

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

utm_campaign=studio_cloud_launch&utm_content=sidebar)



Python (/python) > Statistical Charts (/python/statistical-charts) > Linear and Non-Linear Trendlines

Suggest an edit to this (<https://github.com/plotly/plotly.py/edit/doc-prod/doc/python/linear-fits.md>)

Linear and Non-Linear Trendlines in Python

Add linear Ordinary Least Squares (OLS) regression trendlines or non-linear Locally Weighted Scatterplot Smoothing (LOWESS) trendlines to scatterplots in Python. Options for moving averages (rolling means) as well as exponentially-weighted and expanding functions.

Multiple

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)

Linking

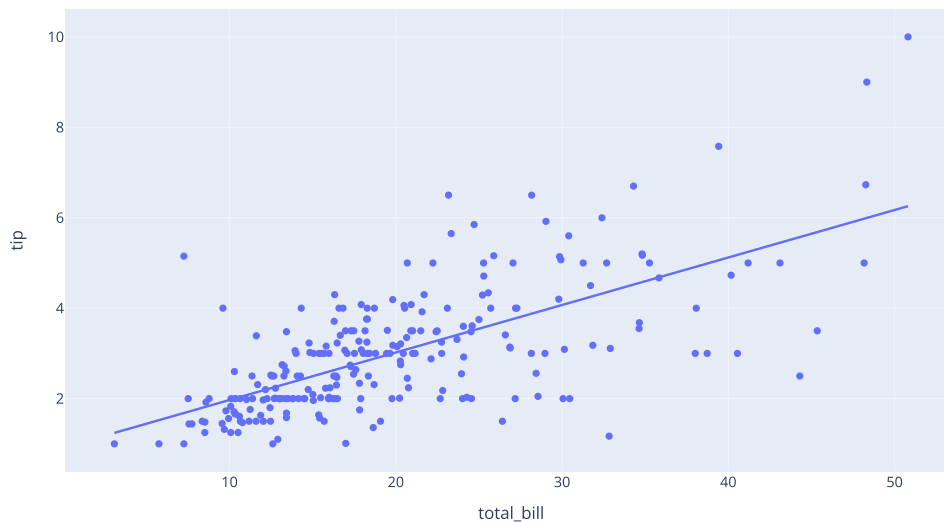
Linear fit trendlines with Plotly Express

[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 \(/python/styling-plotly-express/\)](#).

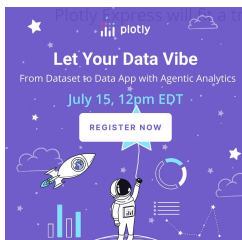
Plotly Express allows you to add [Ordinary Least Squares \(https://en.wikipedia.org/wiki/Ordinary_least_squares\)](https://en.wikipedia.org/wiki/Ordinary_least_squares) regression trendline to scatterplots with the `trendline` argument. In order to do so, you will need to [install statsmodels and its dependencies \(https://www.statsmodels.org/stable/install.html\)](https://www.statsmodels.org/stable/install.html). Hovering over the trendline will show the equation of the line and its R-squared value.

```
import plotly.express as px

df = px.data.tips()
fig = px.scatter(df, x="total_bill", y="tip", trendline="ols")
fig.show()
```



Fitting multiple lines and retrieving the model parameters



...trendline per trace, and allows you to access the underlying model parameters for all the models.

```
import plotly.express as px

df = px.data.tips()
fig = px.scatter(df, x="total_bill", y="tip", facet_col="smoker", color="sex", trendline="ols")
fig.show()

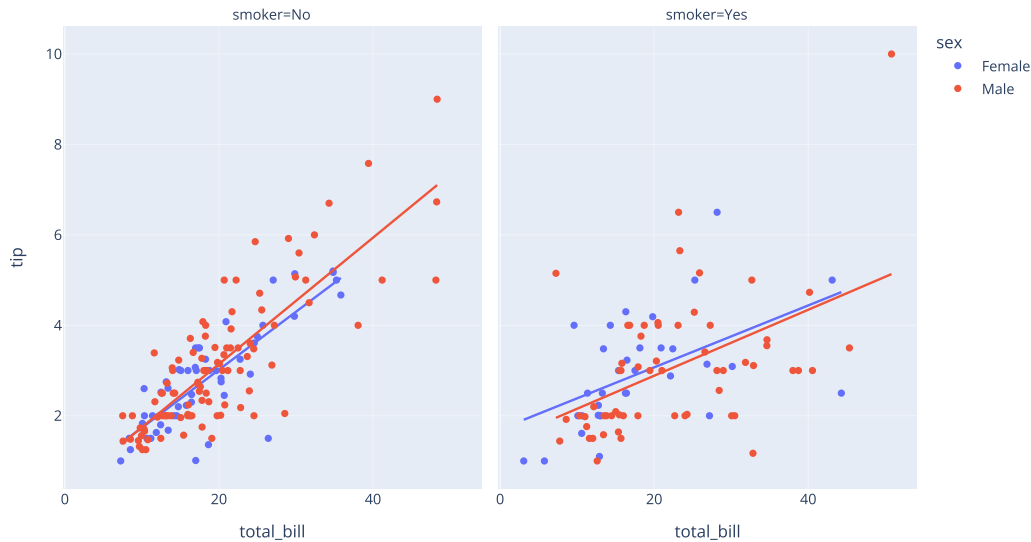
results = px.get_trendline_results(fig)
print(results)

results.query("sex == 'Male' and smoker == 'Yes'").px_fit_results.iloc[0].summary()
```

