Continuous Error Bands in Python

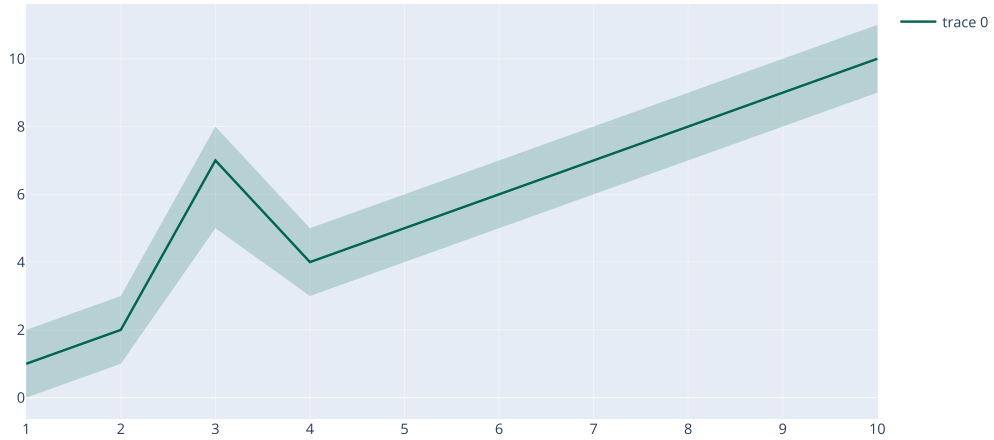
Add continuous error bands to charts in Python with Plotly.

Continuous error bands are a graphical representation of error or uncertainty as a shaded region around a main trace, rather than as discrete whisker-like error bars. They can be implemented in a manner similar to filled area plots (/python/filled-area-plots/) using `scatter` traces with the `fill` attribute.

Filling within a single trace

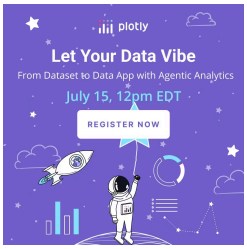
In this example we show how to construct a trace that goes from low to high X values along the upper Y edge of a region, and then from high to low X values along the lower Y edge of the region. This trace is then 'self-filled' using fill='toself'.

```
import plotly.graph_objs as go

x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
y = [1, 2, 7, 4, 5, 6, 7, 8, 9, 10]
y_upper = [2, 3, 8, 5, 6, 7, 8, 9, 10, 11]
y_lower = [0, 1, 5, 3, 4, 5, 6, 7, 8, 9]


fig = go.Figure([
    go.Scatter(
        x=x,
        y=y,
        line=dict(color='rgb(0,100,80)'),
        mode='lines'
    ),
    go.Scatter(
        x=x+x[::-1], # x, then x reversed
        y=y_upper+y_lower[::-1], # upper, then lower reversed
        fill='toself',
        fillcolor='rgba(0,100,80,0.2)',
        line=dict(color='rgba(255,255,255,0)'),
        hoverinfo="skip",
        showlegend=False
    )
])
fig.show()
```

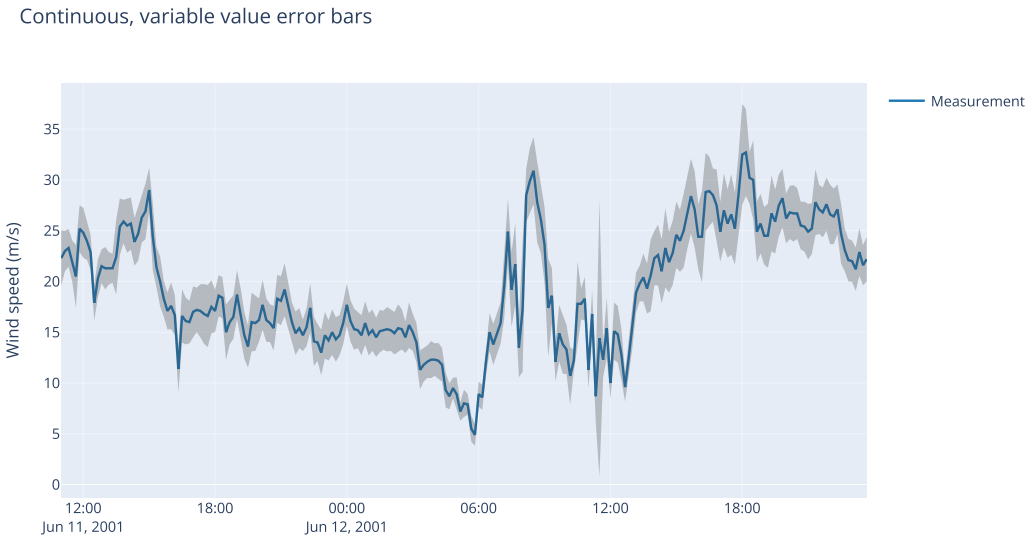


Filling between two traces

In this example we show how to construct the bounds of the band using two traces, with the lower trace using fill='tonexty' to fill an area up to the upper trace.



```
import plotly.graph_objs as go
import pandas as pd

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

fig = go.Figure([
    go.Scatter(
        name='Measurement',
        x=df['Time'],
        y=df['10 Min Sampled Avg'],
        mode='lines',
        line=dict(color='rgb(31, 119, 180)'),
    ),
    go.Scatter(
        name='Upper Bound',
        x=df['Time'],
        y=df['10 Min Sampled Avg']+df['10 Min Std Dev'],
        mode='lines',
        marker=dict(color="#444"),
        line=dict(width=0),
        showlegend=False
    ),
    go.Scatter(
        name='Lower Bound',
        x=df['Time'],
        y=df['10 Min Sampled Avg']-df['10 Min Std Dev'],
        marker=dict(color="#444"),
        line=dict(width=0),
        mode='lines',
        fillcolor='rgba(68, 68, 68, 0.3)',
        fill='tonexty',
        showlegend=False
    )
])
fig.update_layout(
    yaxis=dict(title=dict(text='Wind speed (m/s)')),
    title=dict(text='Continuous, variable value error bars'),
    hovermode="x"
)
fig.show()
```



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
```