

Express graphics with Plotly

A closer look at the design and implementation of Plotly Express



Steven Liu

Follow

May 29 · 5 min read

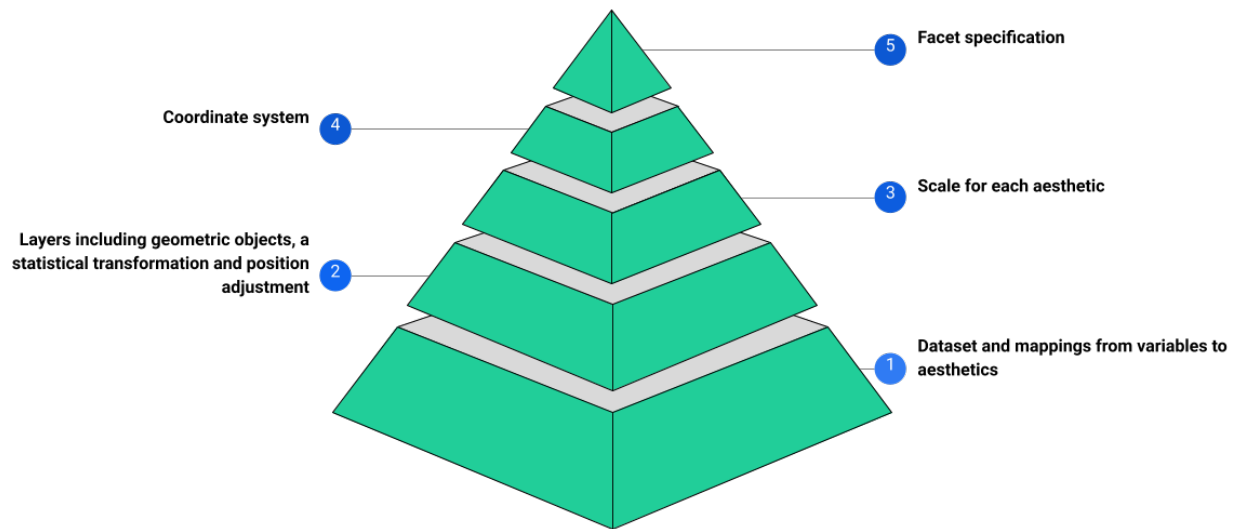
Earlier in March [plotly](#) introduced a new visualization library, Plotly Express. The main goal was to enable users to rapidly create figures by trading off some control earlier on in the visualization process, in exchange for a simple and expressive API.

In this post, we will take a closer look at the design philosophy behind Plotly Express and end with some examples to get you started!

The Layered Grammar of Graphics

Grammar refers to the system and structure of a language that helps us understand each other better. A large part of what makes Plotly Express so intuitive and easy to use is that it is influenced by the grammar of graphics. Having a grammatical foundation allows us to understand even the most complicated graphs because they all share the same basic building blocks. Once we have a grasp of these fundamental units, we can easily create a strong visual with a single succinct line of code.

The Plotly Express design is inspired by Hadley Wickham's **layered grammar of graphics**, which is “based around the idea of building up a graphic from multiple layers of data” (Wickham).



