plotly

Interactive Visualizations with Plotly

Celia López | Customer Success Engineer

PyData Madrid May 2022

What is Plotly?

Plotly, as a company, offers:

- **Plotly:** data visualization library for several programming languages (Python, R, Julia)
- **Dash Open Source:** Python and R library for building interactive web apps, no HTML, CSS or JS knowledge needed.
- **Dash Enterprise:** Dash for companies, with advanced functionalities and support.

4M+ monthly Plotly downloads

300k+ monthly Dash downloads

Session Structure

1. What is Plotly and why I would be interested.
2. How to create and modify Plotly plots.
3. How to keep learning Plotly and solve issues.

plotly

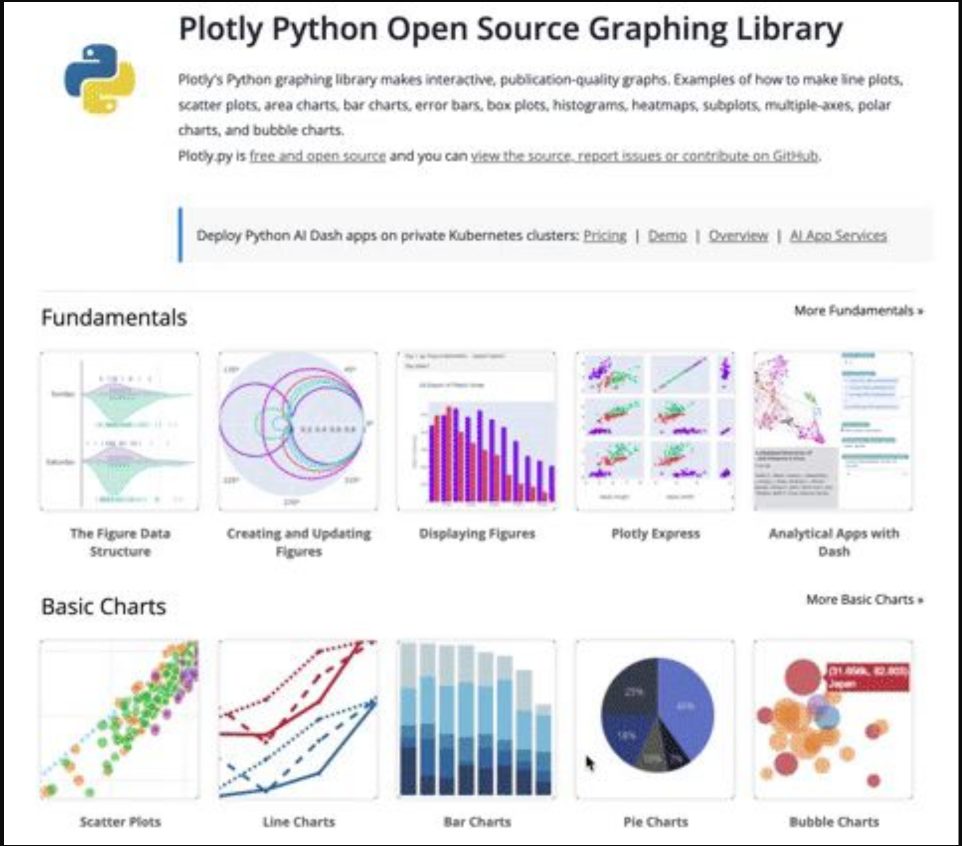
What does Plotly offer?

Its advantages:

- **Interactivity**
- Flexibility
- Easy to learn (comprehensive Docs + open community active support)

+100 types of plots:

- 3D plots
- Maps
- Mixed subplots



Plotly Python Open Source Graphing Library

Plotly's Python graphing library makes interactive, publication-quality graphs. Examples of how to make line plots, scatter plots, area charts, bar charts, error bars, box plots, histograms, heatmaps, subplots, multiple-axes, polar charts, and bubble charts.

Plotly.py is free and open source and you can view the source, report issues or contribute on GitHub.

Deploy Python AI Dash apps on private Kubernetes clusters: Pricing | Demo | Overview | AI App Services

Fundamentals

More Fundamentals »

The Figure Data Structure | Creating and Updating Figures | Displaying Figures | Plotly Express | Analytical Apps with Dash

Basic Charts

More Basic Charts »

Scatter Plots | Line Charts | Bar Charts | Pie Charts | Bubble Charts

plotly

Creating and modifying plots with Plotly

Plotly's most used modules are:

- **plotly.express (px):** for creating plots in a fast and simple way.
- **plotly.graph_objects (go):** for creating and modifying plots in a more elaborate way.

```
!pip install plotly

import plotly.express as px
import plotly.graph_objects as go
import plotly.io as pio
from plotly.subplots import make_subplots
```

