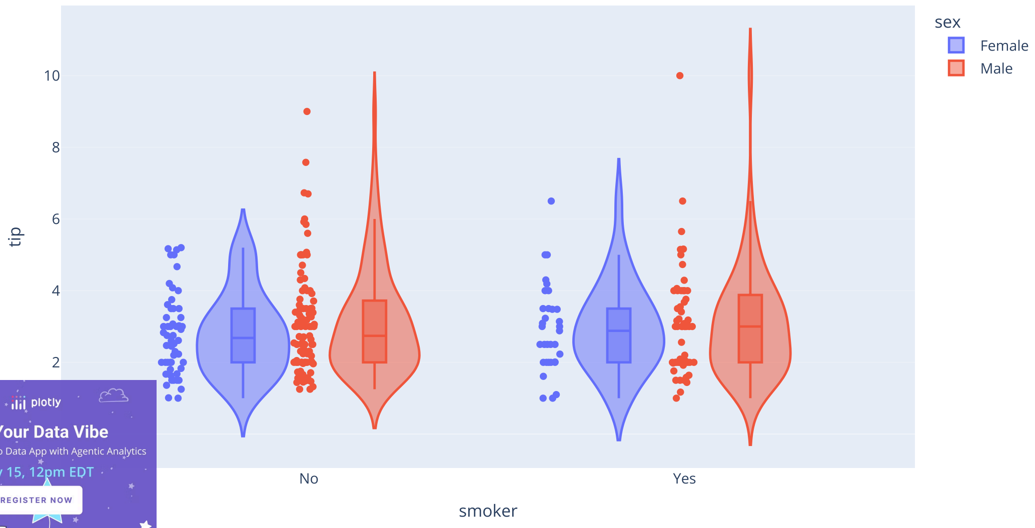
```
import plotly.express as px

df = px.data.tips()
fig = px.violin(df, y="total_bill", box=True, # draw box plot inside the violin
                points='all', # can be 'outliers', or False
               )
fig.show()
```

ting

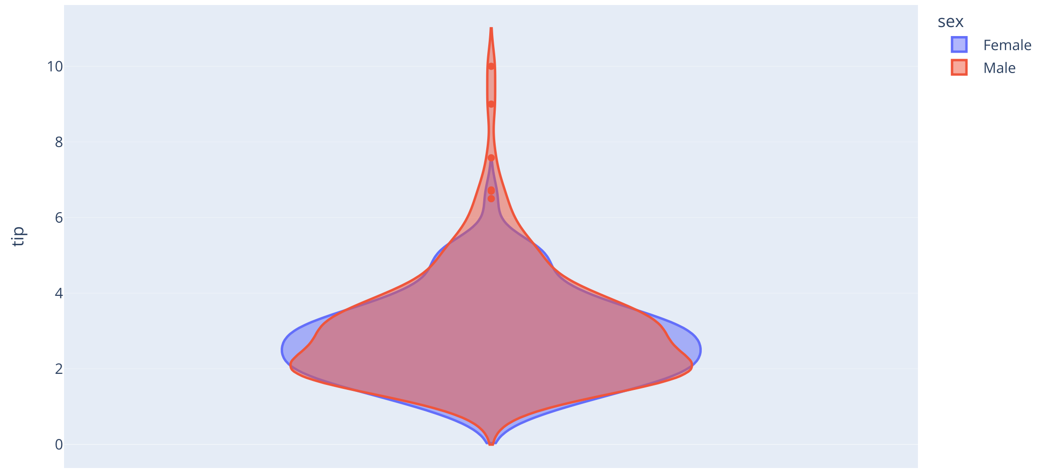


Multiple Violin Plots

```
import plotly.express as px

df = px.data.tips()
fig = px.violin(df, y="tip", x="smoker", color="sex", box=True, points="all",
          hover_data=df.columns)
fig.show()
```



```
import plotly.express as px

df = px.data.tips()
fig = px.violin(df, y="tip", color="sex",
                violinmode='overlay', # draw violins on top of each other
                # default violinmode is 'group' as in example above
                hover_data=df.columns)
fig.show()
```

