

Plotly Express Cheatsheet



Plotly is a graphing library for interactive, publication-quality graphs. It is *free* and *open source*.

1. Install plotly express

```
$ pip install plotly[express]
```

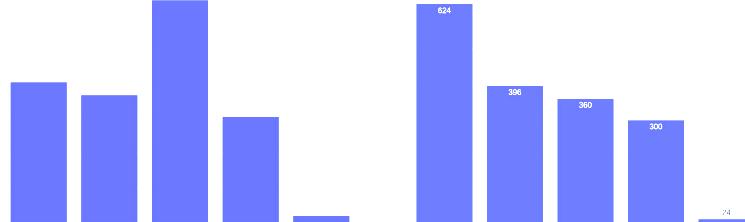
2. Import plotly express

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

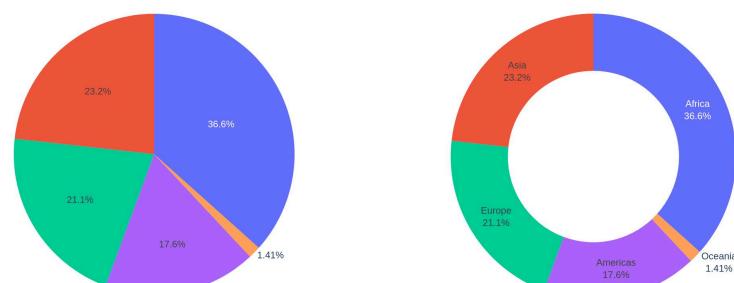
One Variable

DISCRETE

```
px.histogram(df, <column>)
px.histogram(df, <column>, text_auto=True) \
    .update_xaxes(categoryorder="total
descending")
```



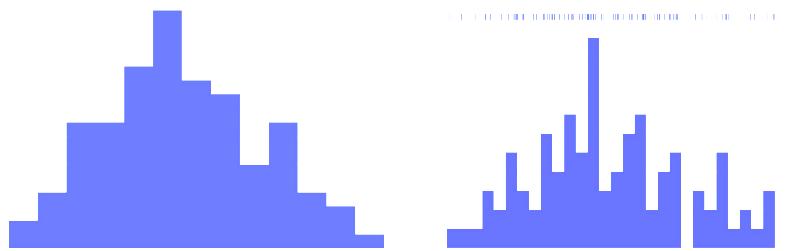
```
px.pie(df, <column>)
px.pie(df, <column>, hole=0.6) \
    .update_traces(textinfo='percent+label')
```



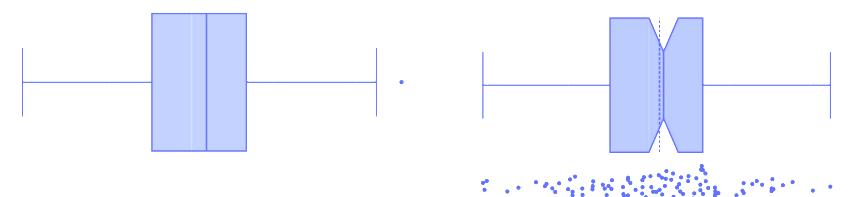
One Variable

CONTINUOUS

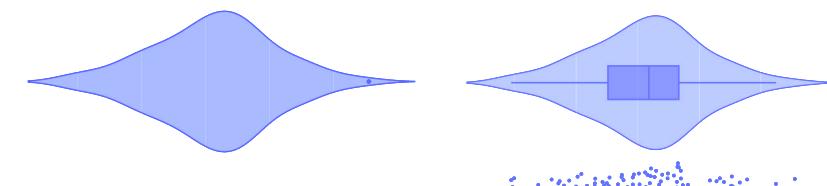
```
px.histogram(df, <column>
px.histogram(df, <column>, nbins=50, \
marginal='rug')
```



```
px.box(df, <column>)
px.box(df, <column>, notched=True,
points='all').update_traces(boxmean=True)
```



