plotly | Graphing Libraries (https://plotly.com/)(/graphing-libraries/)

utm_campaign=studio_cloud_launch&utm_content=sidebar)

Python (/python) > Statistical Charts (/python/statistical-charts) > Box Plots Suggest an edit to this page (https://github.com/plotly/plotly.py/edit/doc-prod/doc/python/box-plots.md)

Box Plots in Python

How to make Box Plots in Python with Plotly.

ting

hms

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)

A box plot (https://en.wikipedia.org/wiki/Box_plot) is a statistical representation of the distribution of a variable through its quartiles. The ends of the box represent the lower and upper quartiles, while the median (second quartile) is marked by a line inside the box. For other statistical representations of numerical data, see other statistical charts (https://plotly.com/python/statistical-charts/).

ng Data

ıting

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

es

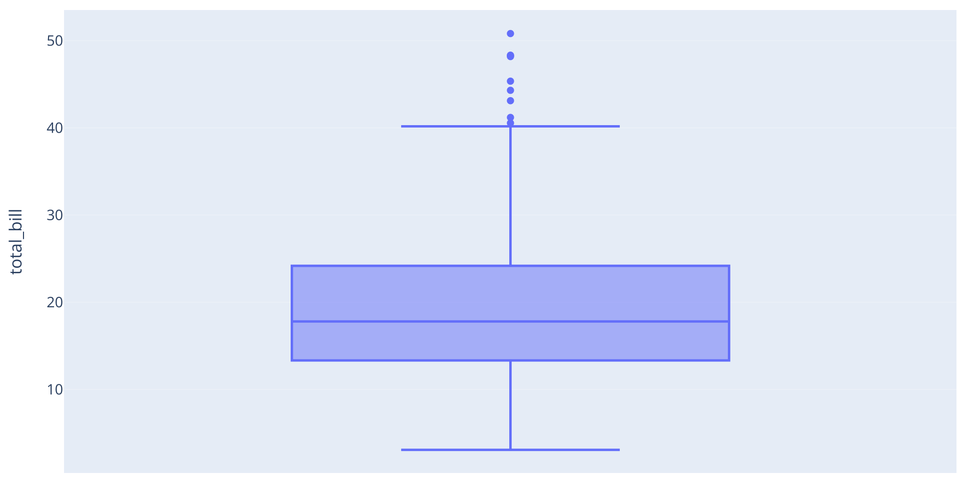
Box Plot with plotly.express

iation

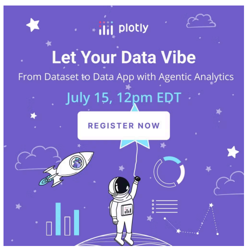
Plotly Express (/python/plotly-express/) is the easy-to-use, high-level interface to Plotly, which operates on a variety of types of data (/python/px-arguments/) and produces easy-to-style figures (/python/styling-plotly-express/).

In a box plot created by px.box, the distribution of the column given as y argument is represented.

```
import plotly.express as px
df = px.data.tips()
fig = px.box(df, y="total_bill")
fig.show()
```



If a column name is given as x argument, a box plot is drawn for each value of x.



```
import plotly.express as px
df = px.data.tips()
fig = px.box(df, x="time", y="total_bill")
fig.show()
```

ting

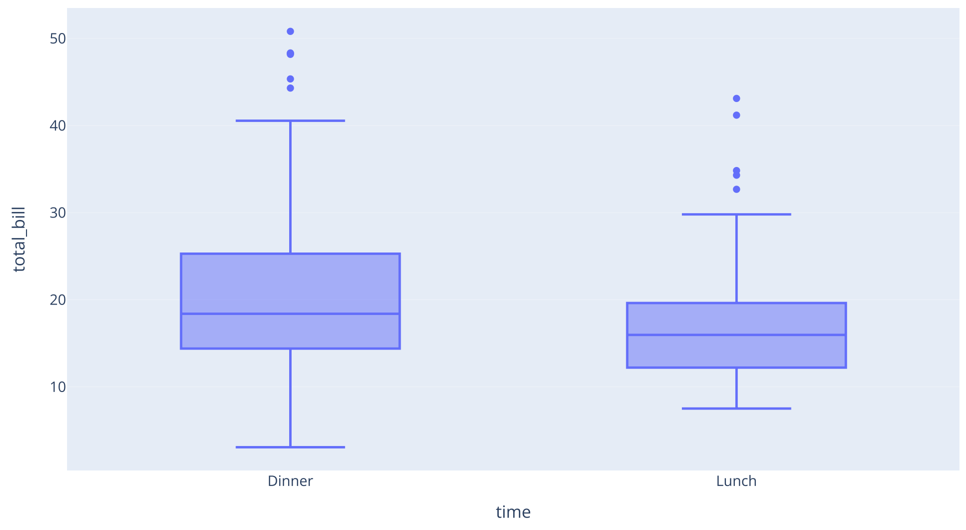
hms

ng Data

ıting

es

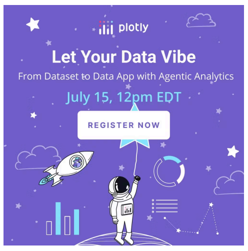
iation



Box Plots 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
import plotly.express as px

app = Dash(__name__)


