



Heatmaps in Python

How to make Heatmaps in Python with Plotly.

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)

The term "heatmap" usually refers to a Cartesian plot with data visualized as colored rectangular tiles, which is the subject of this page. It is also sometimes used to refer to [actual maps with density data displayed as color intensity](https://plotly.com/python/tile-density-heatmaps/) ([/python/tile-density-heatmaps/](https://plotly.com/python/tile-density-heatmaps/)).

Plotly supports two different types of colored-tile heatmaps:

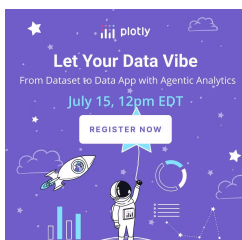
1. **Matrix Heatmaps** accept a 2-dimensional matrix or array of data and visualizes it directly. This type of heatmap is the subject of this page.
2. **Density Heatmaps** accept data as a list and visualizes aggregated quantities like counts or sums of this data. Please refer to the [2D Histogram documentation](https://plotly.com/python/2D-Histogram/) ([/python/2D-Histogram/](https://plotly.com/python/2D-Histogram/)) for this kind of figure.

Heatmaps with Plotly Express

[Plotly Express](https://plotly.com/python/plotly-express/) ([/python/plotly-express/](https://plotly.com/python/plotly-express/)) is the easy-to-use, high-level interface to Plotly, which [operates on a variety of types of data](https://plotly.com/python/px-arguments/) ([/python/px-arguments/](https://plotly.com/python/px-arguments/)) and produces [easy-to-style figures](https://plotly.com/python/styling-plotly-express/) ([/python/styling-plotly-express/](https://plotly.com/python/styling-plotly-express/)). With `px.imshow`, each value of the input array or data frame is represented as a heatmap pixel.

The `px.imshow()` function can be used to display heatmaps (as well as full-color images, as its name suggests). It accepts both array-like objects like lists of lists and numpy or xarray arrays, as well as supported [DataFrame objects](https://plotly.com/python/px-arguments/#supported-dataframes) ([/python/px-arguments/#supported-dataframes](https://plotly.com/python/px-arguments/#supported-dataframes)).

For more examples using `px.imshow`, including examples of faceting and animations, as well as full-color image display, see the [the imshow documentation page](https://plotly.com/python/imshow/) ([/python/imshow/](https://plotly.com/python/imshow/)).

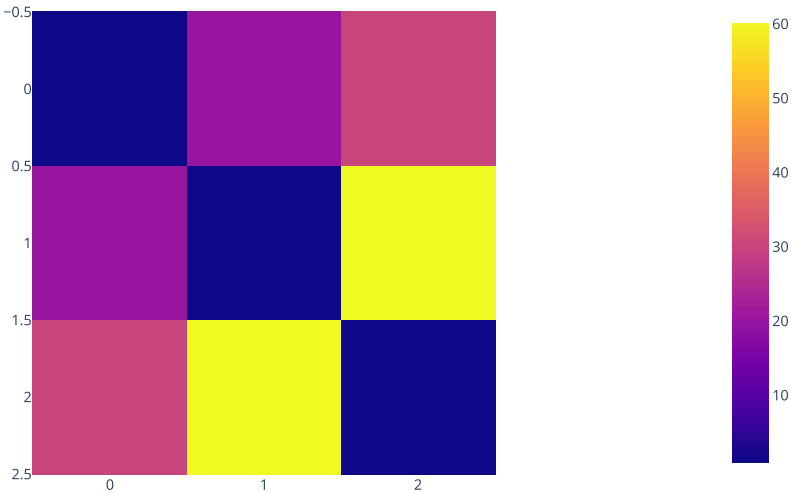


```
import plotly.express as px

fig = px.imshow([[1, 20, 30],
                 [20, 1, 60],
                 [30, 60, 1]])

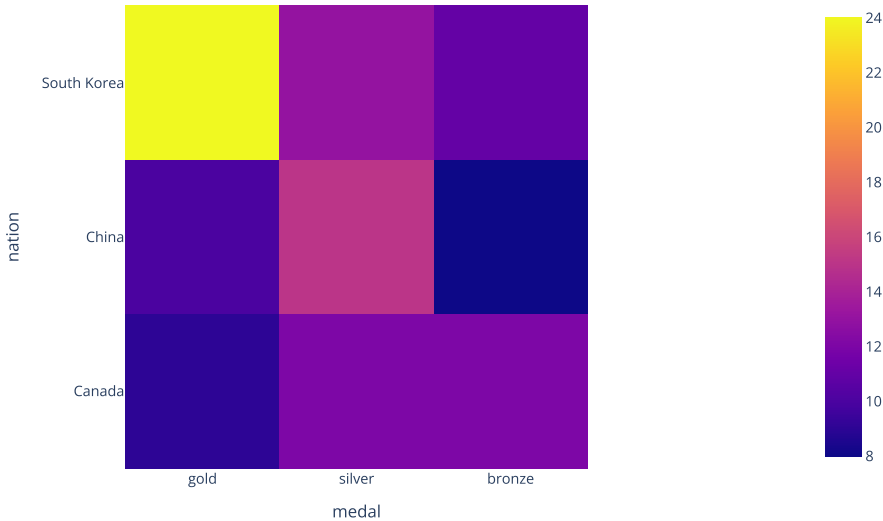
fig.show()
```

