***Python (/python) > (/python/) > Smoothing***

Suggest an edit to this page(https://github.com/plotly/plotly.py/edit/doc-prod/doc/python/smoothing.md)

# Smoothing in Python

Learn how to perform smoothing using various methods in Python.

## Imports

The tutorial below imports NumPy (http://www.numpy.org/), Pandas (https://pandas.pydata.org/docs/user_guide/10min.html), SciPy (https://www.scipy.org/) and Plotly (https://plotly.com/python/getting-started/).

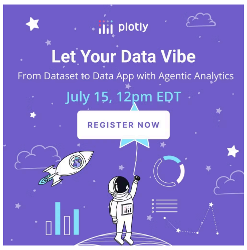
```
import plotly.graph_objects as go

import numpy as np
import pandas as pd
import scipy

from scipy import signal
```

## Savitzky-Golay Filter

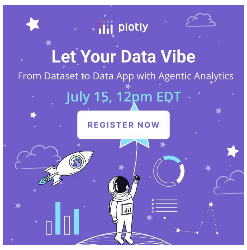
Smoothing is a technique that is used to eliminate noise from a dataset. There are many algorithms and methods to accomplish this but all have the same general purpose of 'roughing out the edges' or 'smoothing' some data.

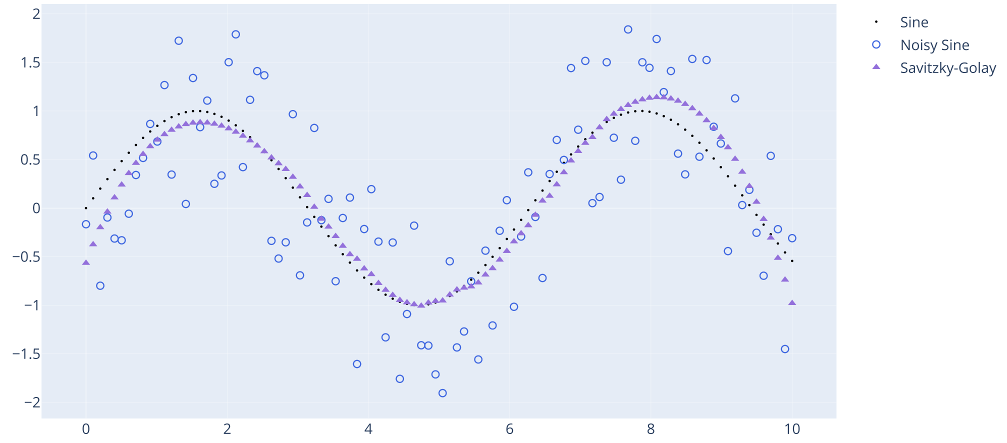
There is reason to smooth data if there is little to no small-scale structure in the data. The danger to this thinking is that one may skew the representation of the data enough to change its perceived meaning, so for the sake of scientific honesty it is an imperative to at the very minimum explain one's reason's for using a smoothing algorithm to their dataset.

In this example we use the Savitzky-Golay Filter (https://en.wikipedia.org/wiki/Savitzky%E2%80%93Golay_filter), which fits subsequent windows of adjacent data with a low-order polynomial.



```
import plotly.graph_objects as go

import numpy as np
import pandas as pd
import scipy

from scipy import signal

np.random.seed(1)

x = np.linspace(0, 10, 100)
y = np.sin(x)
noise = 2 * np.random.random(len(x)) - 1 # uniformly distributed between -1 and 1
y_noise = y + noise

fig = go.Figure()
fig.add_trace(go.Scatter(
    x=x,
    y=y,
    mode='markers',
    marker=dict(size=2, color='black'),
    name='Sine'
))

fig.add_trace(go.Scatter(
    x=x,
    y=y_noise,
    mode='markers',
    marker=dict(
        size=6,
        color='royalblue',
        symbol='circle-open'
    ),
    name='Noisy Sine'
))

fig.add_trace(go.Scatter(
    x=x,
    y=signal.savgol_filter(y_noise,
                           53, # window size used for filtering
                           3), # order of fitted polynomial
    mode='markers',
    marker=dict(
        size=6,
        color='mediumpurple',
        symbol='triangle-up'
    ),
    name='Savitzky-Golay'
))


fig.show()
```





## Triangular Moving Average

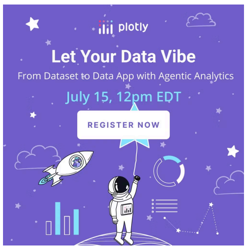
Another method for smoothing is a moving average. There are various forms of this, but the idea is to take a window of points in your dataset, compute an average of the points, then shift the window over by one point and repeat. This will generate a bunch of points which will result in the smoothed data.