Plots created with **px** and **go:**

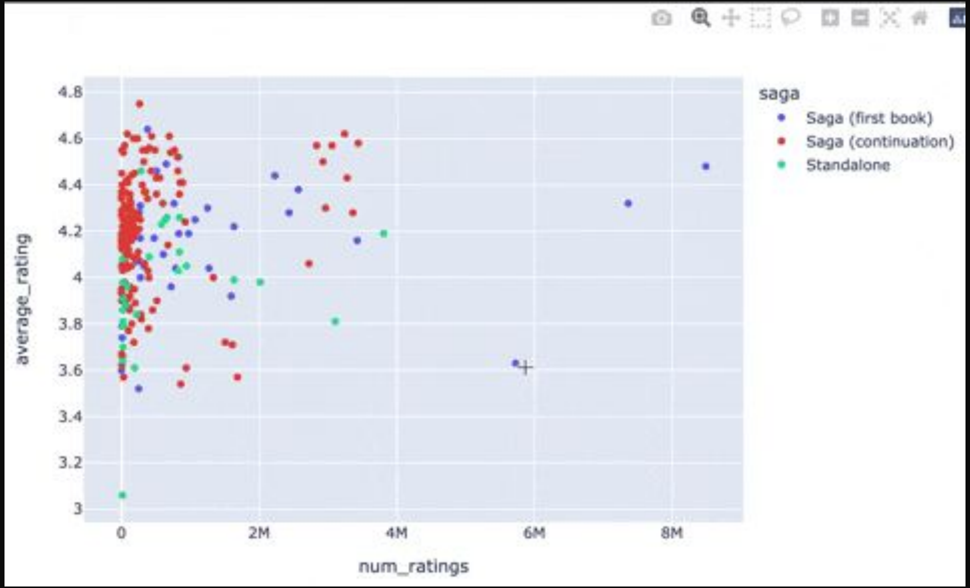
- Have the same internal structure (they are the same object type).
- Can be modified with the same methods: `.update_layout()`, `.add_traces()`
- Can be combined. In fact, it is common to create a plot with **`px`** and modify it with **`go`**.

plotly

Creating plots: Scatter plot interactivo

```
import plotly.express as px
fig_scatter = px.scatter(
    books_df_scatter, x="num_ratings", y="average_rating", color="saga",
    hover_data=["title", "series_name", "series_n"])

fig_scatter.show()
```

- Additional information about a point when users **hover** over it.
- Menu with options: **download**, zoom, selection.
- Zoom on **selected region**.
- **Zoom reset** on double click or when clicking 'reset axes'

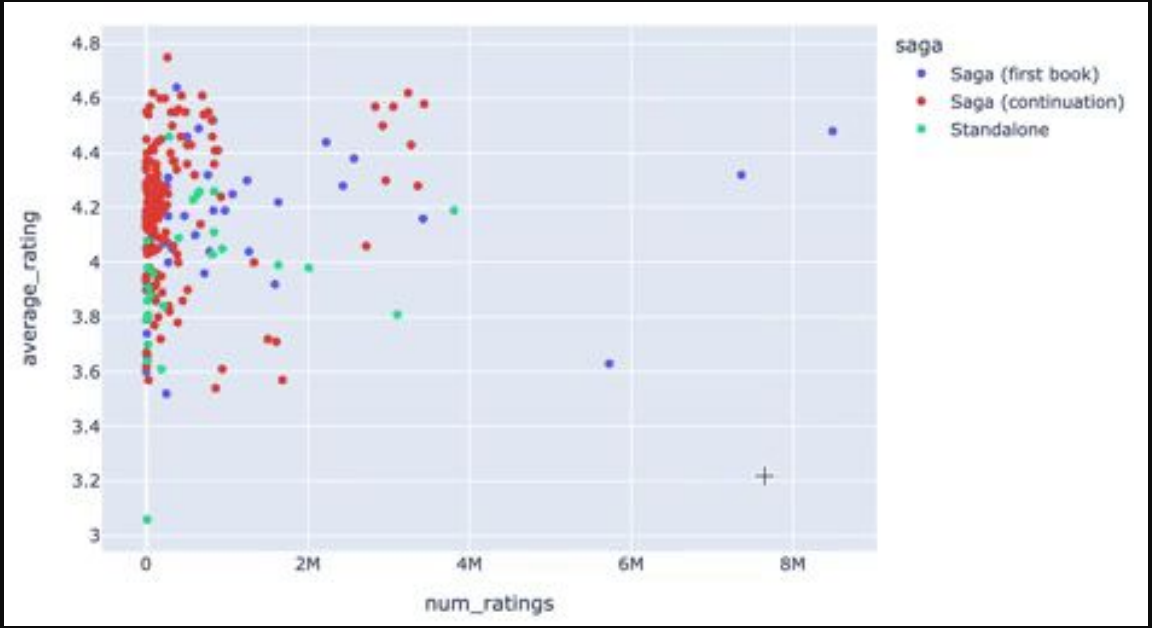


plotly

Creating plots: Interactive legend + Hover data

```
fig_scatter = px.scatter(
   books_df_scatter, x="num_ratings", y="average_rating", color="saga",
   hover_data=["title", "series_name", "series_n"],
   )
fig_scatter.show()
```