a
show
ects
:ls

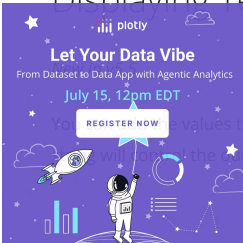


```
import plotly.express as px

df = px.data.medals_wide(indexed=True)
fig = px.imshow(df)
fig.show()
```



Displaying Text on Heatmaps



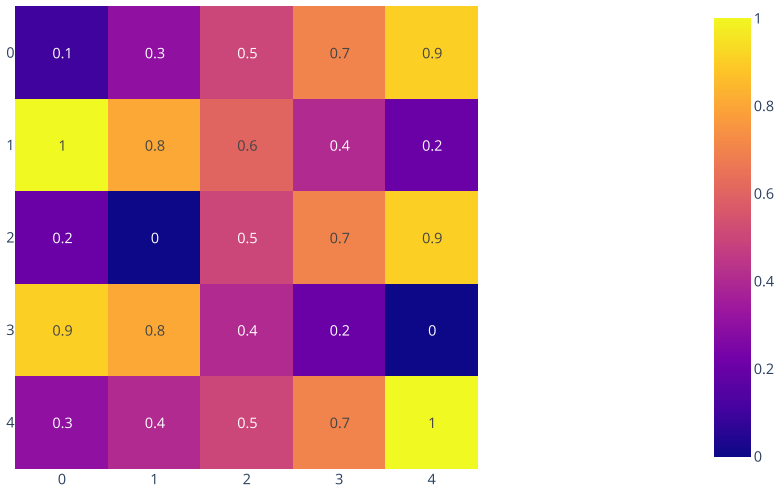
to the figure as text using the `text_auto` argument. Setting it to `True` will display the values on the bars, and setting it to a d3-format formatting

```
import plotly.express as px

z = [[.1, .3, .5, .7, .9],
      [1, .8, .6, .4, .2],
      [.2, 0, .5, .7, .9],
      [.9, .8, .4, .2, 0],
      [.3, .4, .5, .7, 1]]

fig = px.imshow(z, text_auto=True)
fig.show()
```

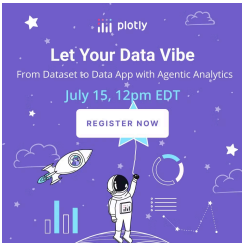
a
:how
ects
:ls



Heatmaps in Dash

Dash (<https://plotly.com/dash/>) is the best way to build analytical apps in Python using Plotly figures. To run the app below, run pip install dash, click "Download" to get the code and run python app.py.