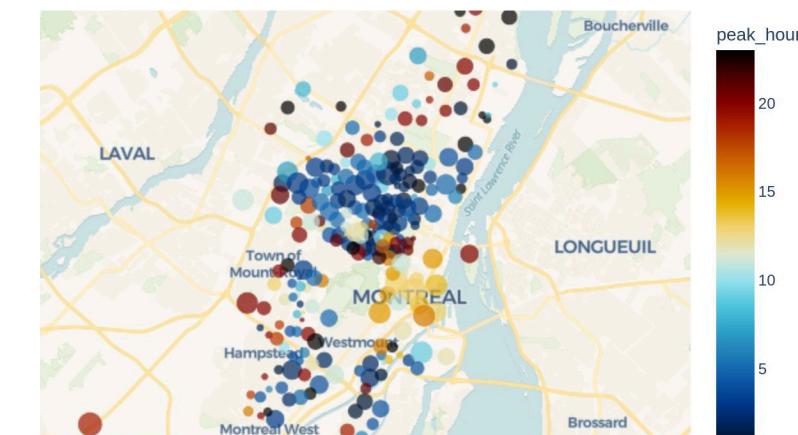
```
px.violin(df, <column>)
px.violin(df, <column>, box=True, points='all')
```



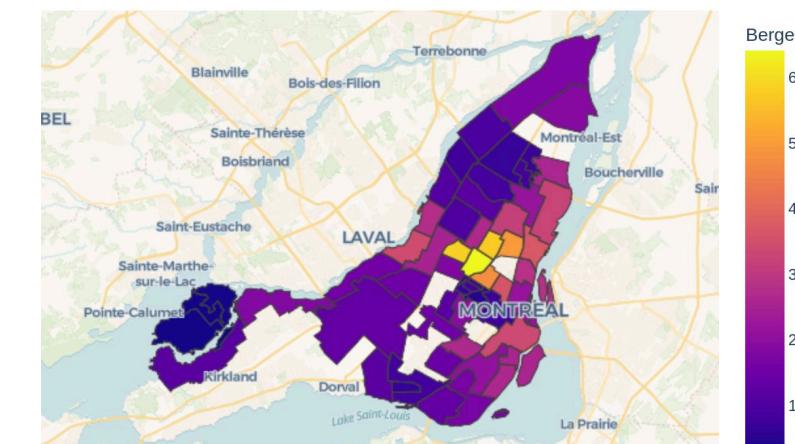
Spatial

MAPS

```
px.scatter_map(px.data.carshare(),
lat='centroid_lat', lon='centroid_lon',
color='peak_hour', size='car_hours')
```



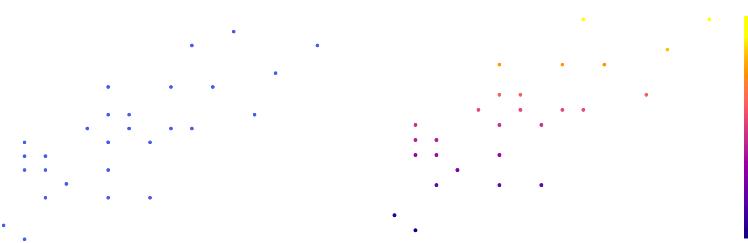
```
px.choropleth_map(px.data.election(),
geojson=px.data.election_geojson(),
color='Bergeron', locations='district',
featureidkey='properties.district',
center={'lat': 45.551, 'lon': -73.707},
zoom=9)
```



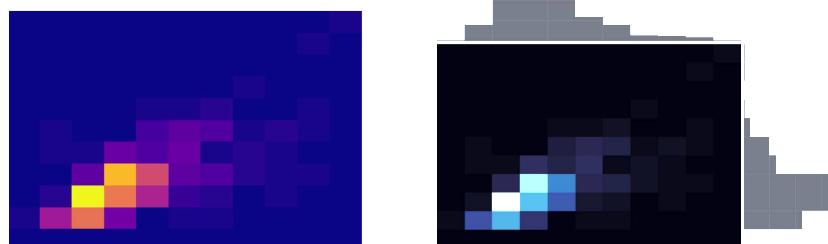
Two Variables

BOTH CONTINUOUS

```
px.scatter(df, <x_col>, <y_col>)
px.scatter(df, <x_col>, <y_col>, \
    color=<y_col>)
```