app.layout = html.Div([
    html.H4("Analysis of the restaurant's revenue"),
    html.P("x-axis:"),
    dcc.Checklist(
        id='x-axis',
        options=['smoker', 'day', 'time', 'sex'],
        value=['time'],
        inline=True
    ),
    html.P("y-axis:"),
    dcc.RadioItems(
        id='y-axis',
        options=['total_bill', 'tip', 'size'],
        value='total_bill',
        inline=True
    ),
    dcc.Graph(id="graph"),
])
```

DOWNLOAD

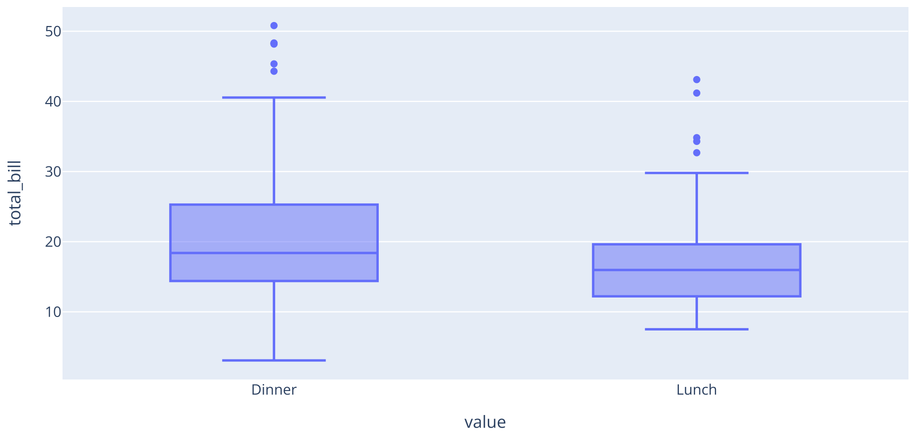
Analysis of the restaurant's revenue

x-axis:

smoker day [✓ time] sex

y-axis:

[✓ total_bill] tip size



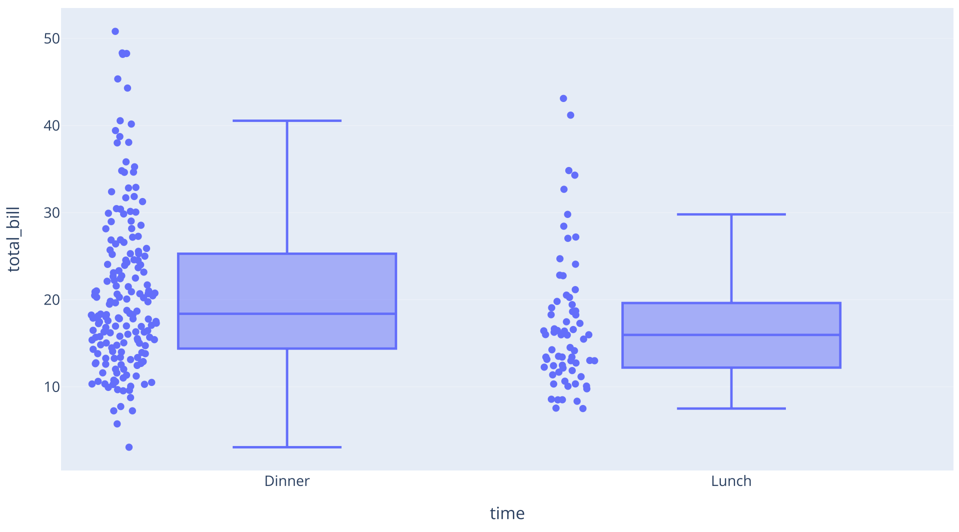
Sign up for Dash Club → Free cheat sheets plus updates from Chris Parmer and Adam Schroeder delivered to your inbox every two months. Includes tips and tricks, community apps, and deep dives into the Dash architecture. Join now (https://go.plotly.com/dash-club?utm_source=Dash+Club+2022&utm_medium=graphing_libraries&utm_content=inline).

Display the underlying data

With the points argument, display underlying data points with either all points (all), outliers only (outliers, default), or none of them (False).

```
import plotly.express as px
df = px.data.tips()
fig = px.box(df, x="time", y="total_bill", points="all")
fig.show()
```

ting
hms
ng Data
ıting
es
iation



Choosing The Algorithm For Computing Quartiles

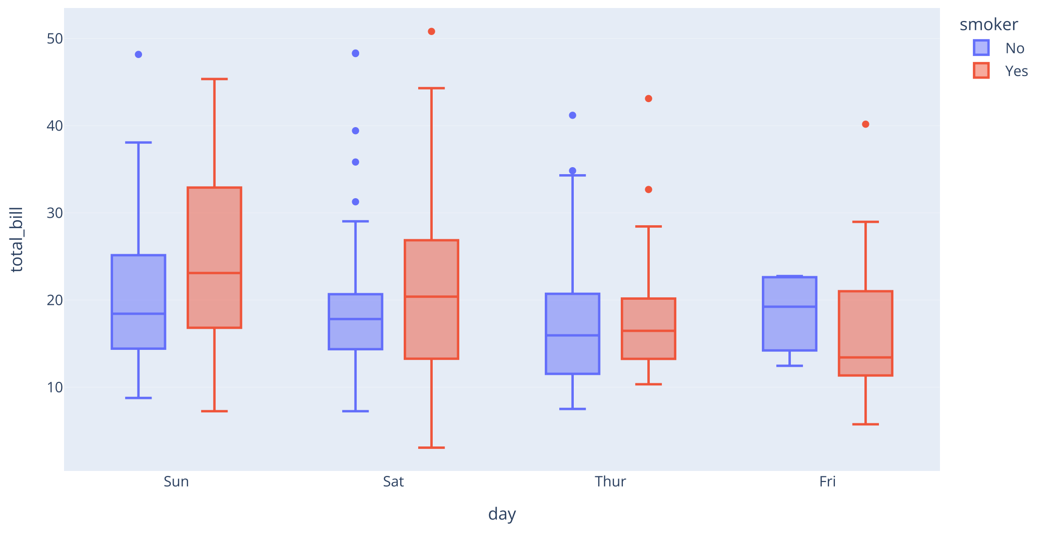
By default, quartiles for box plots are computed using the linear method (for more about linear interpolation, see #10 listed on http://jse.amstat.org/v14n3/langford.html (http://jse.amstat.org/v14n3/langford.html) and https://en.wikipedia.org/wiki/Quartile (https://en.wikipedia.org/wiki/Quartile) for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

```python
import plotly.express as px

df = px.data.tips()

fig = px.box(df, x="day", y="total_bill", color="smoker")
fig.update_traces(quartilemethod="exclusive") # or "inclusive", or "linear" by default
fig.show()
```

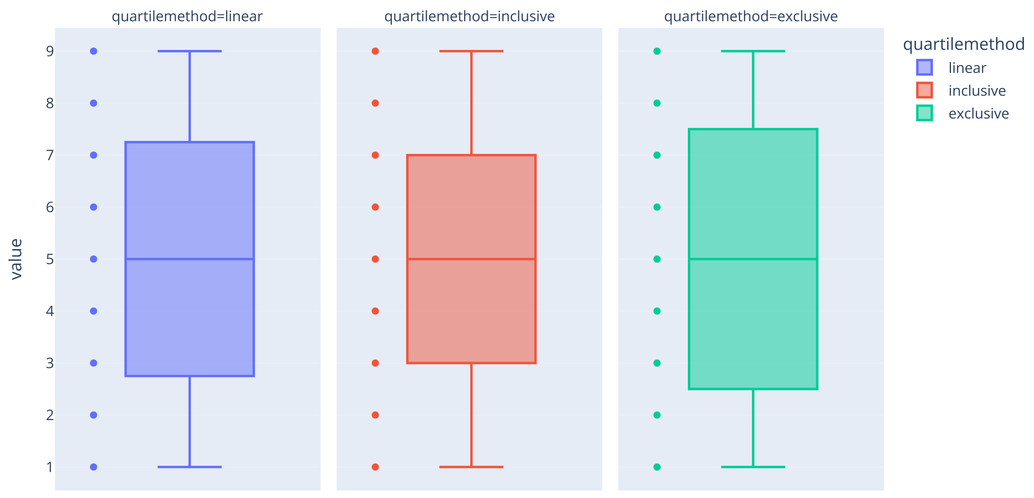


Difference Between Quartile Algorithms

It can sometimes be difficult to see the difference between the linear, inclusive, and exclusive algorithms for computing quartiles. In the following example, the same dataset is visualized using each of the three different quartile computation algorithms.

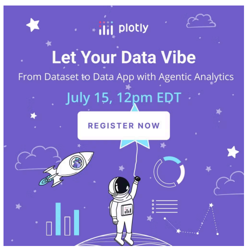
```
import plotly.express as px
import pandas as pd