Get started with [the official Dash docs](https://dash.plotly.com/installation) (<https://dash.plotly.com/installation>) and **learn how to effortlessly style** (<https://plotly.com/dash/design-kit/>) & **deploy** (<https://plotly.com/dash/app-manager/>) **apps like this with Dash Enterprise** (<https://plotly.com/dash/>).



a
:how
ects
:ls

```
from dash import Dash, dcc, html, Input, Output
import plotly.express as px

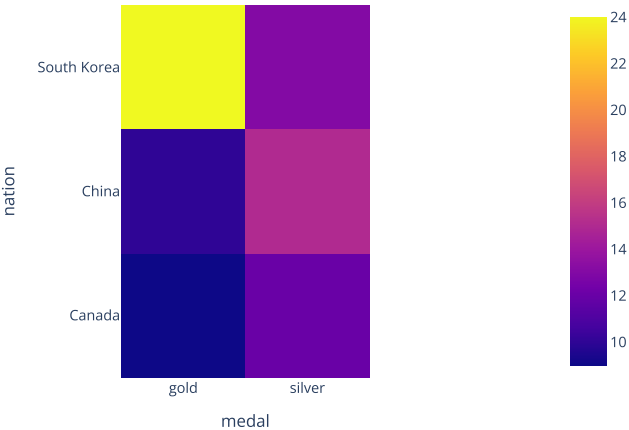
app = Dash(__name__)

app.layout = html.Div([
    html.H4('Olympic medals won by countries'),
    dcc.Graph(id="graph"),
    html.P("Medals included:"),
    dcc.Checklist(
        id='medals',
        options=["gold", "silver", "bronze"],
        value=["gold", "silver"],
    ),
])

@app.callback(
    Output("graph", "figure"),
    Input("medals", "value"))
def filter_heatmap(cols):
    df = px.data.medals_wide(indexed=True) # replace with your own data source
    fig = px.imshow(df[cols])
    return fig
```

DOWNLOAD

Olympic medals won by countries



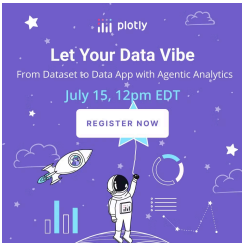
Medals included:

- ☒gold
- ☒silver
- ☐bronze

Sign up for Dash Club → Free cheat sheets plus updates from Chris Parmer and Adam Schroeder delivered to your inbox every two months. Includes tips and tricks, community apps, and deep dives into the Dash architecture. [Join now \(https://go.plotly.com/dash-club?utm_source=Dash+Club+2022&utm_medium=graphing_libraries&utm_content=inline\)](https://go.plotly.com/dash-club?utm_source=Dash+Club+2022&utm_medium=graphing_libraries&utm_content=inline).

Controlling Aspect Ratio

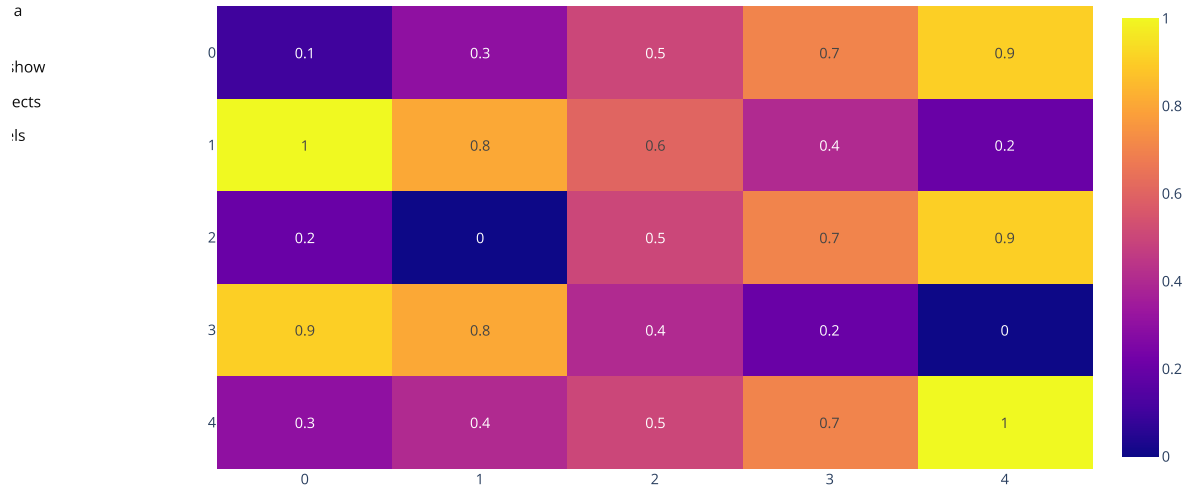
By default, `px.imshow()` produces heatmaps with square tiles, but setting the `aspect` argument to "auto" will instead fill the plotting area with the heatmap, using non-square tiles.



```
import plotly.express as px

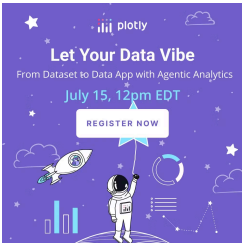
z = [[.1, .3, .5, .7, .9],
      [1, .8, .6, .4, .2],
      [.2, 0, .5, .7, .9],
      [.9, .8, .4, .2, 0],
      [.3, .4, .5, .7, 1]]

fig = px.imshow(z, text_auto=True, aspect="auto")
fig.show()
```