- **Click** = Show/hide group.
- **Double click** = Isolate that group.
- `hover_data` allows us to include additional information from the dataframe that isn't included in other axes (x, y, color).



plotly

Creating plots: Internal Structure

Plot structure is similar to nested dicts:

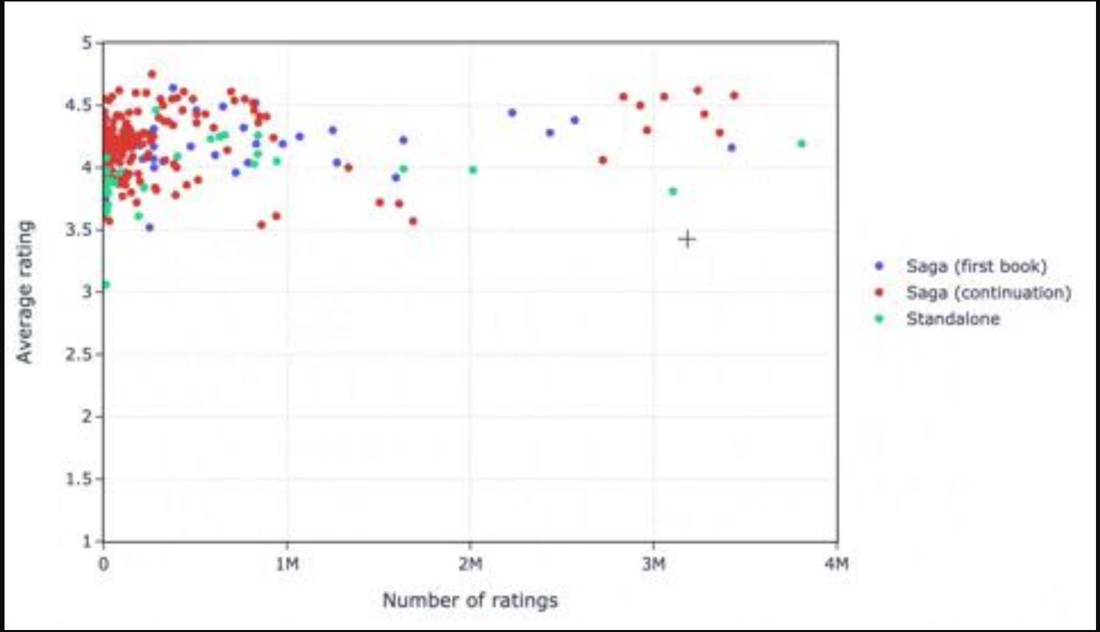
- `data` (list of *traces*)
    - Data and related values (eg. group color).
    - Modified with `.update_traces()`
- `layout`
    - Appearance.
    - Modified with `.update_layout()`

```
<class 'plotly.graph_objs._figure.Figure'>

Figure({
    'data': [{'hovertemplate': 'num_ratings=%{x}<br>average_rating=%{y}<extra></extra>',
              'legendgroup': '',
              'marker': {'color': '#636efa', 'symbol': 'circle'},
              'mode': 'markers',
              'name': '',
              'orientation': 'v',
              'showlegend': False,
              'type': 'scatter',
              'x': array([8491079, 3277548, ...,   15518,  285067,   16896]),
              'xaxis': 'x',
              'y': array([4.48, 4.43, 4.58, ..., 3.64, 4.46, 3.86]),
              'yaxis': 'y'}],
    'layout': {'legend': {'tracegroupgap': 0},
               'margin': {'t': 60},
               'template': '...',
               'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0],
                         'title': {'text': 'num_ratings'}},
               'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0],
                         'title': {'text': 'average_rating'}}}
})
```