a layered grammar of graphics

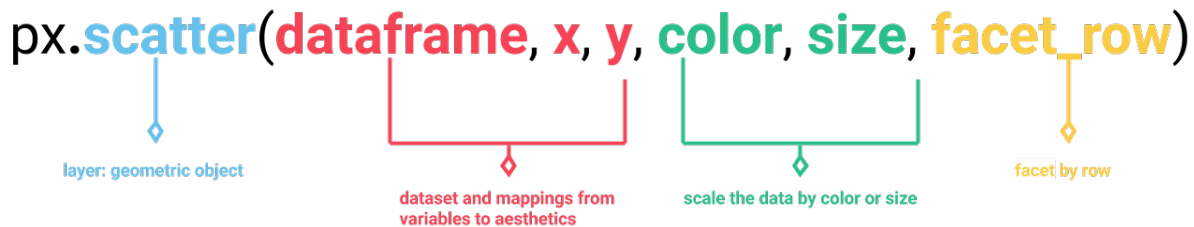
1. **Data** is where it all begins, and we start by mapping our variables to the x and y positions. These are also referred to as *aesthetics* because they are things that we can visualize.
2. **Layers** are what see on the plot. The statistical transformation layer allows us to manipulate the data by applying some function, for example, a boxplot which captures the distribution of values. On the other hand, the geometric object controls the type of plot that is created, such as a scatterplot or barplot. Finally, a position adjustment allows us to move around the geometric elements in the plot by stacking or grouping them.
3. **Scale** controls how we perceive the data. For example, we can scale by color to represent discrete variables or we can scale by size for continuous values. Choosing a good scale is crucial for producing a good graphic because if misused, it can potentially misrepresent the data and deceive the reader.

4. **Coordinate system** describes the position of the objects on the plane of the plot, and controls how the axes and gridlines are drawn. Most commonly, we will use the Cartesian coordinate system for two dimensional data.
5. **Facets** allows us to subset the entire dataset into several smaller plots to investigate patterns between variables.

By following this powerful layered grammar structure, we can build any plot we want.

Express graphics

Plotly Express has the same grammatical units, but it is structured a little differently. Consider the `scatter` function:



API for a scatterplot

All the layers are obvious with the minor exception of the geometric object, which sits on the outside. There is no need to specify a coordinate layer because by default, this will map to Cartesian coordinates. In addition, there are more arguments that provide a greater degree of control over the plot. For example, we can specify the `labels` dictionary to supply our own custom column names. Check out the documentation for a full list of the provided arguments.

Now that we have an understanding of the design, let's create some plots!

line_polar

We will get some analytics data for the 2018–2019 season from NBAstuffer. Assuming Golden State plays Milwaukee in the Finals, let's compare each teams statistics in sum.

The `line_polar` function can be easily used to create a radar plot, a good technique for comparing several variables at once.