ess
the

ultiple

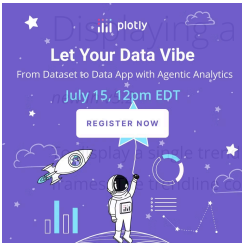
hing



sex	smoker	px_fit_results
0	Female	No <statsmodels.regression.linear_model.Regressio...
1	Female	Yes <statsmodels.regression.linear_model.Regressio...
2	Male	No <statsmodels.regression.linear_model.Regressio...
3	Male	Yes <statsmodels.regression.linear_model.Regressio...

OLS Regression Results						
Dep. Variable:	y	R-squared:	0.232			
Model:	OLS	Adj. R-squared:	0.219			
Method:	Least Squares	F-statistic:	17.56			
Date:	Tue, 08 Jul 2025	Prob (F-statistic):	9.61e-05			
Time:	20:26:57	Log-Likelihood:	-101.03			
No. Observations:	60	AIC:	206.1			
Df Residuals:	58	BIC:	210.2			
Df Model:	1					
Covariance Type:	nonrobust					
	coef	std err	t	P> t 	[0.025	0.975]
const	1.4253	0.424	3.361	0.001	0.576	2.274
x1	0.0730	0.017	4.190	0.000	0.038	0.108
Omnibus:	21.841	Durbin-Watson:	1.383			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	33.031			
Skew:	1.315	Prob(JB):	6.72e-08			
Kurtosis:	5.510	Cond. No.	60.4			

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

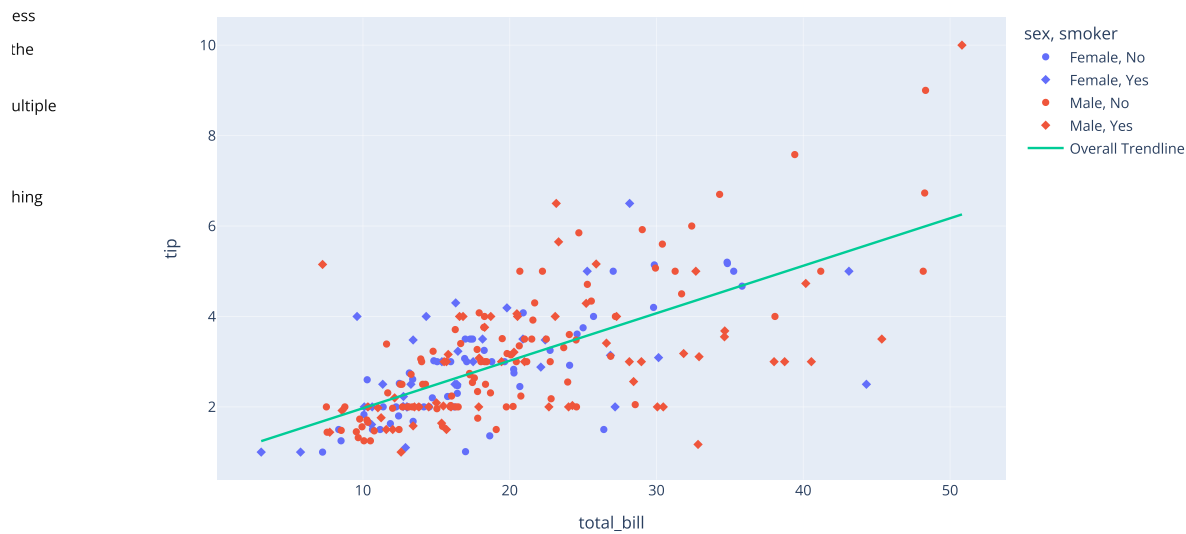


single trendline with multiple traces

line using the entire dataset, set the trendline_scope argument to "overall". The same trendline will be overlaid on all facets and animation color can be overridden with trendline_color_override.