plotly

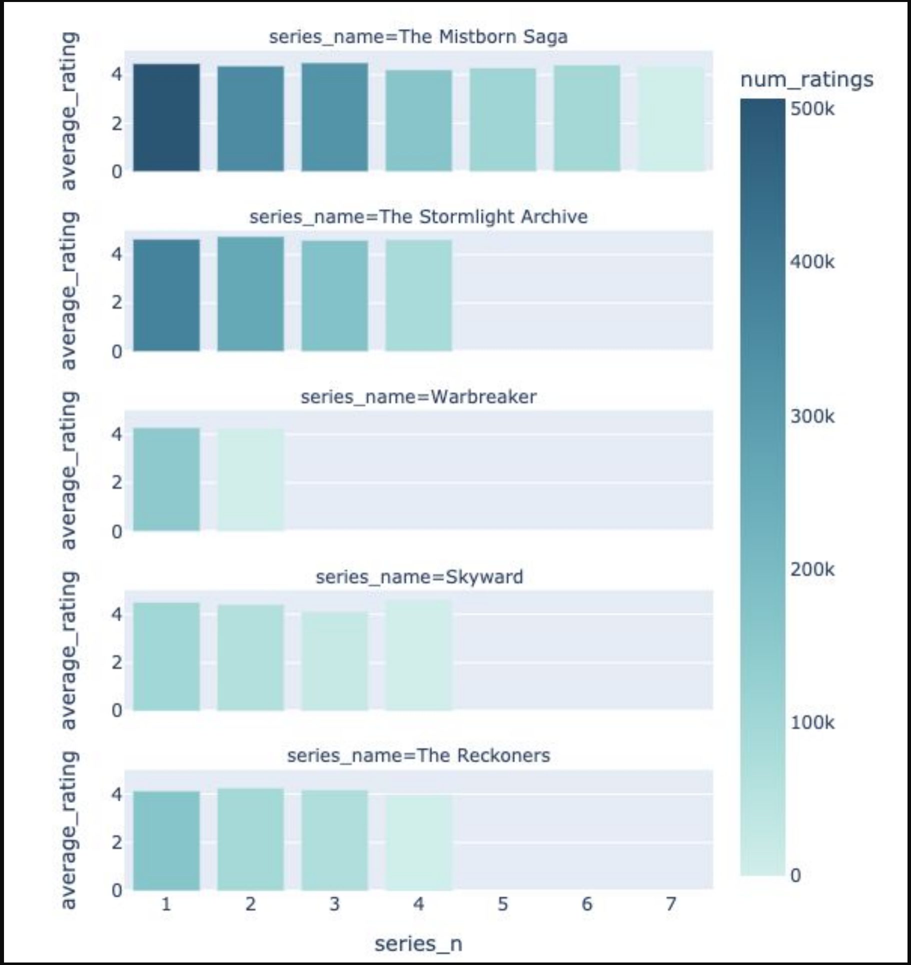
Modifying plots: layout and data/traces

```
# Changing appearance
# No need to reassign to fig_scatter
fig_scatter.update_layout(
   template = 'plotly_white+borders',
   xaxis = {'title':'Number of ratings',
           'range':[0, 4000000]},
   yaxis = {'title':'Average rating',
           'range':[1,5]},
   legend = {'title': None,
            'x': 1.02,
            'y': 0.5}
)

# Specifying structure of hover info
fig_scatter.update_traces(
   hovertemplate = '<b>%{customdata[0]}</b>
<br>%{customdata[1]} %{customdata[2]}'
)

fig_scatter.show()
```



Creating plots: Multiple plots

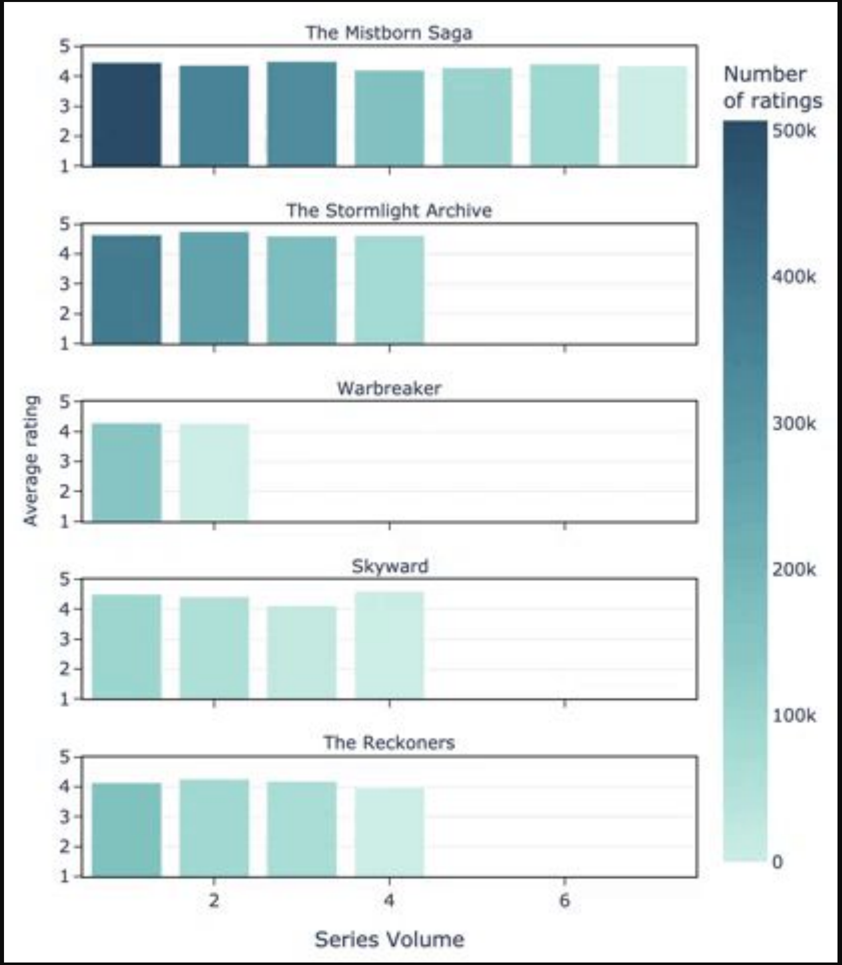
```
fig_sanderson = px.bar(
    sanderson_sagas_df,
    x='series_n', y='average_rating',
    # Create a plot for each 'series_name' value
    facet_col='series_name', facet_col_wrap=1,
    # Specify continuous var for the color and its palette
    color = 'num_ratings',
    color_continuous_scale = px.colors.sequential.Teal,
    hover_data = ['title'],
    # Specify width and height
    width = 600, height = 700
)

fig_sanderson.show()
```