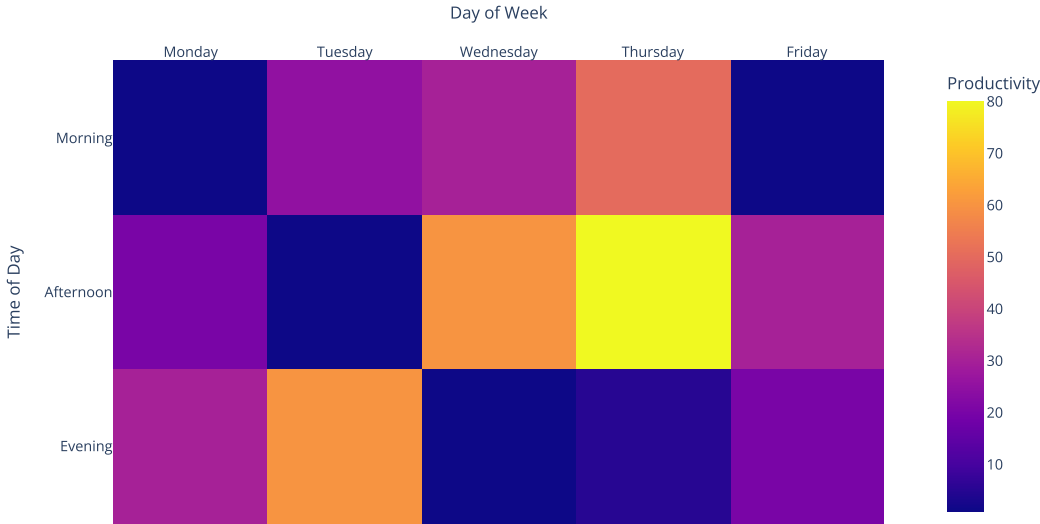
Customizing the axes and labels on a heatmap

You can use the x, y and labels arguments to customize the display of a heatmap, and use .update_xaxes() to move the x axis tick labels to the top:



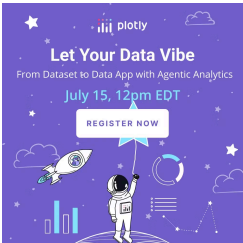
```
import plotly.express as px
data=[[1, 25, 30, 50, 1], [20, 1, 60, 80, 30], [30, 60, 1, 5, 20]]
fig = px.imshow(data,
                 labels=dict(x="Day of Week", y="Time of Day", color="Productivity"),
                 x=['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday'],
                 y=['Morning', 'Afternoon', 'Evening'])
fig.update_xaxes(side="top")
fig.show()
```