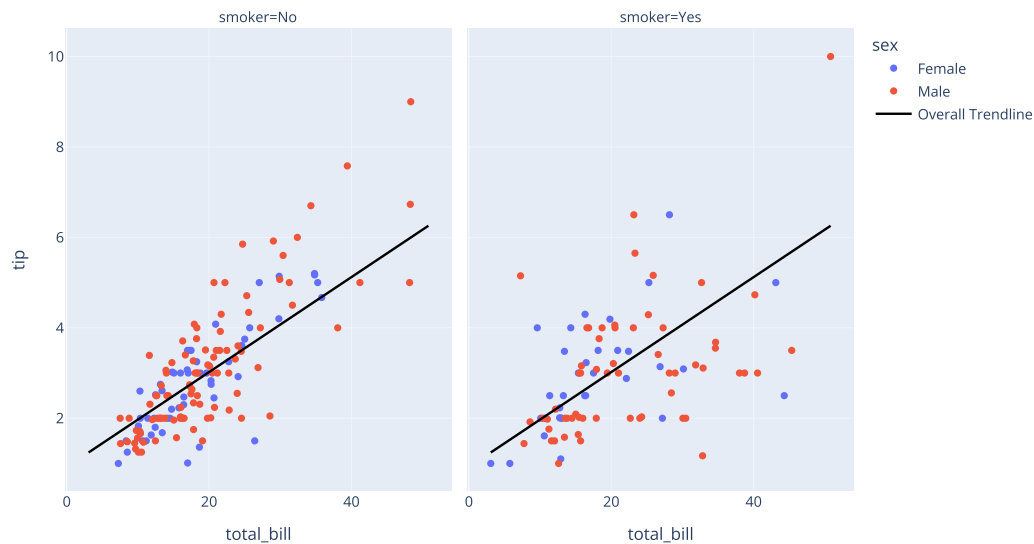
```
import plotly.express as px
```

```
df = px.data.tips()
fig = px.scatter(df, x="total_bill", y="tip", symbol="smoker", color="sex", trendline="ols", trendline_scope="overall")
fig.show()
```

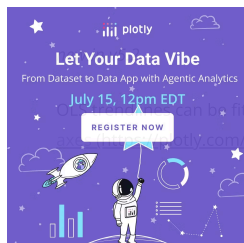


```
import plotly.express as px
```

```
df = px.data.tips()
fig = px.scatter(df, x="total_bill", y="tip", facet_col="smoker", color="sex",
                trendline="ols", trendline_scope="overall", trendline_color_override="black")
fig.show()
```



OLS Parameters



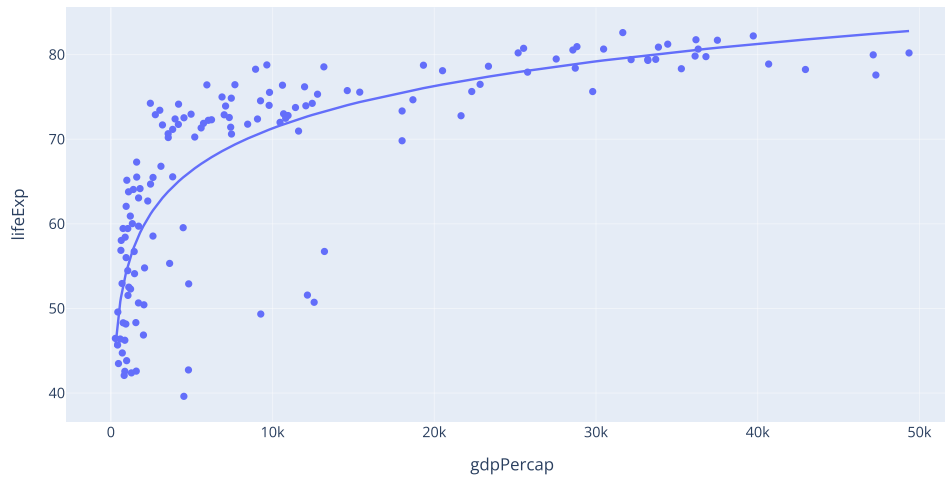
with log transformations to both X or Y data using the `trendline_options` argument, independently of whether or not the plot has [logarithmic](https://plotly.com/python/log-plot/)

```
import plotly.express as px

df = px.data.gapminder(year=2007)
fig = px.scatter(df, x="gdpPercap", y="lifeExp",
                 trendline="ols", trendline_options=dict(log_x=True),
                 title="Log-transformed fit on linear axes")

fig.show()
```