data = [1,2,3,4,5,6,7,8,9]
df = pd.DataFrame(dict(
    linear=data,
    inclusive=data,
    exclusive=data
)).melt(var_name="quartilemethod")


fig = px.box(df, y="value", facet_col="quartilemethod", color="quartilemethod",
             boxmode="overlay", points='all')

fig.update_traces(quartilemethod="linear", jitter=0, col=1)
fig.update_traces(quartilemethod="inclusive", jitter=0, col=2)
fig.update_traces(quartilemethod="exclusive", jitter=0, col=3)

fig.show()
```

ting
hms
ng Data
ıting
es
iation



Styled box plot

For the interpretation of the notches, see https://en.wikipedia.org/wiki/Box_plot#Variations (https://en.wikipedia.org/wiki/Box_plot#Variations).



```python
import plotly.express as px
df = px.data.tips()
fig = px.box(df, x="time", y="total_bill", color="smoker",
             notched=True, # used notched shape
             title="Box plot of total bill",
             hover_data=["day"] # add day column to hover data
            )
fig.show()
```

ting

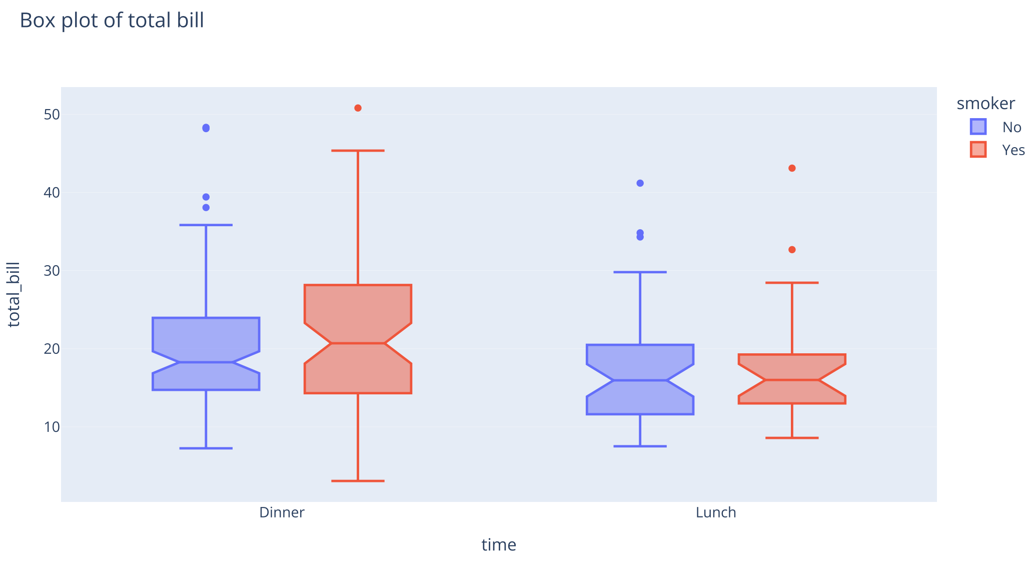
hms

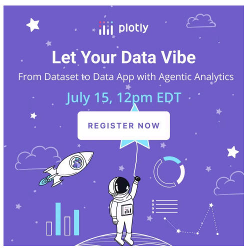
ng Data

ıting

es

iation





Box plot with go.Box

If Plotly Express does not provide a good starting point, it is also possible to use the more generic go.Box class from plotly.graph_objects (/python/graph-objects/). All available options for go.Box are described in the reference page https://plotly.com/python/reference/box/ (https://plotly.com/python/reference/box/).

Basic Box Plot

ting

hms

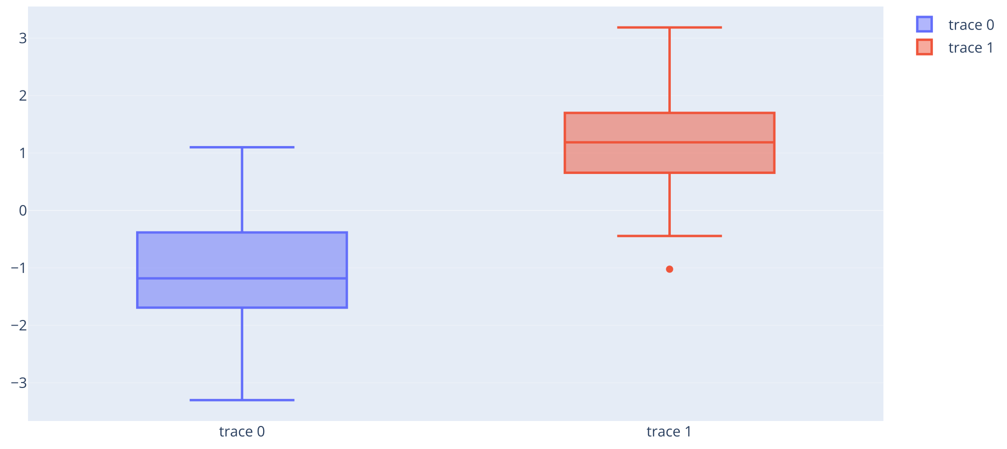
ng Data

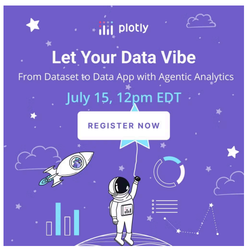
ıting

ɜs

ʲiation