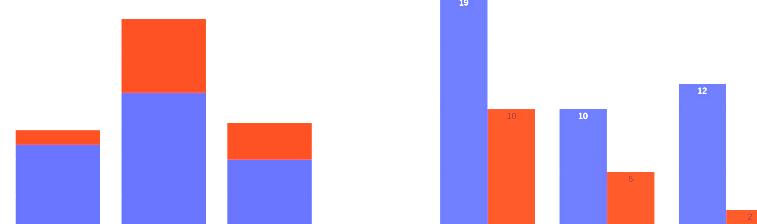


```
px.density_heatmap(df, <x_col>, <y_col>)
px.density_heatmap(df, <x_col>, <y_col>, \
    marginal_x='histogram', marginal_y='histogram', \
    color_continuous_scale=px.colors.sequential.ice)
```



BOTH DISCRETE

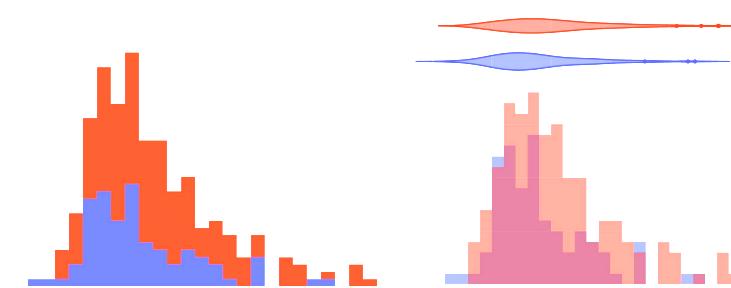
```
px.histogram(df, <x_col>, color=<y_col>)
px.histogram(df, <x_col>, color=<y_col>, \
    barmode='group', text_auto=True)\ \
    .update_xaxes(categoryorder='total descending')
```



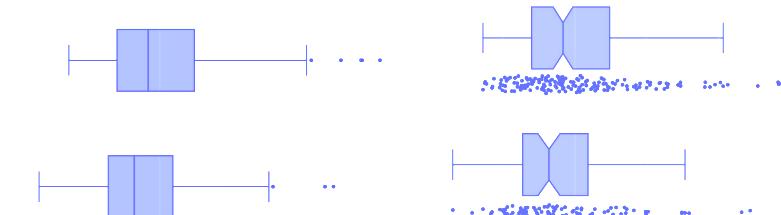
Two Variables

ONE CONTINUOUS ONE DISCRETE

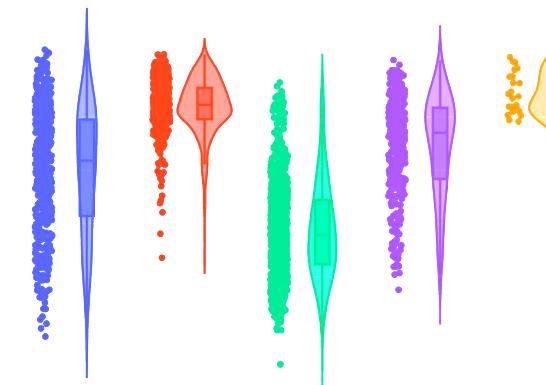
```
px.histogram(df, <continuous_col>, \
    color=<discrete_col>)
px.histogram(df, <continuous_col>, \
    color=<discrete_col>, marginal='violin' \
    barmode='overlay')
```



```
px.box(df, <x_col>, <y_col>)
px.box(df, <x_col>, <y_col>, \
    points='all', notched=True)
```



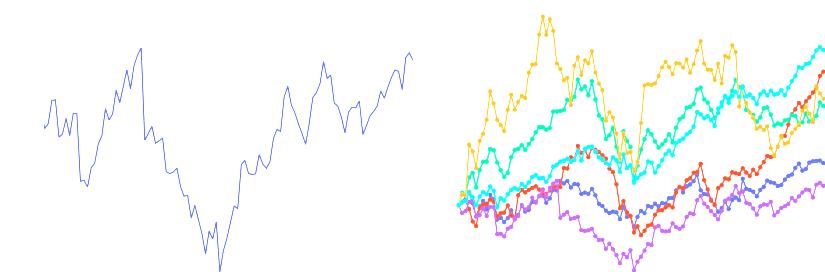
```
px.violin(px.data.gapminder(), \
    x='continent', y='lifeExp', \
    color='continent', points='all', box=True)
```