Log-transformed fit on linear axes

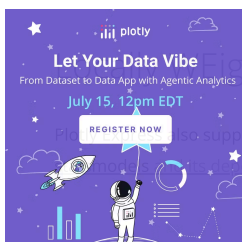
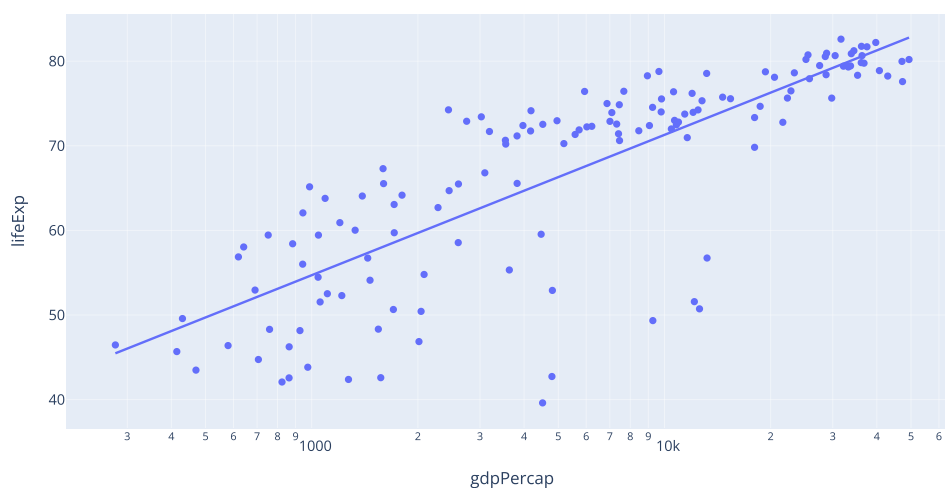


```
import plotly.express as px

df = px.data.gapminder(year=2007)
fig = px.scatter(df, x="gdpPercap", y="lifeExp", log_x=True,
                 trendline="ols", trendline_options=dict(log_x=True),
                 title="Log-scaled X axis and log-transformed fit")

fig.show()
```

Log-scaled X axis and log-transformed fit



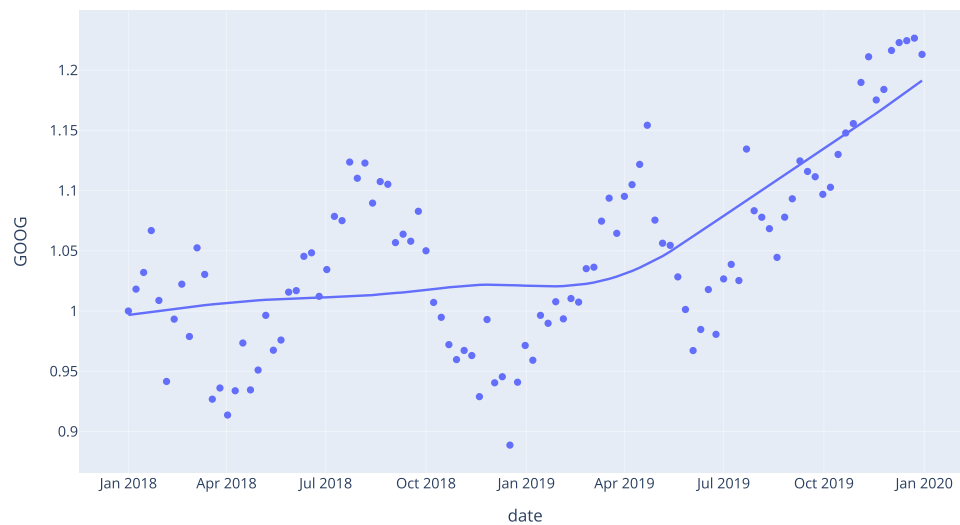
Weighted Scatterplot Smoothing (LOWESS)