```
import plotly.graph_objects as go
import numpy as np
np.random.seed(1)

y0 = np.random.randn(50) - 1
y1 = np.random.randn(50) + 1

fig = go.Figure()
fig.add_trace(go.Box(y=y0))
fig.add_trace(go.Box(y=y1))

fig.show()
```





Basic Horizontal Box Plot

```
import plotly.graph_objects as go
import numpy as np

x0 = np.random.randn(50)
x1 = np.random.randn(50) + 2 # shift mean

fig = go.Figure()
# Use x instead of y argument for horizontal plot
fig.add_trace(go.Box(x=x0))
fig.add_trace(go.Box(x=x1))

fig.show()
```

ting

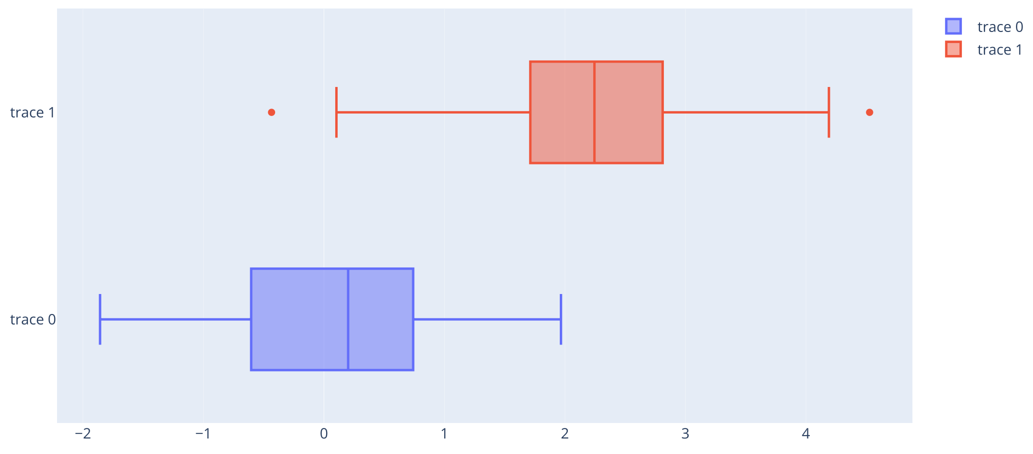
hms

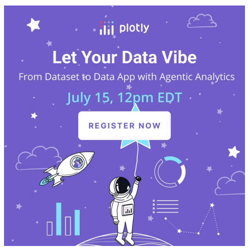
ng Data

ıting

es

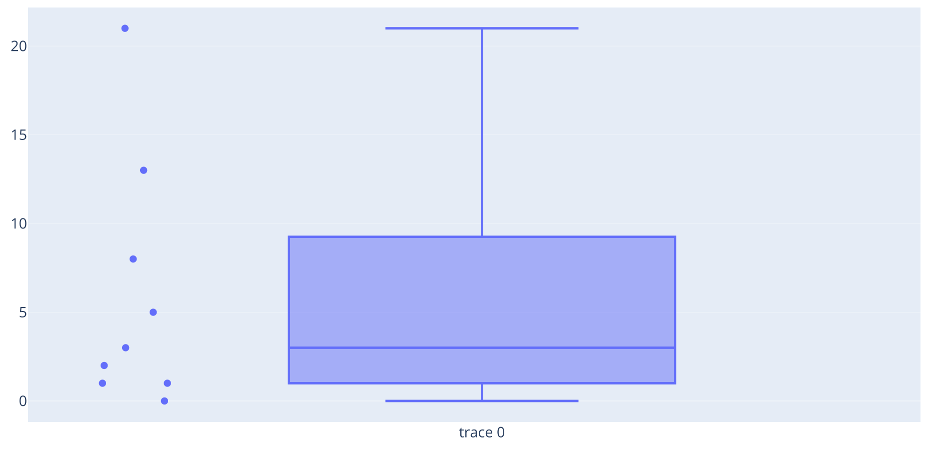
iation





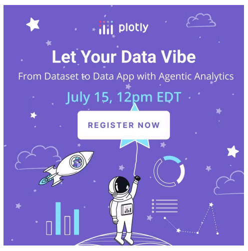
Box Plot That Displays The Underlying Data

```
import plotly.graph_objects as go

fig = go.Figure(data=[go.Box(y=[0, 1, 1, 2, 3, 5, 8, 13, 21],
            boxpoints='all', # can also be outliers, or suspectedoutliers, or False
            jitter=0.3, # add some jitter for a better separation between points
            pointpos=-1.8 # relative position of points wrt box
              )])

fig.show()
```

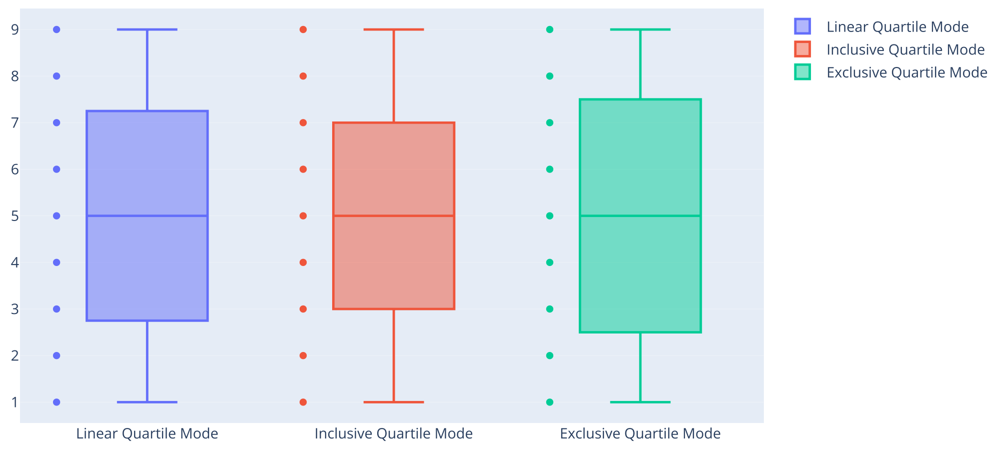


Modifying The Algorithm For Computing Quartiles

For an explanation of how each algorithm works, see Choosing The Algorithm For Computing Quartiles.



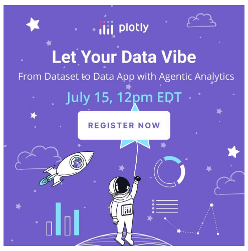
```
import plotly.graph_objects as go

data = [1, 2, 3, 4, 5, 6, 7, 8, 9]

fig = go.Figure()
fig.add_trace(go.Box(y=data, quartilemethod="linear", name="Linear Quartile Mode"))
fig.add_trace(go.Box(y=data, quartilemethod="inclusive", name="Inclusive Quartile Mode"))
fig.add_trace(go.Box(y=data, quartilemethod="exclusive", name="Exclusive Quartile Mode"))
fig.update_traces(boxpoints='all', jitter=0)
fig.show()
```



Box Plot With Precomputed Quartiles

You can specify precomputed quartile attributes rather than using a built-in quartile computation algorithm.

This could be useful if you have already pre-computed those values or if you need to use a different algorithm than the ones provided.