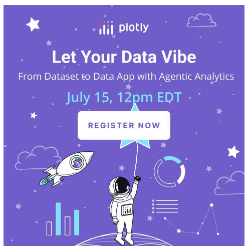


ting

Violin Plot with go.Violin

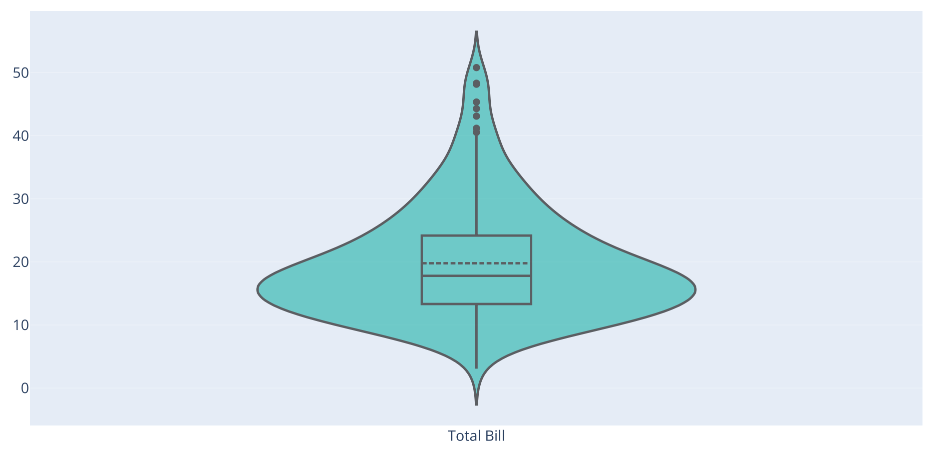
If Plotly Express does not provide a good starting point, you can use the more generic go.Violin class from plotly.graph_objects (/python/graph-objects/). All the options of go.Violin are documented in the reference https://plotly.com/python/reference/violin/ (https://plotly.com/python/reference/violin/)

Basic Violin Plot

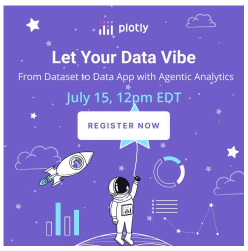


```
import plotly.graph_objects as go

import pandas as pd

df = pd.read_csv("https://raw.githubusercontent.com/plotly/datasets/master/violin_data.csv")

fig = go.Figure(data=go.Violin(y=df['total_bill'], box_visible=True, line_color='black',
                               meanline_visible=True, fillcolor='lightseagreen', opacity=0.6,
                               x0='Total Bill'))

fig.update_layout(yaxis_zeroline=False)
fig.show()
```