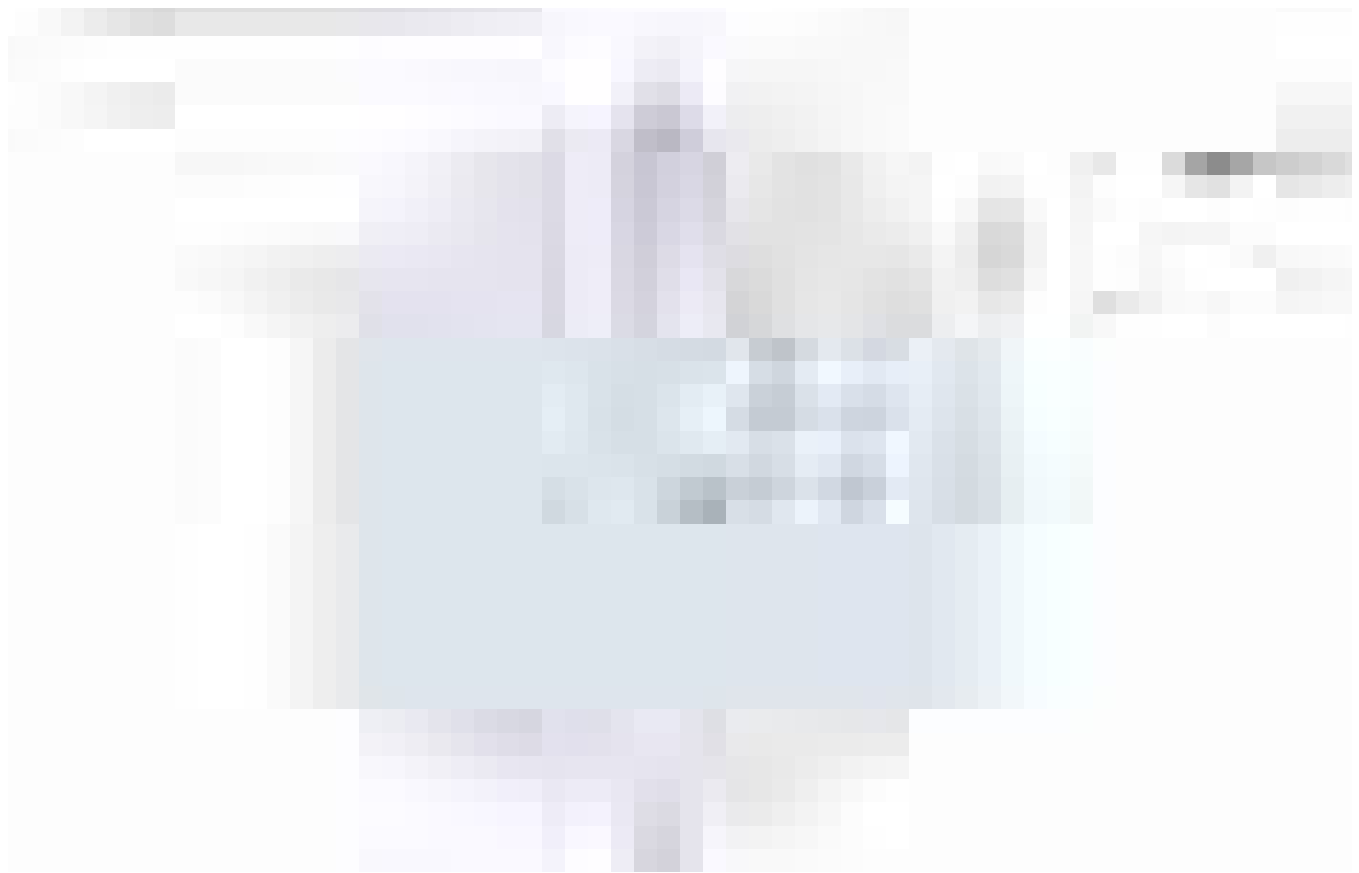
```
1 px.line_polar(teamFinal, r='value', theta='variable', color='team',  
2               hover_name='team', line_close=True,  
3               title='Golden State vs. Milwaukee',  
4               color_discrete_map={'Gol': '#FDB927', 'Mil': '#00471B'})
```

linePolar.py hosted with ❤ by GitHub

[view raw](#)

1. **Data and mappings:** `teamFinal` is the dataset and `r` maps values to the coordinates. `Theta` sets the variables along the radial axis in polar coordinates.
2. **Layers:** `line_polar` creates the geometric object. The coordinate system is already built into the `line_polar` function, negating the need to specify one.
3. **Scale:** By setting `color` to `team`, it maps the values between each team to a specified color.


The rest of the arguments simply control the design elements of the plot. We include a title for clarity and manually set the color to each team's respective colors.



```
1 px.bar_polar(df, r='APG', theta='TEAM', color='POS', hover_name='FULL NAME',  
2               template='plotly_dark', title='Team assists per game',  
3               color_discrete_sequence=px.colors.sequential.Agsunset)
```