Plotly supports non-linear [LOWESS](https://en.wikipedia.org/wiki/Local_regression) (https://en.wikipedia.org/wiki/Local_regression) trendlines. In order use this feature, you will need to [install](https://www.statsmodels.org/stable/install.html) dependencies (<https://www.statsmodels.org/stable/install.html>).

```
import plotly.express as px

df = px.data.stocks(datetimes=True)
fig = px.scatter(df, x="date", y="GOOG", trendline="lowess")
fig.show()
```

ess
the
ultiple
hing

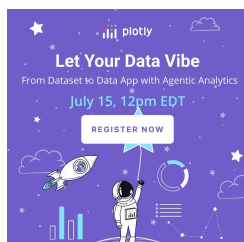
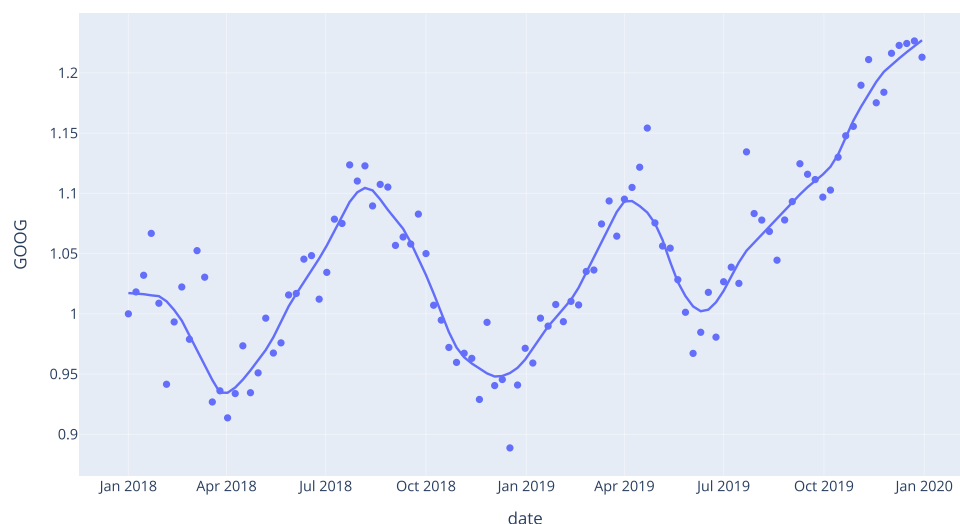


new in v5.2

The level of smoothing can be controlled via the `frac` trendline option, which indicates the fraction of the data that the LOWESS smoother should include. The default is a fairly smooth line with `frac=0.6666` and lowering this fraction will give a line that more closely follows the data.

```
import plotly.express as px

df = px.data.stocks(datetimes=True)
fig = px.scatter(df, x="date", y="GOOG", trendline="lowess", trendline_options=dict(frac=0.1))
fig.show()
```



Moving Averages

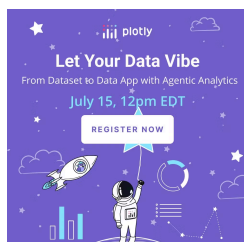
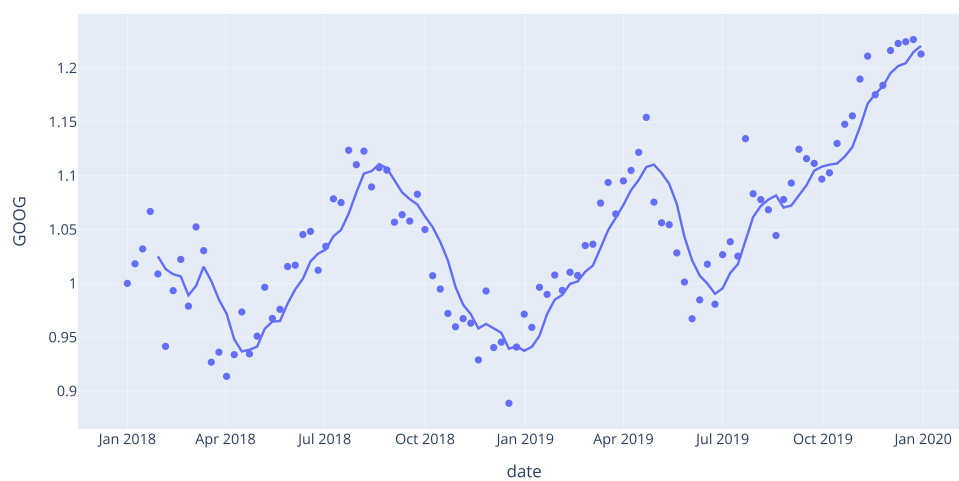
new in v5.2

Plotly Express can leverage Pandas' [rolling](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.rolling.html) (<https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.rolling.html>), [ewm](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.ewm.html) (<https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.ewm.html>) and [expanding](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.expanding.html) (<https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.expanding.html>) functions in trendlines as well, for example to display moving averages. Values passed to `trendline_options` are passed directly to the underlying Pandas function (with the exception of the `function` and `function_options` keys, see below).

```
import plotly.express as px

df = px.data.stocks(datetimes=True)
fig = px.scatter(df, x="date", y="GOOG", trendline="rolling", trendline_options=dict(window=5),
                title="5-point moving average")
fig.show()
```