a
:how
ects
:ls



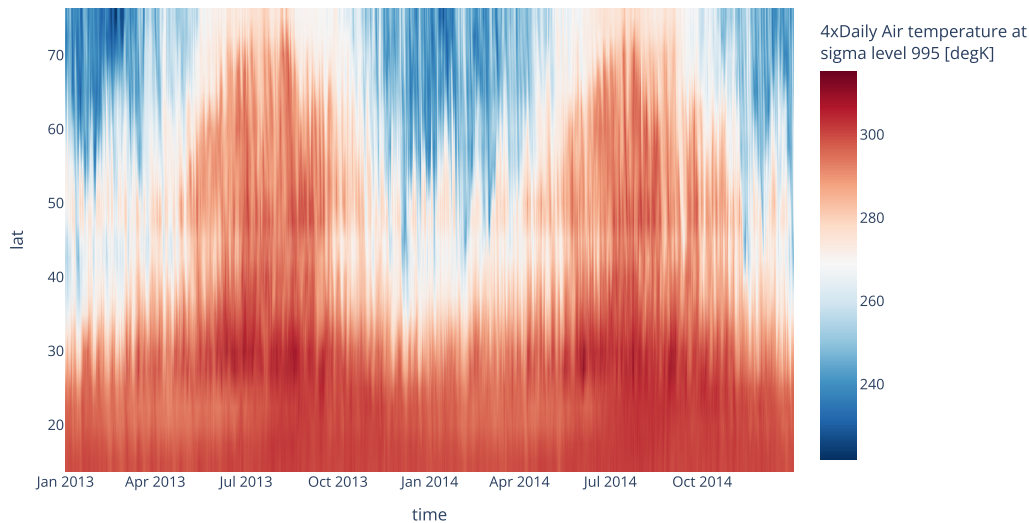
Display an xarray image with px.imshow

xarrays (<http://xarray.pydata.org/en/stable/>) are labeled arrays (with labeled axes and coordinates). If you pass an xarray image to px.imshow, its axes labels and coordinates will be used for axis titles. If you don't want this behavior, you can pass img.values which is a NumPy array if img is an xarray. Alternatively, you can override axis titles hover labels and colorbar title using the labels attribute, as above.



```
import plotly.express as px
import xarray as xr
# Load xarray from dataset included in the xarray tutorial
airtemps = xr.tutorial.open_dataset('air_temperature').air.sel(lon=250.0)
fig = px.imshow(airtemps.T, color_continuous_scale='RdBu_r', origin='lower')
fig.show()
```

a
:how
ects
:ls



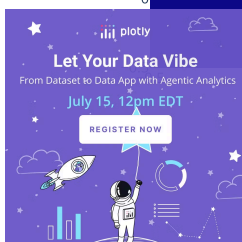
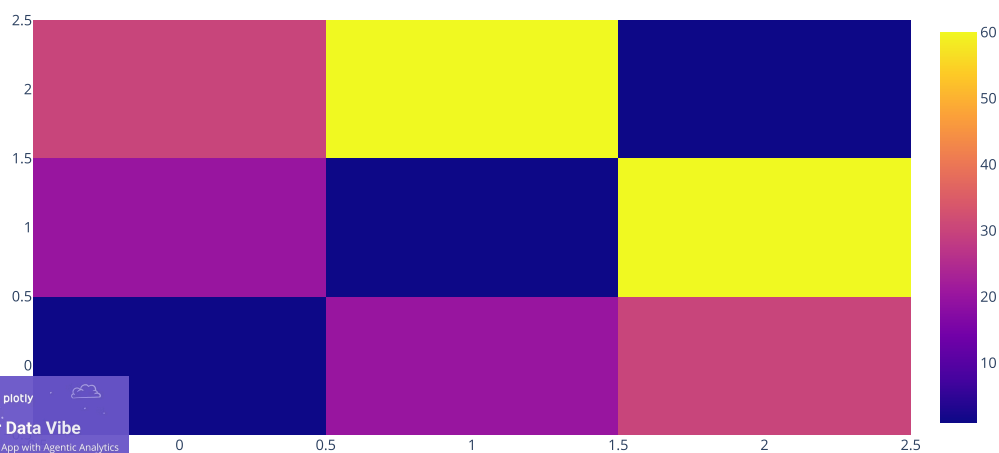
Basic Heatmap with plotly.graph_objects

If Plotly Express does not provide a good starting point, it is also possible to use [the more generic go.Heatmap class from plotly.graph_objects \(/python/graph-objects/\)](#).

```
import plotly.graph_objects as go

fig = go.Figure(data=go.Heatmap(
    z=[[1, 20, 30],
        [20, 1, 60],
        [30, 60, 1]]))

fig.show()
```