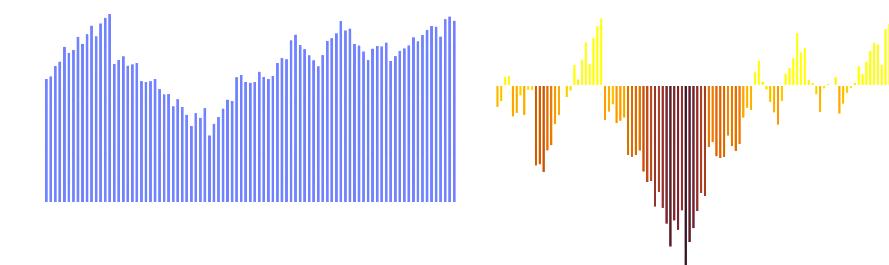
Temporal

TIME SERIES

```
px.line(px.data.stocks(), x='date', y='FB')
px.line(px.data.stocks(), x='date', \
    y=px.data.stocks().columns, markers=True)
```



```
px.bar(px.data.stocks(), x=df.index, y='FB')
df = px.data.stocks(indexed=True)-1
px.bar(df, x=df.index, y='FB', color='FB', \
    color_continuous_scale=px.colors.sequential.solar)
```



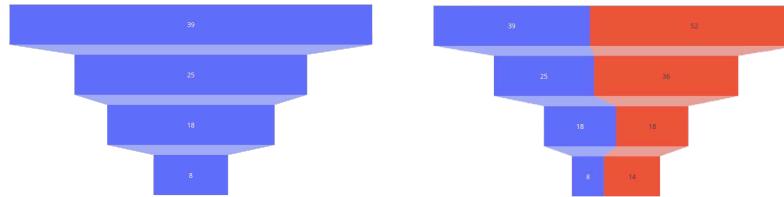
Plotly Express plots can be easily customized:

- Control common parameters like width & height, titles, labeling and colors using built-in function arguments
- Update figure attributes using update methods
- Use **Plotly's theming/templating mechanism** via the *template* function argument
- Set default values for common parameters using *px.defaults*

Funnel

FREQUENCIES

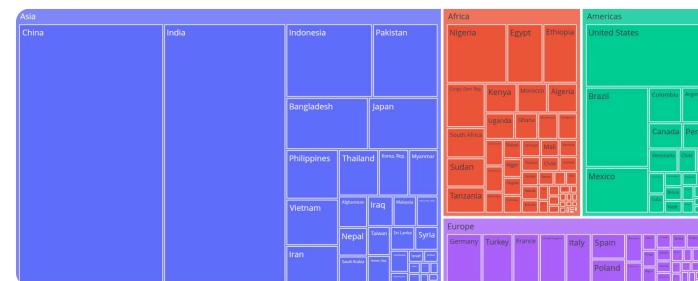
```
s = ['Impressions', 'Clicks', 'Downloads']
a = DataFrame(dict(c=[10, 9, 7], s=s, g='a'))
b = DataFrame(dict(c=[15, 8, 4], s=s, g='b'))
px.funnel(a, x='c', y='s')
px.funnel(concat([a, b]), x='c', y='s', color='g')
```