```
import plotly.graph_objects as go

fig = go.Figure()

fig.add_trace(go.Box(q1=[ 1, 2, 3 ], median=[ 4, 5, 6 ],
                  q3=[ 7, 8, 9 ], lowerfence=[-1, 0, 1],
                  upperfence=[7, 8, 9], mean=[ 2.2, 2.8, 3.2 ],
                  sd=[ 0.2, 0.4, 0.6 ], notchspan=[ 0.2, 0.4, 0.6 ], name="Precompiled Quartiles"))

fig.show()
```

ting

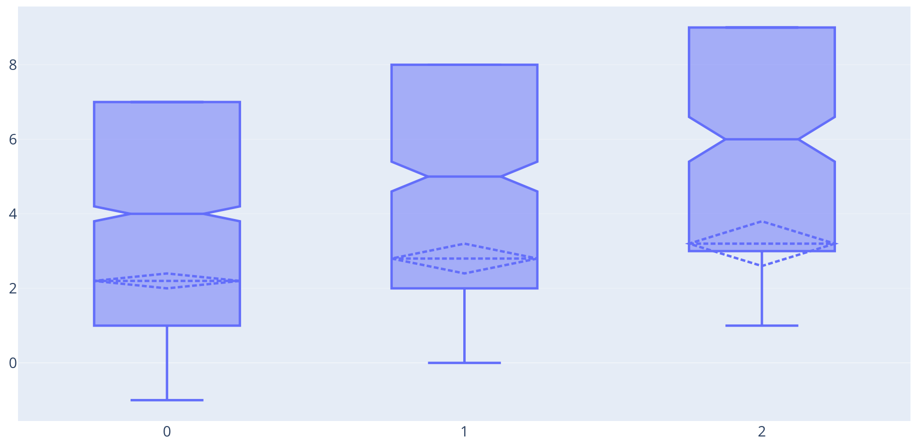
hms

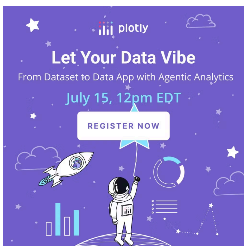
ng Data

ıting

es

iation





Colored Box Plot

```
import plotly.graph_objects as go
import numpy as np

y0 = np.random.randn(50)
y1 = np.random.randn(50) + 1 # shift mean

fig = go.Figure()
fig.add_trace(go.Box(y=y0, name='Sample A',
                marker_color = 'indianred'))
fig.add_trace(go.Box(y=y1, name = 'Sample B',
                marker_color = 'lightseagreen'))

fig.show()
```

ting

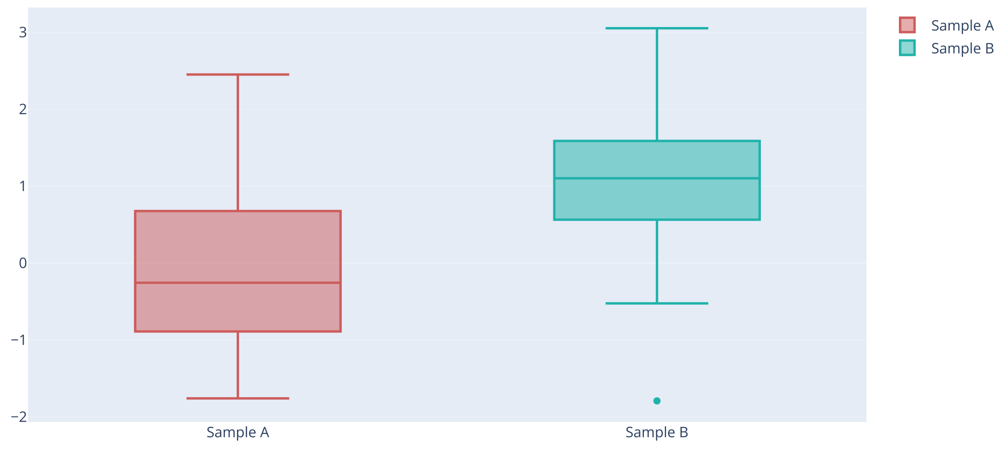
hms

ng Data

ıting

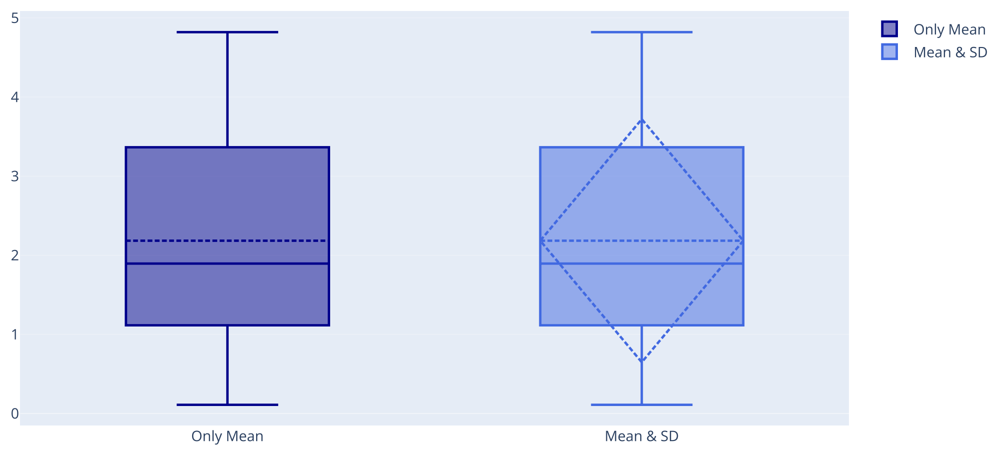
es

iation



Box Plot Styling Mean & Standard Deviation

```
import plotly.graph_objects as go

fig = go.Figure()
fig.add_trace(go.Box(
    y=[2.37, 2.16, 4.82, 1.73, 1.04, 0.23, 1.32, 2.91, 0.11, 4.51, 0.51, 3.75, 1.35, 2.98, 4.50, 0.18, 4.66, 1.30, 2.06, 1.19],
    name='Only Mean',
    marker_color='darkblue',
    boxmean=True # represent mean
))
fig.add_trace(go.Box(
    y=[2.37, 2.16, 4.82, 1.73, 1.04, 0.23, 1.32, 2.91, 0.11, 4.51, 0.51, 3.75, 1.35, 2.98, 4.50, 0.18, 4.66, 1.30, 2.06, 1.19],
    name='Mean & SD',
    marker_color='royalblue',
    boxmean='sd' # represent mean and standard deviation
))

fig.show()
```



Styling Outliers

The example below shows how to use the boxpoints argument. If "outliers", only the sample points lying outside the whiskers are shown. If "suspectedoutliers", the outlier points are shown and points either less than 4Q1-3Q3 or greater than 4Q3-3Q1 are highlighted (using outliercolor). If "all", all sample points are shown. If False, only the boxes are shown with no sample points.