ting

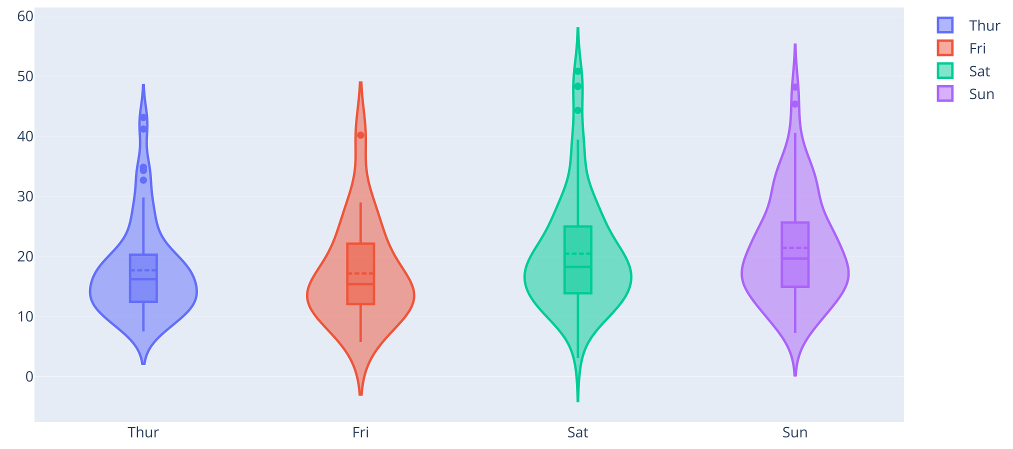


Multiple Traces



```
import plotly.graph_objects as go

import pandas as pd

df = pd.read_csv("https://raw.githubusercontent.com/plotly/datasets/master/violin_data.csv")

fig = go.Figure()

days = ['Thur', 'Fri', 'Sat', 'Sun']

for day in days:
    fig.add_trace(go.Violin(x=df['day'][df['day'] == day],
                            y=df['total_bill'][df['day'] == day],
                            name=day,
                            box_visible=True,
                            meanline_visible=True))

fig.show()
```

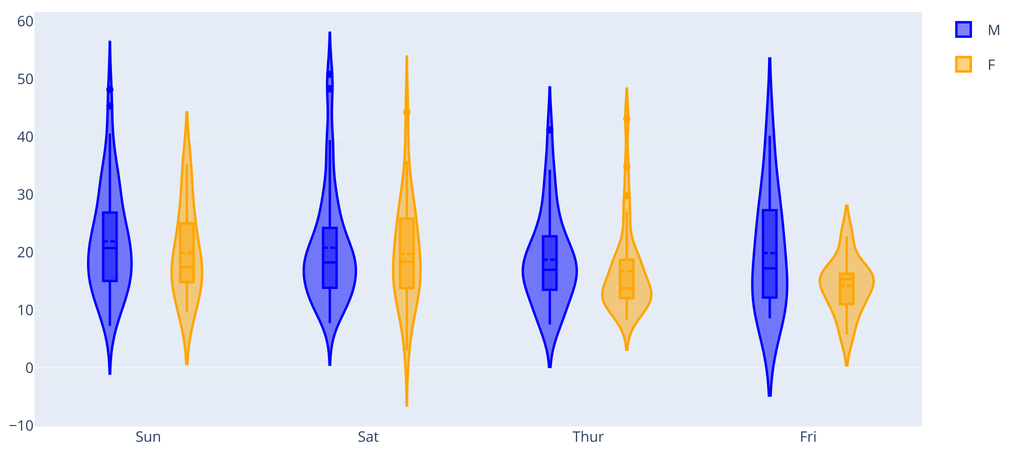
ting



Grouped Violin Plot

```
import plotly.graph_objects as go

import pandas as pd

df = pd.read_csv("https://raw.githubusercontent.com/plotly/datasets/master/violin_data.csv")

fig = go.Figure()

fig.add_trace(go.Violin(x=df['day'][ df['sex'] == 'Male' ],
                        y=df['total_bill'][ df['sex'] == 'Male' ],
                        legendgroup='M', scalegroup='M', name='M',
                        line_color='blue')
             )
fig.add_trace(go.Violin(x=df['day'][ df['sex'] == 'Female' ],
                        y=df['total_bill'][ df['sex'] == 'Female' ],
                        legendgroup='F', scalegroup='F', name='F',
                        line_color='orange')
             )

fig.update_traces(box_visible=True, meanline_visible=True)
fig.update_layout(violinmode='group')
fig.show()
```