barPolar.py hosted with ❤ by GitHub

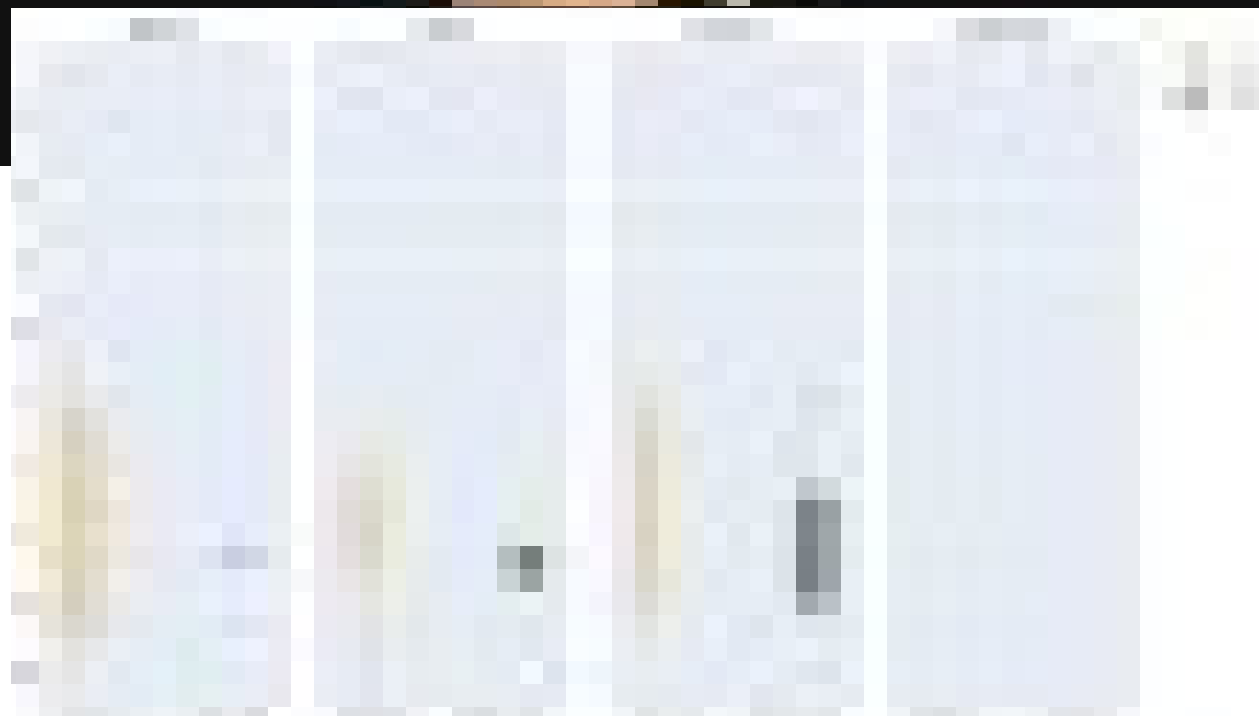
[view raw](#)



```
px.box(final_y='TS%', color='TEAM', facet_col='POS',
       color_discrete_map={'Gol': '#FDB927', 'Mil': '#00471B'})
```

pxBox.py hosted with ❤ by GitHub

[view raw](#)



bar

Say we're interested in the sum of points scored per game by each team, and want to further break that down by position. We can create something like this.

```
1 px.bar(df, x='TEAM', y='PPG', color='POS',
2        color_discrete_sequence=px.colors.sequential.Agsunset,
3        hover_name='FULL NAME', template='plotly_white')
```

pxBar.py hosted with ❤ by GitHub

[view raw](#)



Right off the bat, we observe that the center position is rare. Boston doesn't even have a true center, opting instead for a more hybrid forward-center player. The center also has fewer points per game compared to the rest of their teammates. The only exception is Jokic, who has been playing on a MVP-level this season.

scatter

What if we are interested in the relationship between minutes and points per game?

```
1 px.scatter(df, x='MPG', y='PPG', color='POS',  
2           title='Points and minutes per game', hover_name='FULL NAME')
```

pxScatter.py hosted with ❤ by GitHub

[view raw](#)



Simmons (the yellow dots) cap out at the 35 minute mark. Meanwhile, the only players who practically play the whole game are Paul George and Damian Lillard (the green and purple dots respectively on the far right).

Summary

In this post we looked at the layered grammar of graphics, and the intuition behind how we can use it to write well-informed plots. Then we practiced our grammar by creating several plots with Plotly Express. Now that you have a better sense of these concepts, hopefully it will help you in producing your own plots!

. . .

Thanks for reading, please comment, leave any feedback you may have, or you can reach out to me at [stevhliu](#) 🙌