```python
import plotly.graph_objects as go

fig = go.Figure()
fig.add_trace(go.Box(
    y=[0.75, 5.25, 5.5, 6, 6.2, 6.6, 6.80, 7.0, 7.2, 7.5, 7.5, 7.75, 8.15,
       8.15, 8.65, 8.93, 9.2, 9.5, 10, 10.25, 11.5, 12, 16, 20.90, 22.3, 23.25],
    name="All Points",
    jitter=0.3,
    pointpos=-1.8,
    boxpoints='all', # represent all points
    marker_color='rgb(7,40,89)',
    line_color='rgb(7,40,89)'
))

fig.add_trace(go.Box(
    y=[0.75, 5.25, 5.5, 6, 6.2, 6.6, 6.80, 7.0, 7.2, 7.5, 7.5, 7.75, 8.15,
        8.15, 8.65, 8.93, 9.2, 9.5, 10, 10.25, 11.5, 12, 16, 20.90, 22.3, 23.25],
    name="Only Whiskers",
    boxpoints=False, # no data points
    marker_color='rgb(9,56,125)',
    line_color='rgb(9,56,125)'
))

fig.add_trace(go.Box(
    y=[0.75, 5.25, 5.5, 6, 6.2, 6.6, 6.80, 7.0, 7.2, 7.5, 7.5, 7.75, 8.15,
        8.15, 8.65, 8.93, 9.2, 9.5, 10, 10.25, 11.5, 12, 16, 20.90, 22.3, 23.25],
    name="Suspected Outliers",
    boxpoints='suspectedoutliers', # only suspected outliers
    marker=dict(
        color='rgb(8,81,156)',
        outliercolor='rgba(219, 64, 82, 0.6)',
        line=dict(
            outliercolor='rgba(219, 64, 82, 0.6)',
            outlierwidth=2)),
    line_color='rgb(8,81,156)'
))

fig.add_trace(go.Box(
    y=[0.75, 5.25, 5.5, 6, 6.2, 6.6, 6.80, 7.0, 7.2, 7.5, 7.5, 7.75, 8.15,
        8.15, 8.65, 8.93, 9.2, 9.5, 10, 10.25, 11.5, 12, 16, 20.90, 22.3, 23.25],
    name="Whiskers and Outliers",
    boxpoints='outliers', # only outliers
    marker_color='rgb(107,174,214)',
    line_color='rgb(107,174,214)'
))


fig.update_layout(title_text="Box Plot Styling Outliers")
fig.show()
```

ting

hms

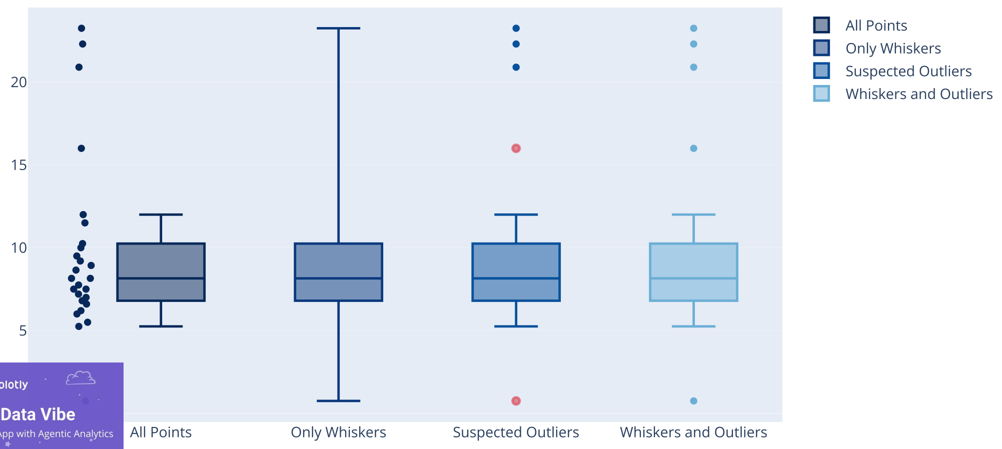
ng Data

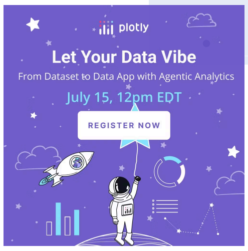
ıting

es

iation

Box Plot Styling Outliers





ting

hms

ng Data

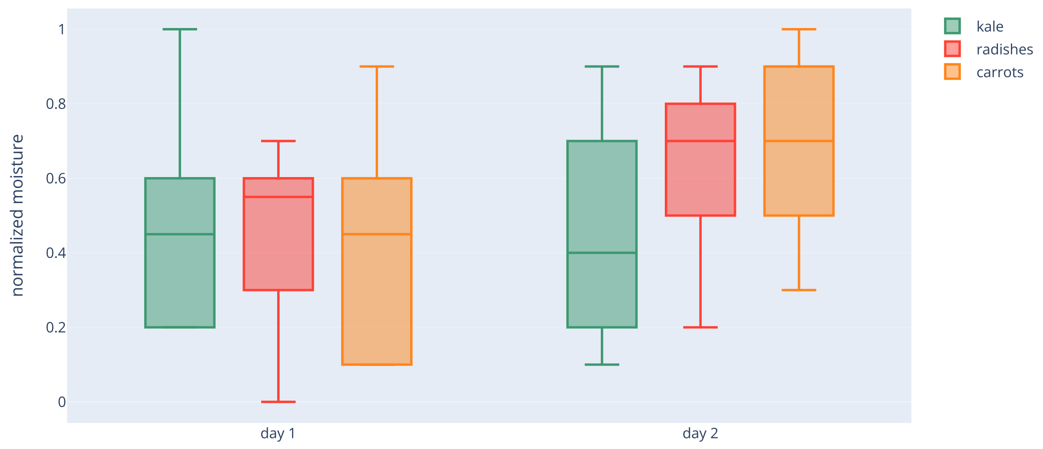
ıting

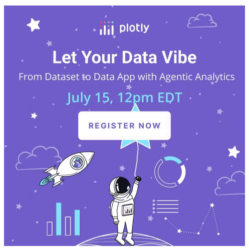
ɘs

ˈiation

Grouped Box Plots

```
import plotly.graph_objects as go

x = ['day 1', 'day 1', 'day 1', 'day 1', 'day 1', 'day 1',
     'day 2', 'day 2', 'day 2', 'day 2', 'day 2', 'day 2']

fig = go.Figure()

fig.add_trace(go.Box(
    y=[0.2, 0.2, 0.6, 1.0, 0.5, 0.4, 0.2, 0.7, 0.9, 0.1, 0.5, 0.3],
    x=x,
    name='kale',
    marker_color='#3D9970'
))
fig.add_trace(go.Box(
    y=[0.6, 0.7, 0.3, 0.6, 0.0, 0.5, 0.7, 0.9, 0.5, 0.8, 0.7, 0.2],
    x=x,
    name='radishes',
    marker_color='#FF4136'
))
fig.add_trace(go.Box(
    y=[0.1, 0.3, 0.1, 0.9, 0.6, 0.6, 0.9, 1.0, 0.3, 0.6, 0.8, 0.5],
    x=x,
    name='carrots',
    marker_color='#FF851B'
))