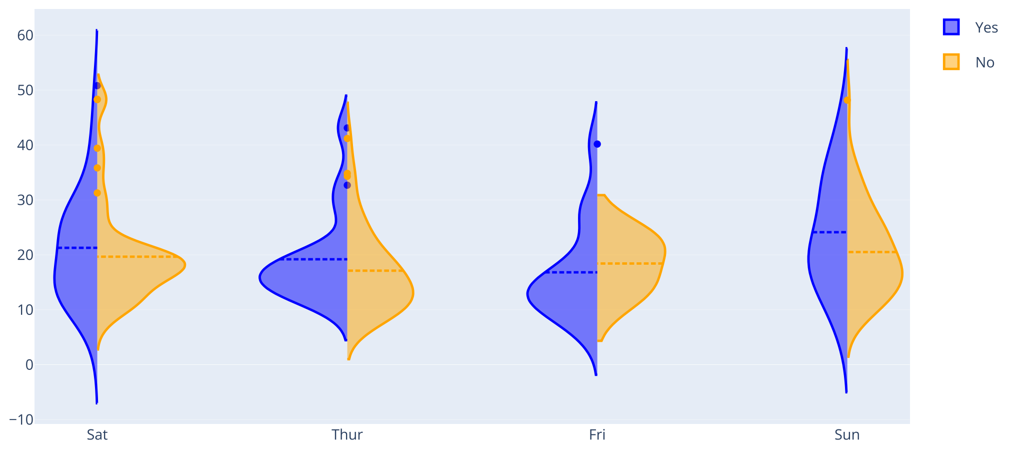
ting



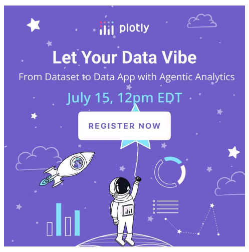
Split Violin Plot

```python
import plotly.graph_objects as go

import pandas as pd

df = pd.read_csv("https://raw.githubusercontent.com/plotly/datasets/master/violin_data.csv")

fig = go.Figure()

fig.add_trace(go.Violin(x=df['day'][ df['smoker'] == 'Yes' ],
                        y=df['total_bill'][ df['smoker'] == 'Yes' ],
                        legendgroup='Yes', scalegroup='Yes', name='Yes',
                        side='negative',
                        line_color='blue')
             )
fig.add_trace(go.Violin(x=df['day'][ df['smoker'] == 'No' ],
                        y=df['total_bill'][ df['smoker'] == 'No' ],
                        legendgroup='No', scalegroup='No', name='No',
                        side='positive',
                        line_color='orange')
             )
fig.update_traces(meanline_visible=True)
fig.update_layout(violingap=0, violinmode='overlay')
fig.show()
```