plotly

Modifying plots: Multiple plots

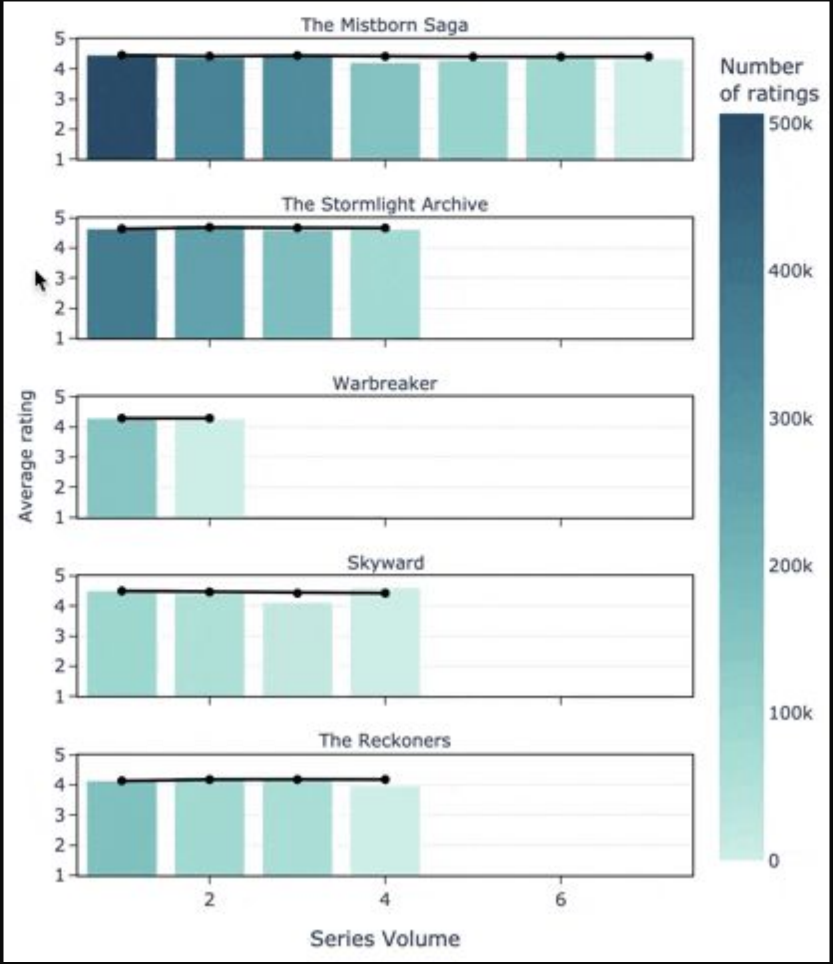
```
# Change appearance
fig_sanderson.update_layout(
   coloraxis_colorbar = {'title':'Number<br>of ratings'},
   xaxis = {'title':'Series Volume'},
   yaxis = {'range':[1,5]},
   template = 'plotly_white+borders'
  )

# Remove = from labels/plot titles
fig_sanderson.for_each_annotation(
  lambda a: a.update(text=a.text.split("=")[-1])
  )

# Remove Y axis title from each subplot
fig_sanderson.for_each_yaxis(
  lambda y: y.update(title=None)
 )

# Add a shared title for Y axis
fig_sanderson.add_annotation(
   text='Average rating',
   x=-0.11, y=0.5,
   xref="paper", yref="paper",
   textangle=-90, showarrow=False
   )
```



plotly

Modifying plots: Adding traces

```
sanderson_sagas = sanderson_sagas_df.series_name.unique()

#  We  use  `[::-1]`  to  reverse  the  order  of
sanderson_sagas  because  in  facet  plots,  row  count
starts from the bottom (and from 1, not 0)

for i, saga in enumerate(sanderson_sagas[::-1]):
   df_i=sanderson_sagas_df.query(f'series_name=="{saga}"')
   fig_sanderson.add_trace(
       go.Scatter(
           x=df_i['series_n'], y=df_i['hist_rating'],
           mode='lines+markers', line_color='black',
           showlegend=False, hoverinfo='none'
           ),
       row=i+1, col=1)

fig_sanderson.show()
```



