

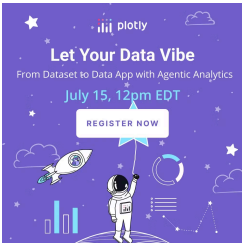
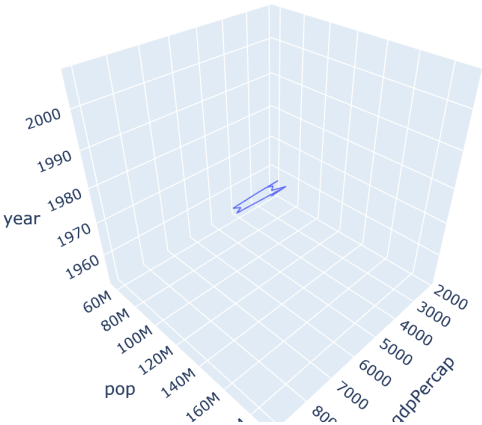
3D Line Plots in Python

How to make 3D Line Plots

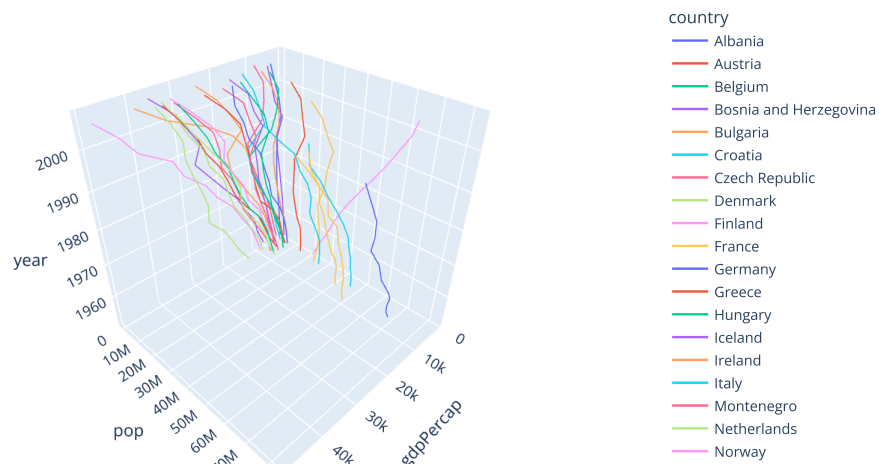
Plotly Studio: Transform any dataset into an interactive data application in minutes with AI. [Sign up for early access now.](https://plotly.com/studio/?utm_medium=graphing-libraries&utm_campaign=studio_early_access&utm_content=sidebar) (https://plotly.com/studio/?utm_medium=graphing-libraries&utm_campaign=studio_early_access&utm_content=sidebar)

3D Line plot with Plotly Express

```
import plotly.express as px
df = px.data.gapminder().query("country=='Brazil'")
fig = px.line_3d(df, x="gdpPercap", y="pop", z="year")
fig.show()
```

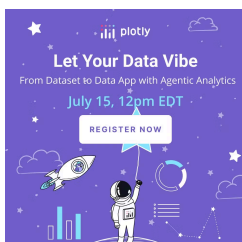


```
import plotly.express as px
df = px.data.gapminder().query("continent=='Europe'")
fig = px.line_3d(df, x="gdpPercap", y="pop", z="year", color='country')
fig.show()
```