ting

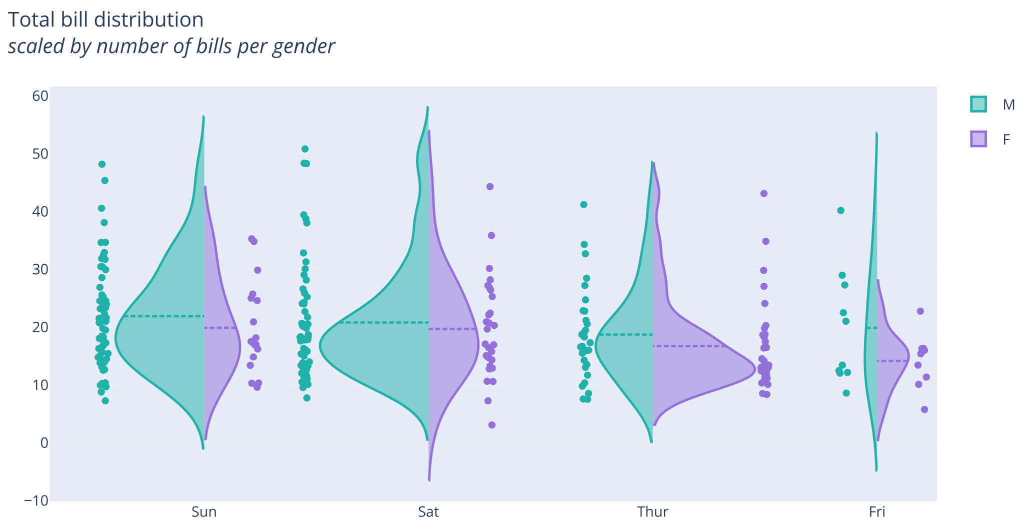


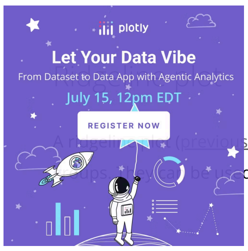
Advanced Violin Plot



```
import plotly.graph_objects as go

import pandas as pd

df = pd.read_csv("https://raw.githubusercontent.com/plotly/datasets/master/violin_data.csv")

pointpos_male = [-0.9,-1.1,-0.6,-0.3]
pointpos_female = [0.45,0.55,1,0.4]
show_legend = [True,False,False,False]

fig = go.Figure()

for i in range(0,len(pd.unique(df['day']))):
    fig.add_trace(go.Violin(x=df['day'][(df['sex'] == 'Male') &
                                        (df['day'] == pd.unique(df['day'])[i])],
                            y=df['total_bill'][(df['sex'] == 'Male')&
                                               (df['day'] == pd.unique(df['day'])[i])],
                            legendgroup='M', scalegroup='M', name='M',
                            side='negative',
                            pointpos=pointpos_male[i], # where to position points
                            line_color='lightseagreen',
                            showlegend=show_legend[i])
             )
    fig.add_trace(go.Violin(x=df['day'][(df['sex'] == 'Female') &
                                        (df['day'] == pd.unique(df['day'])[i])],
                            y=df['total_bill'][(df['sex'] == 'Female')&
                                               (df['day'] == pd.unique(df['day'])[i])],
                            legendgroup='F', scalegroup='F', name='F',
                            side='positive',
                            pointpos=pointpos_female[i],
                            line_color='mediumpurple',
                            showlegend=show_legend[i])
             )

# update characteristics shared by all traces
fig.update_traces(meanline_visible=True,
                  points='all', # show all points
                  jitter=0.05,  # add some jitter on points for better visibility
                  scalemode='count') #scale violin plot area with total count
fig.update_layout(
    title_text="Total bill distribution<br><i>scaled by number of bills per gender",
    violingap=0, violingroupgap=0, violinmode='overlay')
fig.show()
```

ting

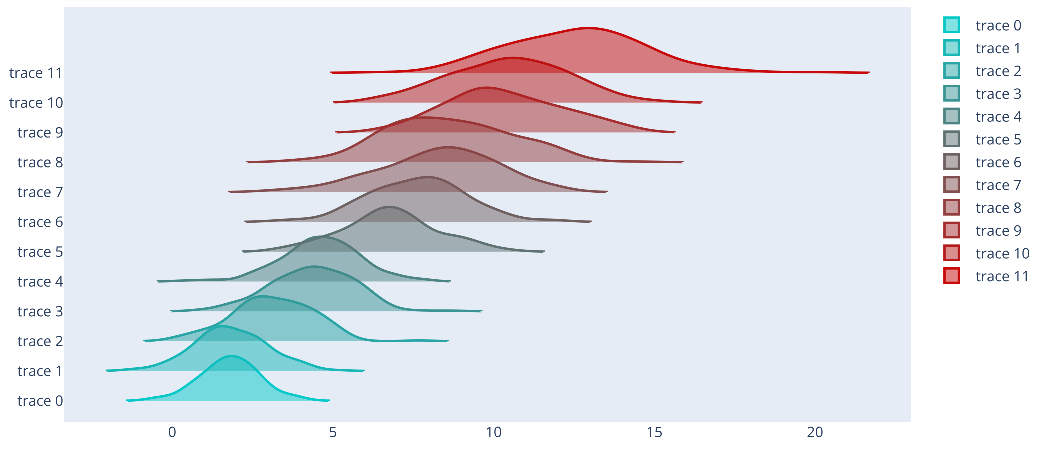




sly known as Joy Plot (https://serialmentor.com/blog/2017/9/15/goodbye-joyplots)) shows the distribution of a numerical value for several
d for visualizing changes in distributions over time or space.

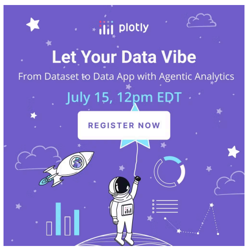
```python
import plotly.graph_objects as go
from plotly.colors import n_colors
import numpy as np
np.random.seed(1)

# 12 sets of normal distributed random data, with increasing mean and standard deviation
data = (np.linspace(1, 2, 12)[:, np.newaxis] * np.random.randn(12, 200) +
            (np.arange(12) + 2 * np.random.random(12))[:, np.newaxis])

colors = n_colors('rgb(5, 200, 200)', 'rgb(200, 10, 10)', 12, colortype='rgb')

fig = go.Figure()
for data_line, color in zip(data, colors):
    fig.add_trace(go.Violin(x=data_line, line_color=color))

fig.update_traces(orientation='h', side='positive', width=3, points=False)
fig.update_layout(xaxis_showgrid=False, xaxis_zeroline=False)
fig.show()
```