Keep learning:
Where to find information

Google

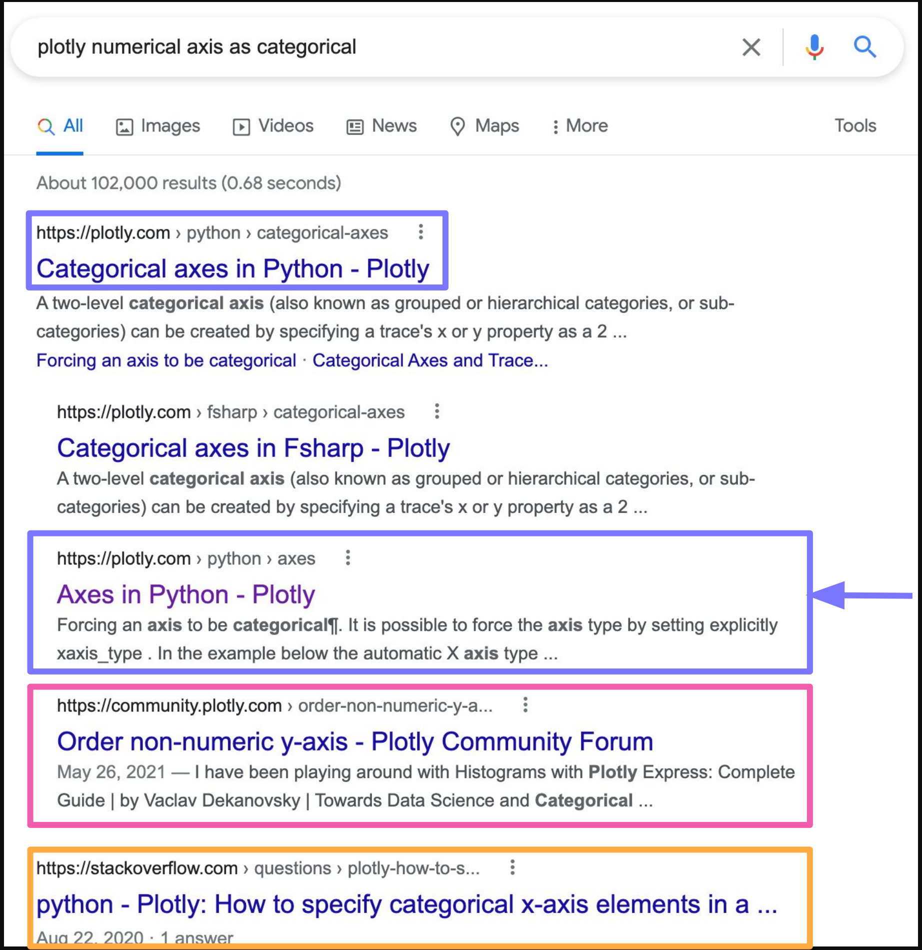
Plotly documentation for Python

- Examples for different plots, functionalities and use cases.
- Reference: Complete list of the arguments and the values they can take.

Official Plotly Forum

Stackoverflow

GitHub Issues


plotly numerical axis as categorical

About 102,000 results (0.68 seconds)

https://plotly.com › python › categorical-axes
Categorical axes in Python - Plotly
A two-level **categorical axis** (also known as grouped or hierarchical categories, or sub-categories) can be created by specifying a trace's x or y property as a 2 ...
Forcing an axis to be categorical · Categorical Axes and Trace...

https://plotly.com › fsharp › categorical-axes
Categorical axes in Fsharp - Plotly
A two-level **categorical axis** (also known as grouped or hierarchical categories, or sub-categories) can be created by specifying a trace's x or y property as a 2 ...

https://plotly.com › python › axes
Axes in Python - Plotly
Forcing an **axis** to be **categorical**¶. It is possible to force the **axis** type by setting explicitly xaxis_type . In the example below the automatic X **axis** type ...

https://community.plotly.com › order-non-numeric-y-a...
Order non-numeric y-axis - Plotly Community Forum
May 26, 2021 — I have been playing around with Histograms with **Plotly** Express: Complete Guide | by Vaclav Dekanovsky | Towards Data Science and **Categorical** ...

https://stackoverflow.com › questions › plotly-how-to-s...
python - Plotly: How to specify categorical x-axis elements in a ...
Aug 22, 2020 · 1 answer