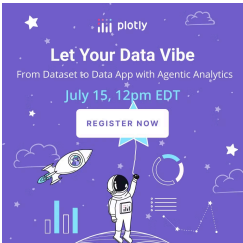
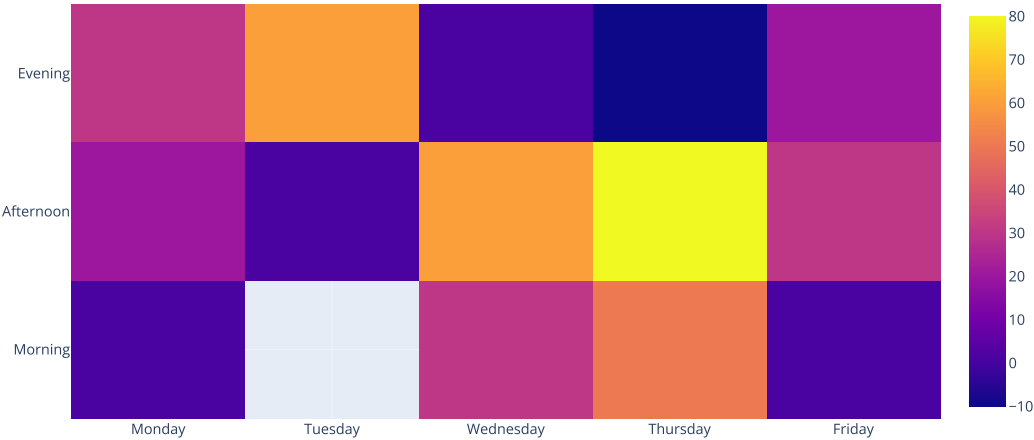
Heatmap with Categorical Axis Labels

In this example we also show how to ignore [hovertext](https://plotly.com/python/hover-text-and-formatting/) when we have missing values in the data by setting the [hoverongaps](https://plotly.com/python/reference/heatmap/#heatmap-hoverongaps) to False.

```
import plotly.graph_objects as go

fig = go.Figure(data=go.Heatmap(
    z=[[1, None, 30, 50, 1], [20, 1, 60, 80, 30], [30, 60, 1, -10, 20]],
    x=['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday'],
    y=['Morning', 'Afternoon', 'Evening'],
    hoverongaps = False))

fig.show()
```



Heatmap with Unequal Block Sizes

```

import plotly.graph_objects as go
import numpy as np

# Build the rectangles as a heatmap
# specify the edges of the heatmap squares
phi = (1 + np.sqrt(5))/2. # golden ratio
xe = [0, 1, 1+(1/(phi**4)), 1+(1/(phi**3)), phi]
ye = [0, 1/(phi**3), 1/phi**3+1/phi**4, 1/(phi**2), 1]

z = [ [13,3,3,5],
      [13,2,1,5],
      [13,10,11,12],
      [13,8,8,8]
    ]

fig = go.Figure(data=go.Heatmap(
    x = np.sort(xe),
    y = np.sort(ye),
    z = z,
    type = 'heatmap',
    colorscale = 'Viridis'))

# Add spiral line plot

def spiral(th):
    a = 1.120529
    b = 0.306349
    r = a*np.exp(-b*th)
    return (r*np.cos(th), r*np.sin(th))

theta = np.linspace(-np.pi/13,4*np.pi,1000); # angle
(x,y) = spiral(theta)

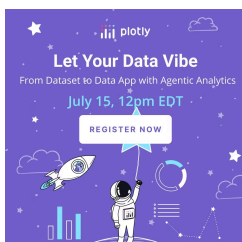
fig.add_trace(go.Scatter(x=-x+x[0], y=y-y[0],
    line=dict(color='white',width=3)))

axis_template = dict(range = [0,1.6], autorange = False,
    showgrid = False, zeroline = False,
    linecolor = 'black', showticklabels = False,
    ticks = '' )

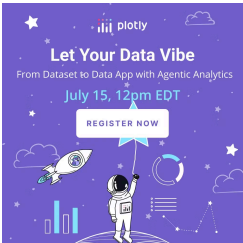
fig.update_layout(margin = dict(t=200,r=200,b=200,l=200),
    xaxis = axis_template,
    yaxis = axis_template,
    showlegend = False,
    width = 700, height = 700,
    autosize = False )

fig.show()