ting

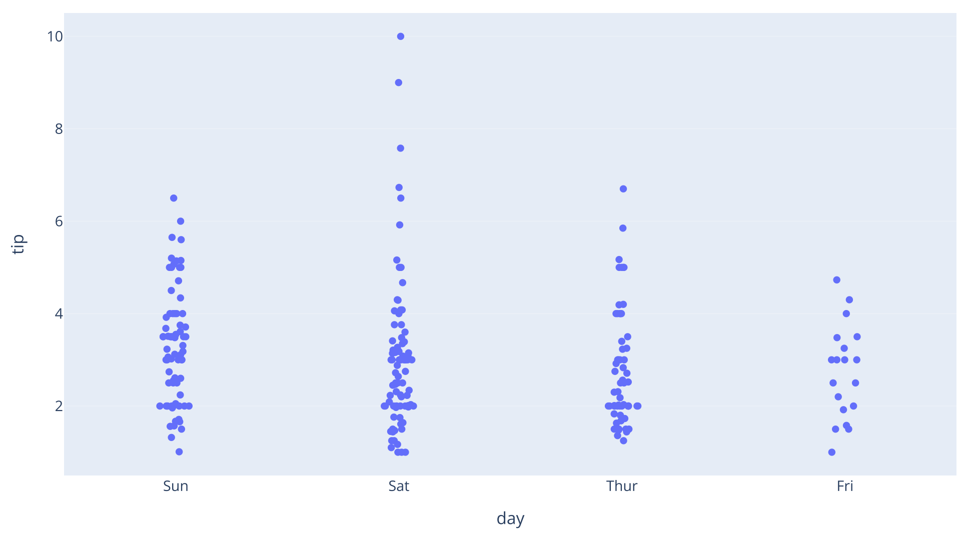


Violin Plot With Only Points

A strip chart (/python/strip-charts/) is like a violin plot with points showing, and no violin:



```
import plotly.express as px
df = px.data.tips()
fig = px.strip(df, x='day', y='tip')
fig.show()
```



ting

Choosing The Algorithm For Computing Quartiles

New in 5.10

By default, quartiles for violin plots are computed using the linear method (for more about linear interpolation, see #10 listed on http://jse.amstat.org/v14n3/langford.html (http://jse.amstat.org/v14n3/langford.html) and https://en.wikipedia.org/wiki/Quartile (https://en.wikipedia.org/wiki/Quartile) for more details).

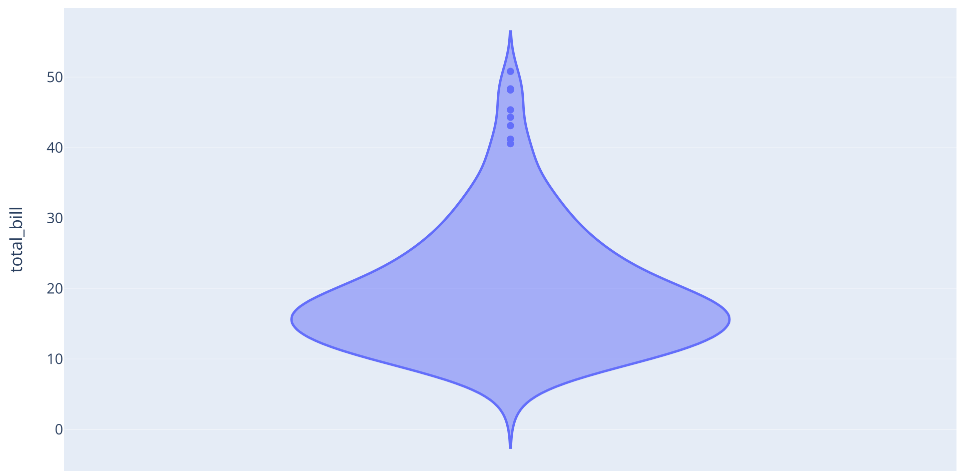
However, you can also choose to use an exclusive or an inclusive algorithm to compute quartiles.