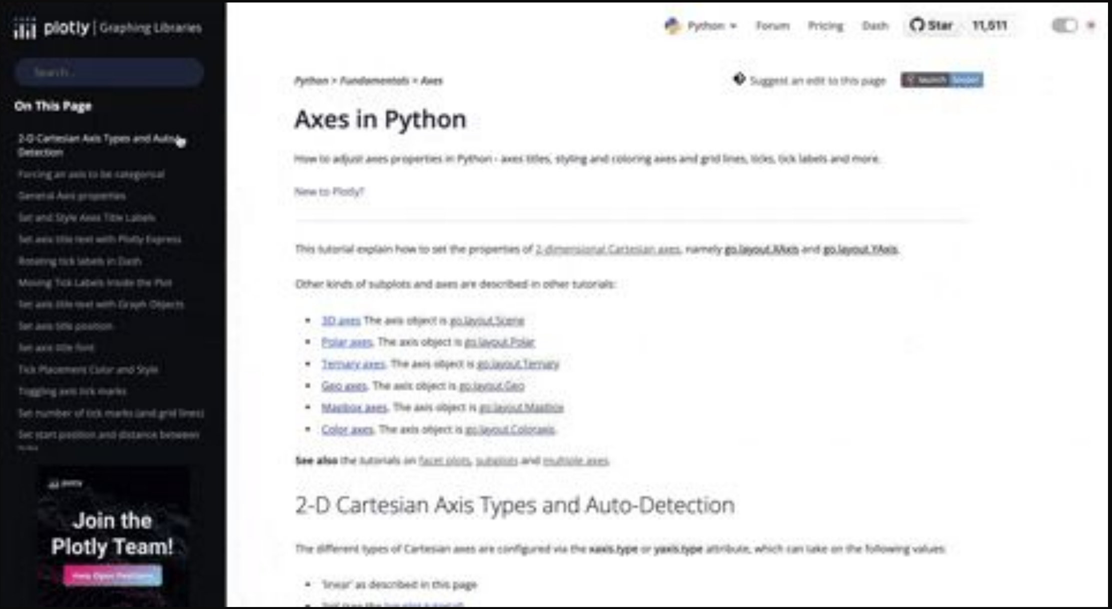
plotly

Keep Learning: How to use the documentation

The easiest way to learn how to do something with Plotly is:

1. Find an example of a plot that has a similar appearance of functionalities. We can find it in **Plotly** plot examples/use cases, in the **Forum** or in **Stackoverflow**.
2. Read the full information of that function in the ***Reference*** to change the details.
3. Ask in the **Forum** if we can't find a solution.

¡! Remember to include a MRE (*Minimal Reproducible Example*) in your question!


plotly | Graphing Libraries

Search...

On This Page

- 2-D Cartesian Axis Types and Auto-Detection
- Forcing an axis to be categorical
- General Axis properties
- Set and Style Axes Title Labels
- Set axis title text with Plotly Express
- Rotating tick labels in Dash
- Moving Tick Labels Inside the Plot
- Set axis title text with Graph Objects
- Set axis title position
- Set axis title font
- Tick Placement Color and Style
- Toggling axis tick marks
- Set number of tick marks (and grid lines)
- Set start position and distance between ticks

Join the Plotly Team!

Python ▾ Forum Pricing Dash Star 11,511

Python > Fundamentals > Axes

Suggest an edit to this page

Axes in Python

How to adjust axes properties in Python - axes titles, styling and coloring axes and grid lines, ticks, tick labels and more.

New to Plotly?

This tutorial explain how to set the properties of 2-dimensional Cartesian axes, namely **go.layout.XAxis** and **go.layout.YAxis**.

Other kinds of subplots and axes are described in other tutorials:

- 3D axes The axis object is go.layout.Scene
- Polar axes. The axis object is go.layout.Polar
- Ternary axes. The axis object is go.layout.Ternary
- Geo axes. The axis object is go.layout.Geo
- Mapbox axes. The axis object is go.layout.Mapbox
- Color axes. The axis object is go.layout.Coloraxis.

See also the tutorials on facet plots, subplots and multiple axes.