Let us look at the common Simple Moving Average first. In the 1D case we have a data set of $N$ points with y-values $y_1, y_2, ..., y_N$. Setting our window size to $n < N$, the new $i^{th}$ y-value after smoothing is computed as:

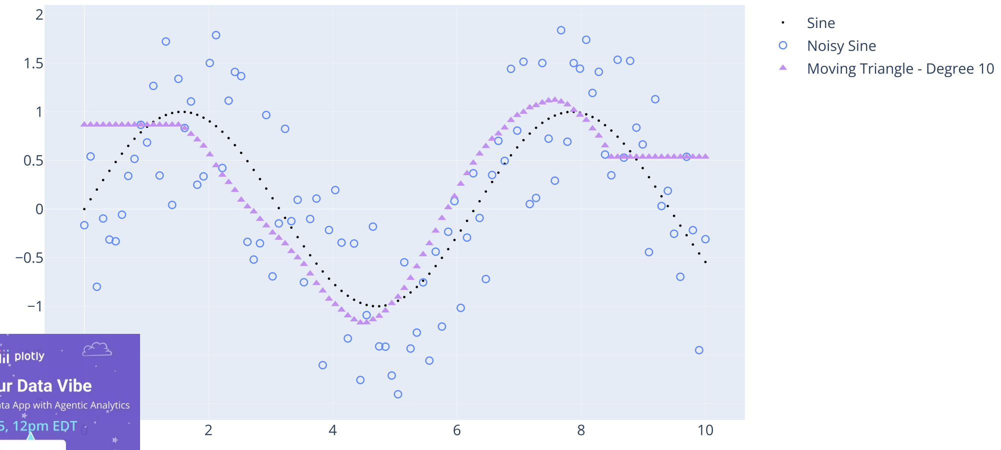
$$SMA_i = \frac{y_i + \ldots + y_{i+n}}{n}$$

In the Triangular Moving Average, two simple moving averages are computed on top of each other, in order to give more weight to closer (adjacent) points. This means that our $SMA_i$ are computed then a Triangular Moving Average $TMA_i$ is computed as:

$$TMA_i = \frac{SMA_i + \ldots + SMA_{i+n}}{n}$$



```
def smoothTriangle(data, degree):
    triangle=np.concatenate((np.arange(degree + 1), np.arange(degree)[::-1])) # up then down
    smoothed=[]

    for i in range(degree, len(data) - degree * 2):
        point=data[i:i + len(triangle)] * triangle
        smoothed.append(np.sum(point)/np.sum(triangle))
    # Handle boundaries
    smoothed=[smoothed[0]]*int(degree + degree/2) + smoothed
    while len(smoothed) < len(data):
        smoothed.append(smoothed[-1])
    return smoothed

fig = go.Figure()
fig.add_trace(go.Scatter(
    x=x,
    y=y,
    mode='markers',
    marker=dict(
        size=2,
        color='rgb(0, 0, 0)',
    ),
    name='Sine'
))

fig.add_trace(go.Scatter(
    x=x,
    y=y_noise,
    mode='markers',
    marker=dict(
        size=6,
        color='#5E88FC',
        symbol='circle-open'
    ),
    name='Noisy Sine'
))

fig.add_trace(go.Scatter(
    x=x,
    y=smoothTriangle(y_noise, 10),  # setting degree to 10
    mode='markers',
    marker=dict(
        size=6,
        color='#C190F0',
        symbol='triangle-up'
    ),
    name='Moving Triangle - Degree 10'
))

fig.show()
```



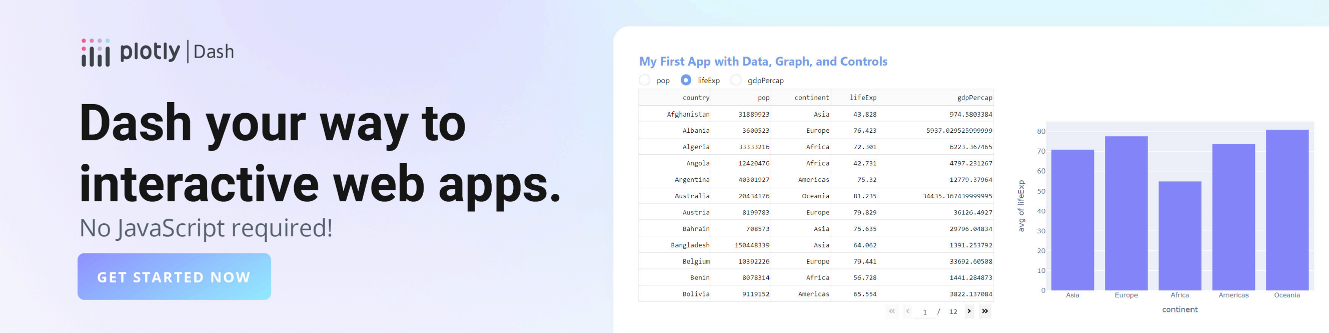
## What About Dash?

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
```



**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/)