fig.update_layout(
    yaxis=dict(
        title=dict(
            text='normalized moisture')
    ),
    boxmode='group' # group together boxes of the different traces for each value of x
)
fig.show()
```





Grouped Horizontal Box Plot

ting

hms

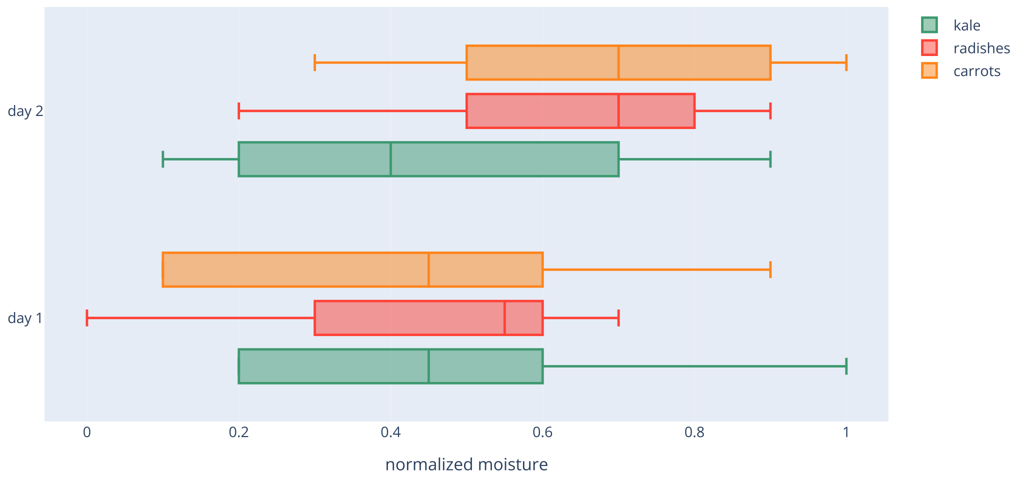
ng Data

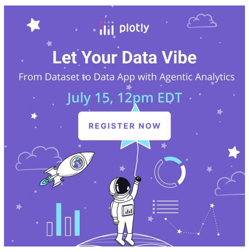
ıting

əs

ʻiation

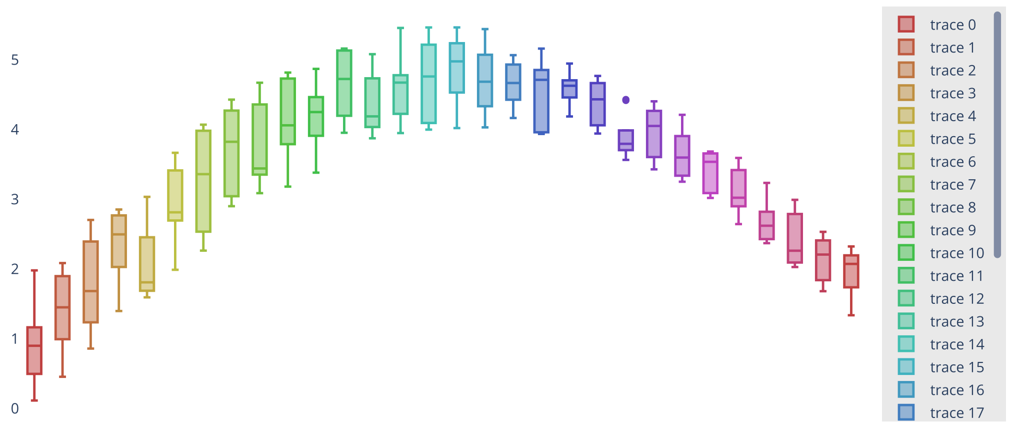
```
import plotly.graph_objects as go

y = ['day 1', 'day 1', 'day 1', 'day 1', 'day 1', 'day 1',
     'day 2', 'day 2', 'day 2', 'day 2', 'day 2', 'day 2']

fig = go.Figure()
fig.add_trace(go.Box(
    x=[0.2, 0.2, 0.6, 1.0, 0.5, 0.4, 0.2, 0.7, 0.9, 0.1, 0.5, 0.3],
    y=y,
    name='kale',
    marker_color='#3D9970'
))
fig.add_trace(go.Box(
    x=[0.6, 0.7, 0.3, 0.6, 0.0, 0.5, 0.7, 0.9, 0.5, 0.8, 0.7, 0.2],
    y=y,
    name='radishes',
    marker_color='#FF4136'
))
fig.add_trace(go.Box(
    x=[0.1, 0.3, 0.1, 0.9, 0.6, 0.6, 0.9, 1.0, 0.3, 0.6, 0.8, 0.5],
    y=y,
    name='carrots',
    marker_color='#FF851B'
))

fig.update_layout(
    xaxis=dict(title=dict(text='normalized moisture'), zeroline=False),
    boxmode='group'
)

fig.update_traces(orientation='h') # horizontal box plots
fig.show()
```





Rainbow Box Plots

```
import plotly.graph_objects as go
import numpy as np

N = 30     # Number of boxes

# generate an array of rainbow colors by fixing the saturation and lightness of the HSL
# representation of colour and marching around the hue.
# Plotly accepts any CSS color format, see e.g. http://www.w3schools.com/cssref/css_colors_legal.asp.
c = ['hsl('+str(h)+',50%'+',50%)' for h in np.linspace(0, 360, N)]

# Each box is represented by a dict that contains the data, the type, and the colour.
# Use list comprehension to describe N boxes, each with a different colour and with different randomly generated data:
fig = go.Figure(data=[go.Box(
    y=3.5 * np.sin(np.pi * i/N) + i/N + (1.5 + 0.5 * np.cos(np.pi*i/N)) * np.random.rand(10),
    marker_color=c[i]
    ) for i in range(int(N))])

# format the layout
fig.update_layout(
    xaxis=dict(showgrid=False, zeroline=False, showticklabels=False),
    yaxis=dict(zeroline=False, gridcolor='white'),
    paper_bgcolor='rgb(233,233,233)',
    plot_bgcolor='rgb(233,233,233)',
)

fig.show()
```



Fully Styled Box Plots

ting

hms

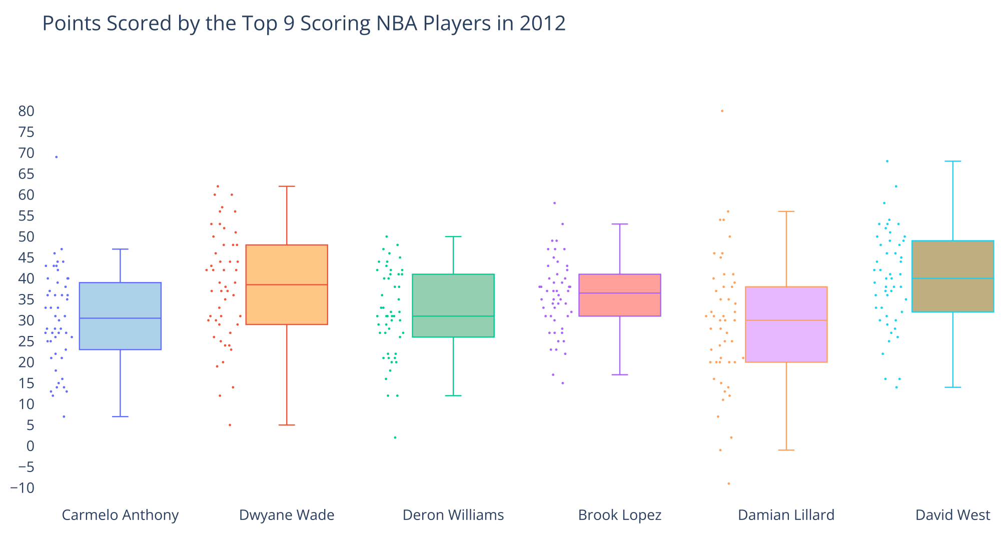
ng Data

uting

es

iation