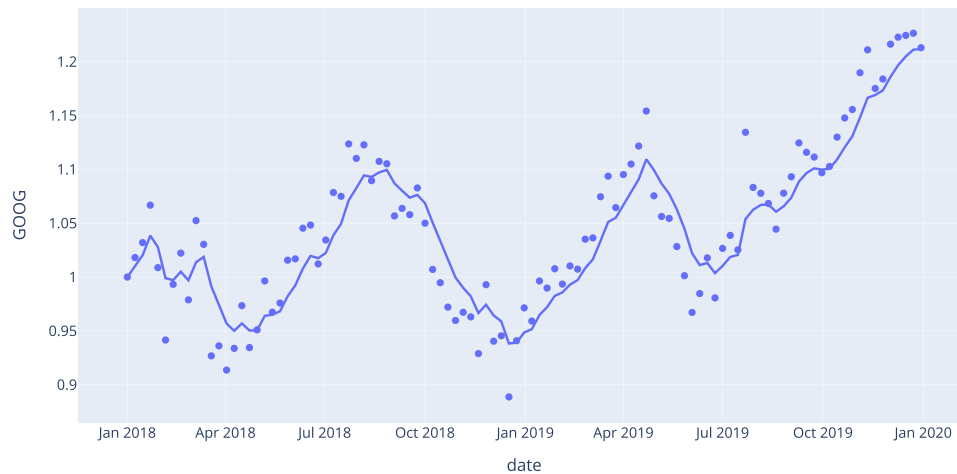
5-point moving average



```
import plotly.express as px

df = px.data.stocks(datetimes=True)
fig = px.scatter(df, x="date", y="GOOG", trendline="ewm", trendline_options=dict(halflife=2),
                title="Exponentially-weighted moving average (halflife of 2 points)")
fig.show()
```

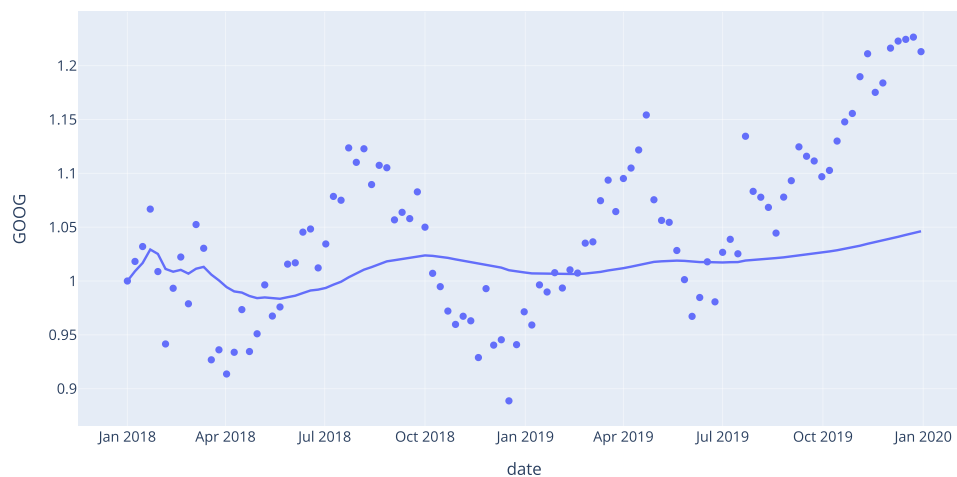
Exponentially-weighted moving average (halflife of 2 points)



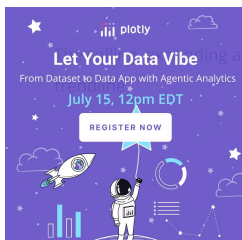
```
import plotly.express as px

df = px.data.stocks(datetimes=True)
fig = px.scatter(df, x="date", y="GOOG", trendline="expanding", title="Expanding mean")
fig.show()
```