The *exclusive* algorithm uses the median to divide the ordered dataset into two halves. If the sample is odd, it does not include the median in either half. Q1 is then the median of the lower half and Q3 is the median of the upper half.

The *inclusive* algorithm also uses the median to divide the ordered dataset into two halves, but if the sample is odd, it includes the median in both halves. Q1 is then the median of the lower half and Q3 the median of the upper half.

```
import plotly.express as px

df = px.data.tips()
fig = px.violin(df, y="total_bill")
fig.update_traces(quartilemethod="exclusive") # or "inclusive", or "linear" by default

fig.show()
```

ting



Reference

See function reference for px.violin() (https://plotly.com/python-api-reference/generated/plotly.express.violin) or https://plotly.com/python/reference/violin/ (https://plotly.com/python/reference/violin/) for more information and chart attribute options!

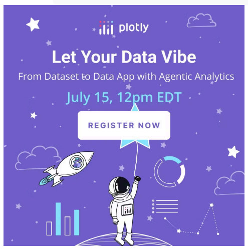
What About Dash?

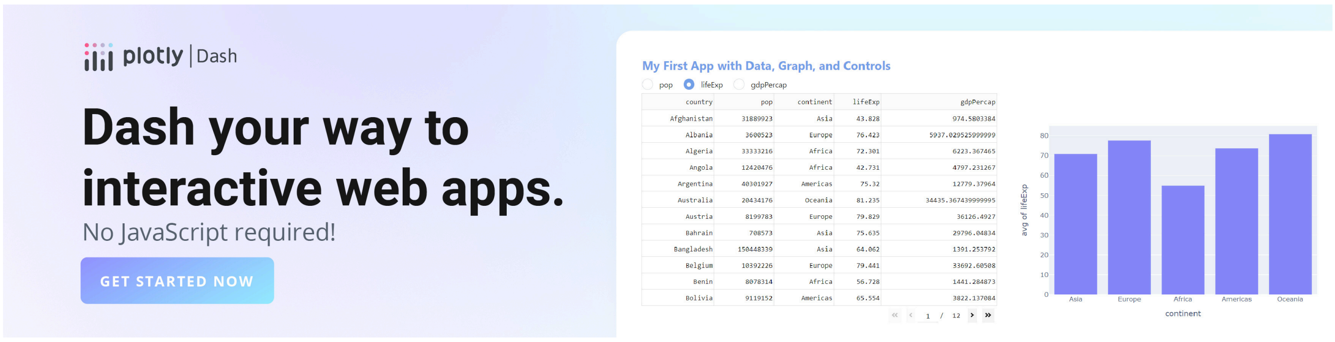
Dash (https://dash.plot.ly/) is an open-source framework for building analytical applications, with no Javascript required, and it is tightly integrated with the Plotly graphing library.

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
```





My First App with Data, Graph, and Controls

○ pop ● lifeExp ○ gdpPercap

country	pop	continent	lifeExp	gdpPercap
Afghanistan	31889923	Asia	43.828	974.5803384
Albania	3600523	Europe	76.423	5937.029525999999
Algeria	33333216	Africa	72.301	6223.367465
Angola	12420476	Africa	42.731	4797.231267
Argentina	40301927	Americas	75.32	12779.37964
Australia	20434176	Oceania	81.235	34435.367439999995
Austria	8199783	Europe	79.829	36126.4927
Bahrain	708573	Asia	75.635	29796.04834
Bangladesh	150448339	Asia	64.062	1391.253792
Belgium	10392226	Europe	79.441	33692.60508
Benin	8078314	Africa	56.728	1441.284873
Bolivia	9119152	Americas	65.554	3822.137084

« ‹ 1 / 12 › »

(https://dash.plotly.com/tutorial?utm_medium=graphing_libraries&utm_content=python_footer)