```
import plotly.graph_objects as go

x_data = ['Carmelo Anthony', 'Dwyane Wade',
          'Deron Williams', 'Brook Lopez',
          'Damian Lillard', 'David West',]

N = 50

y0 = (10 * np.random.randn(N) + 30).astype(int)
y1 = (13 * np.random.randn(N) + 38).astype(int)
y2 = (11 * np.random.randn(N) + 33).astype(int)
y3 = (9 * np.random.randn(N) + 36).astype(int)
y4 = (15 * np.random.randn(N) + 31).astype(int)
y5 = (12 * np.random.randn(N) + 40).astype(int)

y_data = [y0, y1, y2, y3, y4, y5]

colors = ['rgba(93, 164, 214, 0.5)', 'rgba(255, 144, 14, 0.5)', 'rgba(44, 160, 101, 0.5)',
          'rgba(255, 65, 54, 0.5)', 'rgba(207, 114, 255, 0.5)', 'rgba(127, 96, 0, 0.5)']

fig = go.Figure()

for xd, yd, cls in zip(x_data, y_data, colors):
        fig.add_trace(go.Box(
            y=yd,
            name=xd,
            boxpoints='all',
            jitter=0.5,
            whiskerwidth=0.2,
            fillcolor=cls,
            marker_size=2,
            line_width=1)
        )

fig.update_layout(
    title=dict(text='Points Scored by the Top 9 Scoring NBA Players in 2012'),
    yaxis=dict(
        autorange=True,
        showgrid=True,
        zeroline=True,
        dtick=5,
        gridcolor='rgb(255, 255, 255)',
        gridwidth=1,
        zerolinecolor='rgb(255, 255, 255)',
        zerolinewidth=2,
    ),
    margin=dict(
        l=40,
        r=30,
        b=80,
        t=100,
    ),
    paper_bgcolor='rgb(243, 243, 243)',
    plot_bgcolor='rgb(243, 243, 243)',
    showlegend=False
)

fig.show()
```



ting

hms

ng Data

ıting

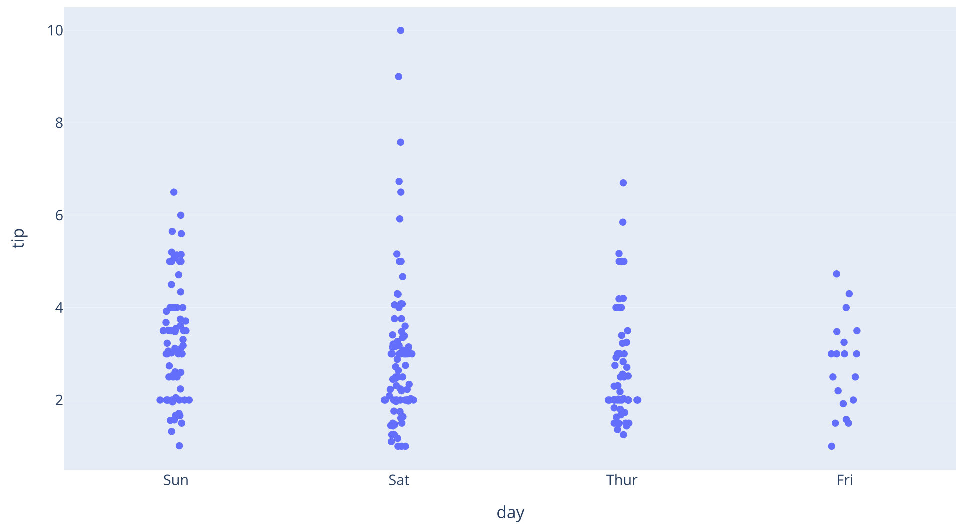
ɜs

ʻiation

Box Plot With Only Points

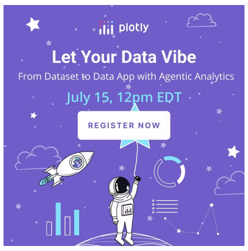
A strip chart (/python/strip-charts/) is like a box plot with points showing, and no box:

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



Reference

See function reference for px.box() (https://plotly.com/python-api-reference/generated/plotly.express.box) or https://plotly.com/python/reference/box/ (https://plotly.com/python/reference/box/) 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:

ting

hms

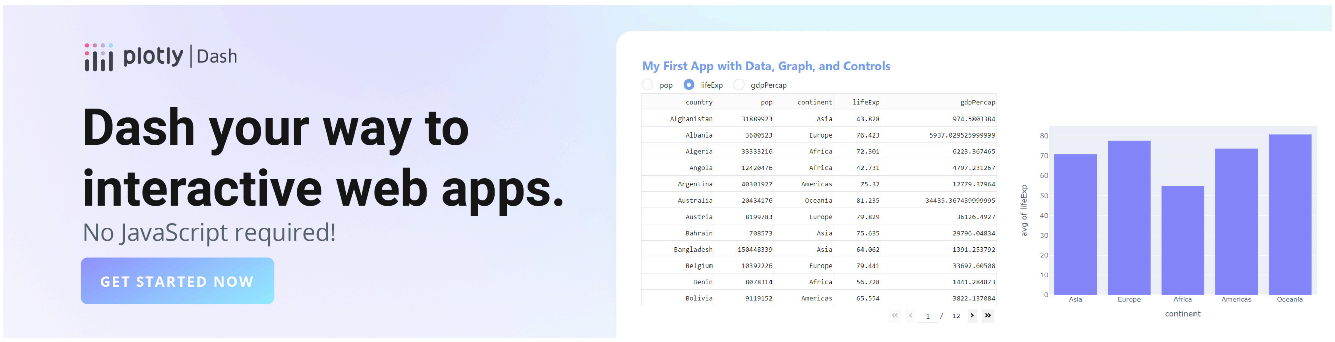
ng Data

ıting

es

iation

```
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
```



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

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

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/)