3D Line Plot of Brownian Motion

Here we represent a trajectory in 3D.



```

import plotly.graph_objects as go
import pandas as pd
import numpy as np

rs = np.random.RandomState()
rs.seed(0)

def brownian_motion(T = 1, N = 100, mu = 0.1, sigma = 0.01, S0 = 20):
    dt = float(T)/N
    t = np.linspace(0, T, N)
    W = rs.standard_normal(size = N)
    W = np.cumsum(W)*np.sqrt(dt) # standard brownian motion
    X = (mu-0.5*sigma**2)*t + sigma*W
    S = S0*np.exp(X) # geometric brownian motion
    return S

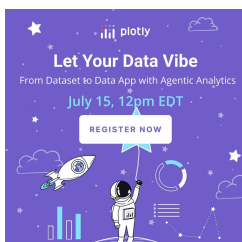
dates = pd.date_range('2012-01-01', '2013-02-22')
T = (dates.max()-dates.min()).days / 365
N = dates.size
start_price = 100
y = brownian_motion(T, N, sigma=0.1, S0=start_price)
z = brownian_motion(T, N, sigma=0.1, S0=start_price)

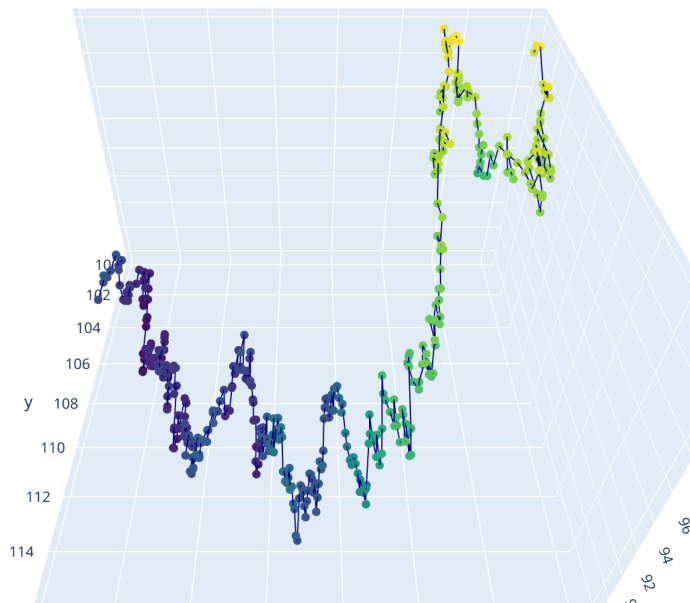
fig = go.Figure(data=go.Scatter3d(
    x=dates, y=y, z=z,
    marker=dict(
        size=4,
        color=z,
        colorscale='Viridis',
    ),
    line=dict(
        color='darkblue',
        width=2
    )
))

fig.update_layout(
    width=800,
    height=700,
    autosize=False,
    scene=dict(
        camera=dict(
            up=dict(
                x=0,
                y=0,
                z=1
            ),
            eye=dict(
                x=0,
                y=1.0707,
                z=1,
            )
        ),
        aspectratio = dict( x=1, y=1, z=0.7 ),
        aspectmode = 'manual'
    ),
)

fig.show()

```





Reference

See [function reference for px.line_3d](https://plotly.com/python-api-reference/generated/plotly.express.line_3d) (https://plotly.com/python-api-reference/generated/plotly.express.line_3d) or <https://plotly.com/python/reference/scatter3d/#scatter3d-marker-line> (<https://plotly.com/python/reference/scatter3d/#scatter3d-marker-line>) for more information and chart attribute options!

What About Dash?

[Dash](https://dash.plot.ly/) (<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>).

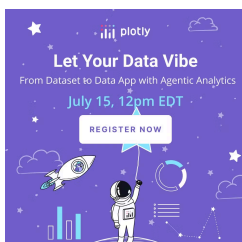
Everywhere in this page that you see `fig.show()`, you can display the same figure in a Dash application by passing it to the `figure` argument of the [Graph component](https://dash.plot.ly/dash-core-components/graph) (<https://dash.plot.ly/dash-core-components/graph>) from the built-in `dash_core_components` 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
```





Dash your way to interactive web apps.

No JavaScript required!

GET STARTED NOW

My First App with Data, Graph, and Controls


pop

lifeExp

gdpPerCap

| country | pop | continent | lifeExp | gdpPerCap |
|-------------|-----------|-----------|---------|--------------------|
| Afghanistan | 31889923 | Asia | 43.828 | 974.5883384 |
| Albania | 2600522 | Europe | 76.422 | 5937.625225999999 |
| Algeria | 33333216 | Africa | 72.361 | 6223.367465 |
| Angola | 12420676 | Africa | 42.731 | 4707.231267 |
| 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 | 1301.253792 |
| Belgium | 10392226 | Europe | 79.441 | 33692.04908 |
| Benin | 8078314 | Africa | 56.728 | 1441.284873 |
| Bolivia | 9119152 | Americas | 65.554 | 3822.137884 |

1 / 12



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)

About Us

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

Products

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

Pricing

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

Support

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