2-D Cartesian Axis Types and Auto-Detection

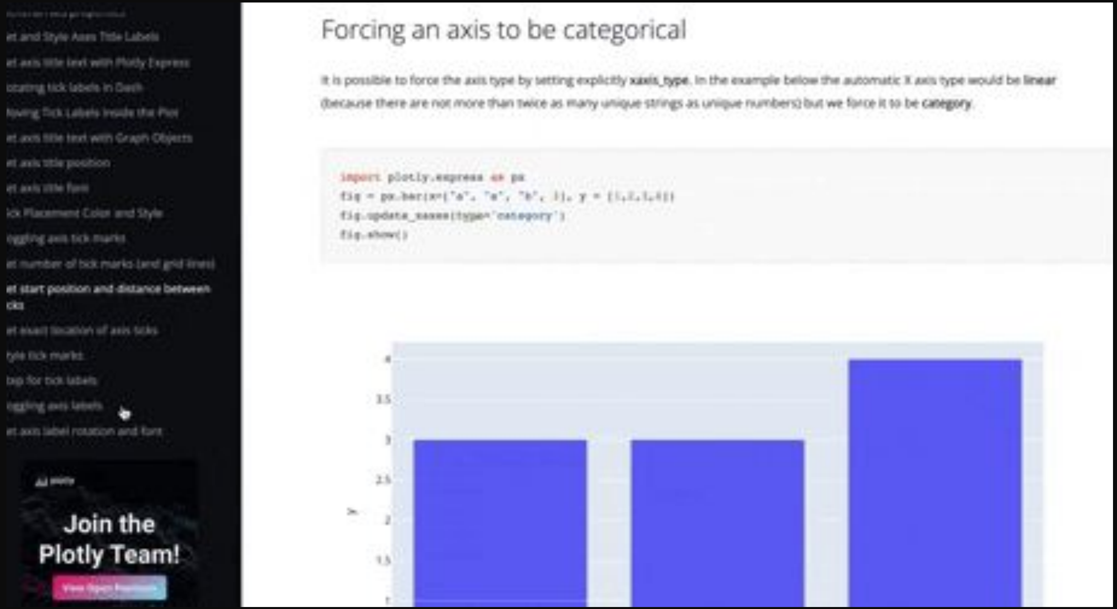
The different types of Cartesian axes are configured via the **xaxis.type** or **yaxis.type** attribute, which can take on the following values:

- 'linear' as described in this page

```
import plotly.express as px
fig = px.bar(x=["a", "a", "b", 3], y = [1,2,3,4])
fig.update_xaxes(type='category')

fig.show()
```

Keep Learning: How to use the documentation

- ***Reference*** is always in the end of the related use case documentation.
- `.update_layout()` and its methods can be used in two ways:
  - Nested dicts, with **`dict()`** or **`{}`**
  - Chaining names with underscores (_).



et and Style Axes Title Labels
et axis title text with Plotly Express
ocating tick labels in Dash
lowing Tick Labels inside the Plot
et axis title text with Graph Objects
et axis title position
et axis title font
ick Placement Color and Style
oggling axis tick marks
et number of tick marks (and grid lines)
et start position and distance between icks
et exact location of axis ticks
tyle tick marks
tep for tick labels
oggling axis labels
et axis label rotation and font
Join the Plotly Team!
View Open Positions
4
3.5
3
2.5
2
1.5
1
y

Forcing an axis to be categorical

It is possible to force the axis type by setting explicitly **xaxis_type**. In the example below the automatic X axis type would be **linear** (because there are not more than twice as many unique strings as unique numbers) but we force it to be **category**.

```
import plotly.express as px
fig = px.bar(x=["a", "a", "b", 3], y = [1,2,3,4])
fig.update_xaxes(type='category')
fig.show()
```

```
fig.update_layout(xaxis = {'type':'category'})
```

```
fig.update_xaxes(type='category')
```

Summary

1. Create a plot with `fig = px.Scatter` (or fig=px.Bar, fig=px.Bar, etc.)
2. Modify it with `fig.update_layout()` and `fig.update_traces()`.
3. Add elements with `fig.add_trace()`, `fig.add_annotation()`, `fig.add_hline()`, `fig.add_shape()`...
4. Let's interact!

Tips to keep learning and solving issues

If you have a question, someone has probably asked that before you:

1. Search in Google, the Documentation or the Forum a similar plot (use **key words**!).
2. When you have identified the function/method that does what you want, check the ***Reference*** to adapt it to your use case.

If you are already a Master of Plotly for Python: Dash Open Source : https://dash.plotly.com/

plotly

Useful Resources

Plotly for Python Documentation: https://plotly.com/python/

How to use the Documentation: https://plotly.com/python/reference/index/

Official Forum: https://community.plotly.com/c/plotly-python/

What is an MRE and how to include one in your Forum questions:
https://community.plotly.com/t/how-to-write-a-minimal-reproducible-example-mre/61502

Charming Data - Plotly and Dash Open Source Tutorial Channel:
https://www.youtube.com/channel/UCqBFsuAz41sqWcFjZkqmJqQ

GitHub plotly.py: https://github.com/plotly/plotly.py/

The data used in this presentation was webscraped with a modified version of goodreads-webscraper by Maria Antoniak y Melanie Walsh: https://github.com/maria-antoniak/goodreads-scraper

plotly

Thank You!

For more information about Plotly and Dash reach out to us at info@plotly.com!

: @plotly : @plotlygraphs

plotly