Expanding mean



Other Functions

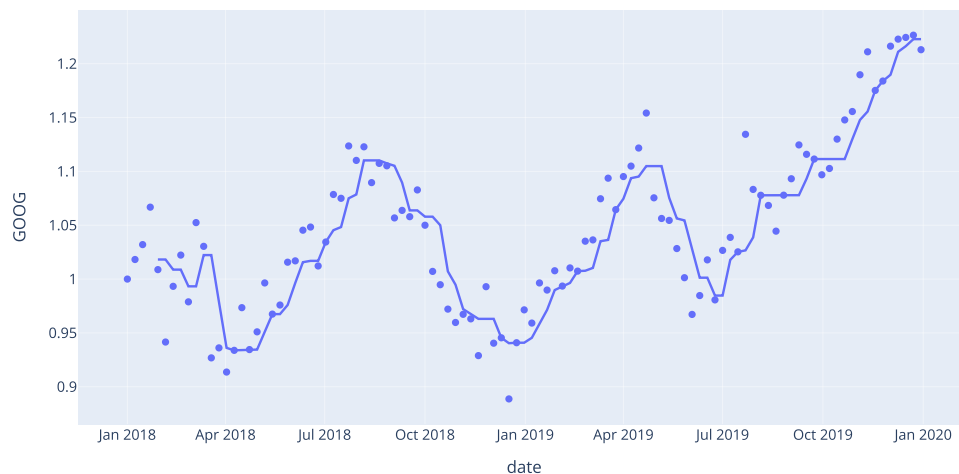


and ewm trendlines support other functions than the default mean, enabling, for example, a moving-median trendline, or an expanding-max

```
import plotly.express as px

df = px.data.stocks(datetimes=True)
fig = px.scatter(df, x="date", y="GOOG", trendline="rolling", trendline_options=dict(function="median", window=5),
                title="Rolling Median")
fig.show()
```

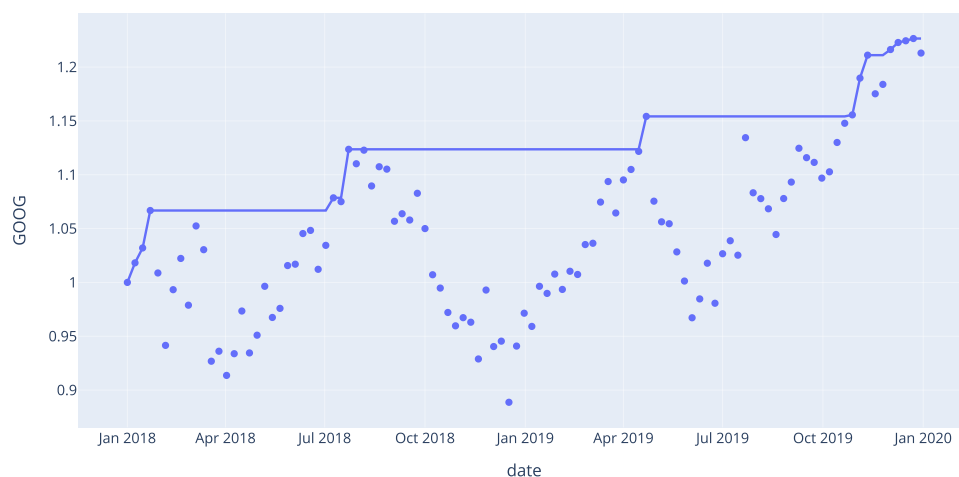
Rolling Median



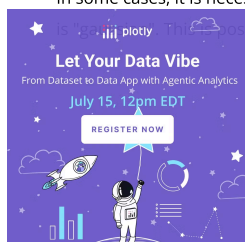
```
import plotly.express as px

df = px.data.stocks(datetimes=True)
fig = px.scatter(df, x="date", y="GOOG", trendline="expanding", trendline_options=dict(function="max"),
                title="Expanding Maximum")
fig.show()
```

Expanding Maximum



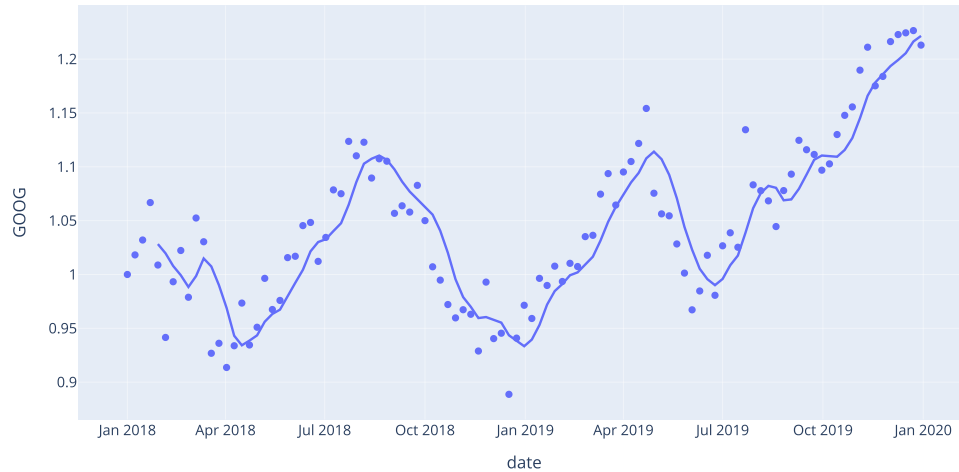
In some cases, it is necessary to pass options into the underlying Pandas function, for example the `std` parameter must be provided if the `win_type` argument to `rolling` is possible with the `function_args` trendline option.




```
import plotly.express as px

df = px.data.stocks(datetimes=True)
fig = px.scatter(df, x="date", y="GOOG", trendline="rolling",
                trendline_options=dict(window=5, win_type="gaussian", function_args=dict(std=2)),
                title="Rolling Mean with Gaussian Window")
fig.show()
```