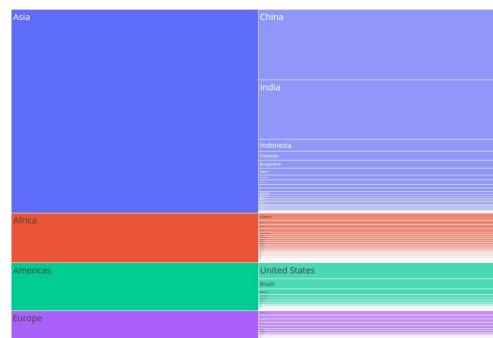
Hierarchy

TWO DISCRETE

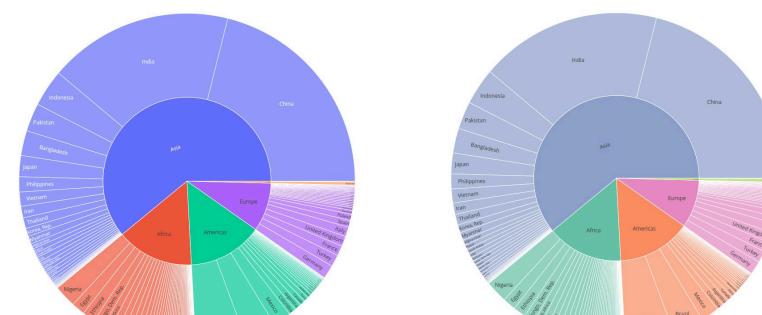
```
df = px.data.gapminder().query('year == 2007')
px.treemap(df, path=['continent', 'country'], \
values='pop')
```



```
px.icicle(df, path=['continent', 'country'], \
values='pop')
```



```
px.sunburst(df, path=['continent', 'country'], \
values='pop')
colors = px.colors.qualitative.Set2
px.sunburst(df, path=['continent', 'country'], \
values='pop', color_discrete_sequence=colors)
```

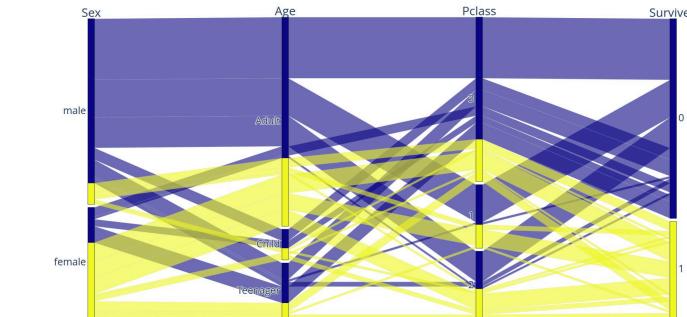


Prototype fast with `px.data` sample datasets, perfect for testing plot settings.

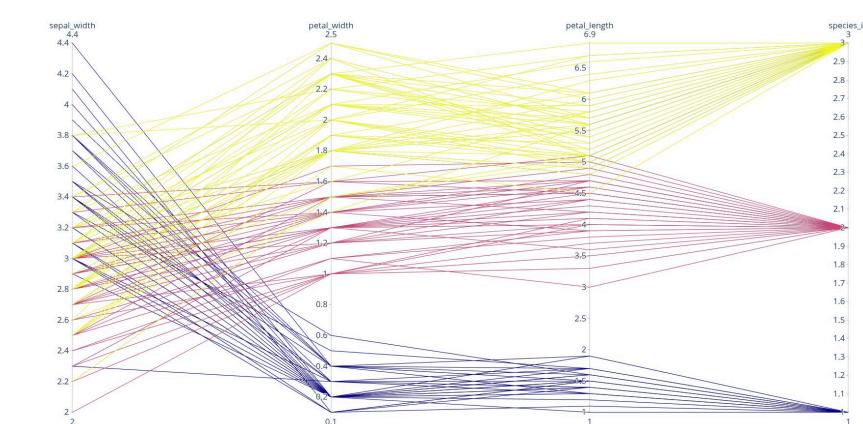
Flow Diagrams

MULTIDIMENSIONAL

```
px.parallel_categories(titanic_df, \
['Sex', 'Age', 'Class', 'Survived'], \
color='Survived')
```



```
df_iris = px.data.iris()
px.parallel_coordinates(df_iris, \
color='species_id')
```



TIPS for readable flow-diagrams plots:

- Limit to 4–6 numeric variables
- Mostly categorical? Use 'parallel_categories' and bin numeric variables
- Order axes logically to reveal patterns
- Tweak colors with `color scales` (`color_continuous_scale`)