Rolling Mean with Gaussian Window



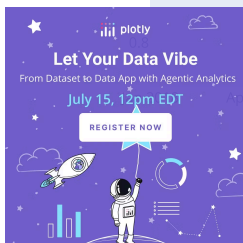
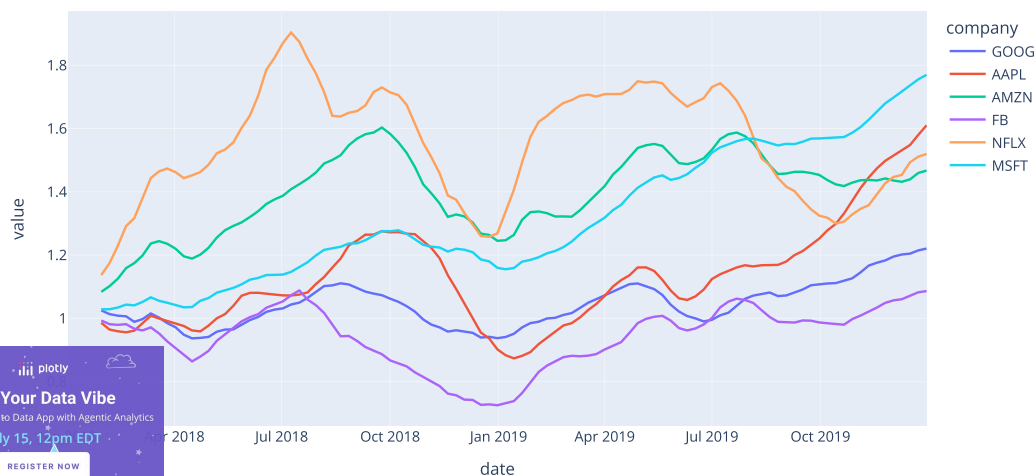
Displaying only the trendlines

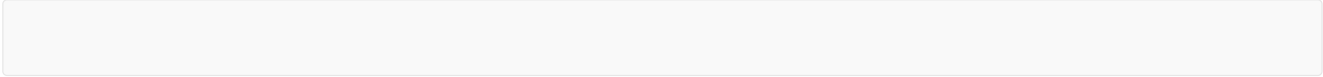
In some cases, it may be desirable to show only the trendlines, by removing the scatter points.

```
import plotly.express as px

df = px.data.stocks(indexed=True, datetimes=True)
fig = px.scatter(df, trendline="rolling", trendline_options=dict(window=5),
                title="5-point moving average")
fig.data = [t for t in fig.data if t.mode == "lines"]
fig.update_traces(showlegend=True) #trendlines have showlegend=False by default
fig.show()
```

5-point moving average





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>).

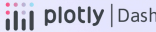
Everywhere in this page that you see fig.show(), you can display the same figure in a Dash application by passing it to the figure argument of the [Graph component](https://dash.plot.ly/dash-core-components/graph) (<https://dash.plot.ly/dash-core-components/graph>) from the built-in dash_core_components 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	33889923	Asia	43.828	976.5883384
Albania	2606023	Europe	76.423	5927.629120999999
Algeria	3333226	Africa	72.261	6223.267465
Angola	22326676	Africa	42.731	4707.211267
Argentina	40301927	Americas	75.32	12779.37964
Australia	20434176	Oceania	81.235	34435.367639999995
Austria	8199783	Europe	79.829	40106.4927
Bahrain	708673	Asia	75.635	29796.44634
Bangladesh	158448339	Asia	64.862	1391.253792
Belgium	10392226	Europe	79.441	33892.48588
Benin	8878314	Africa	56.728	1441.284873
Bolivia	9139152	Americas	65.354	3822.137884



(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/>)

Support

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

Pricing

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