```



a
:how
ects
:ls



Heatmap with Datetime Axis

```
import plotly.graph_objects as go
import datetime
import numpy as np
np.random.seed(1)

programmers = ['Alex', 'Nicole', 'Sara', 'Etienne', 'Chelsea', 'Jody', 'Marianne']

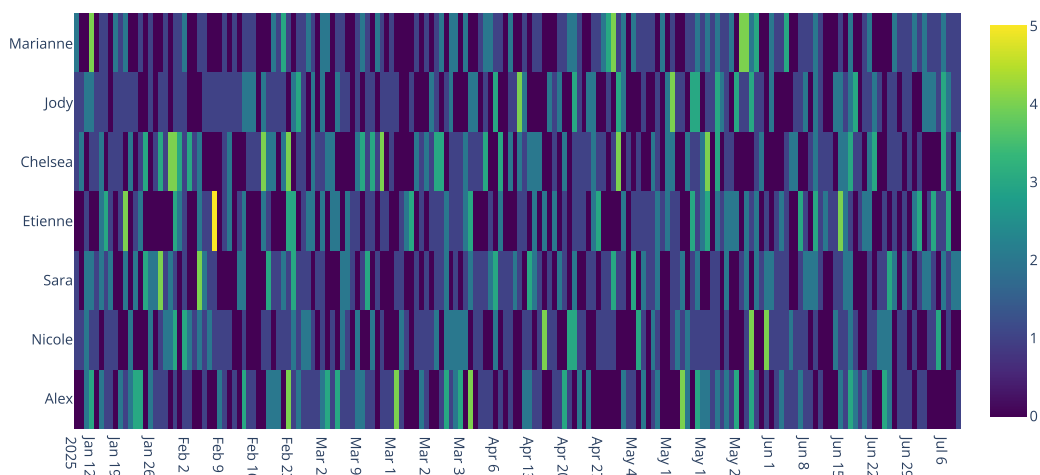
base = datetime.datetime.today()
dates = base - np.arange(180) * datetime.timedelta(days=1)
z = np.random.poisson(size=(len(programmers), len(dates)))

fig = go.Figure(data=go.Heatmap(
    z=z,
    x=dates,
    y=programmers,
    colorscale='Viridis'))

fig.update_layout(
    title=dict(text='GitHub commits per day',
    axis_nticks=36))

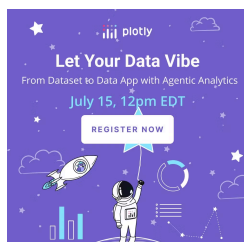
fig.show()
```

GitHub commits per day



Text on Heatmap Points

In this example we add text to heatmap points using `texttemplate`. We use the values from the `text` attribute for the text. We also adjust the font size using `textfont`.

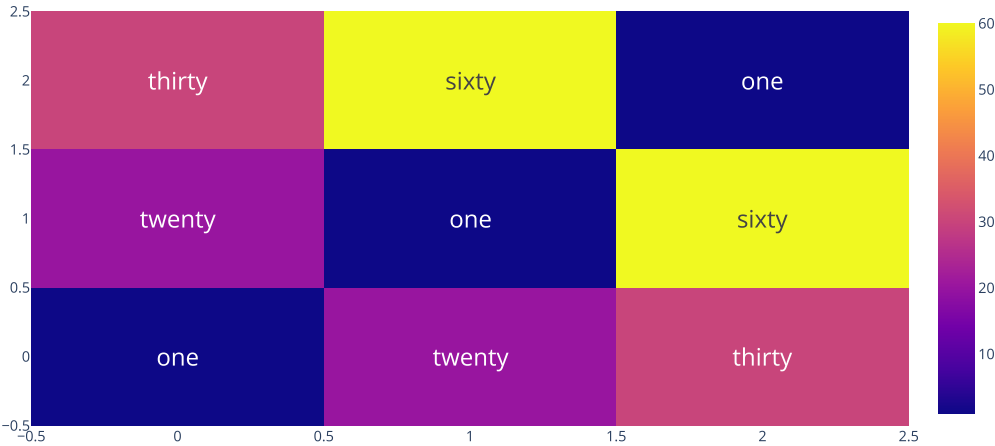


```
import plotly.graph_objects as go

fig = go.Figure(data=go.Heatmap(
    z=[[1, 20, 30],
        [20, 1, 60],
        [30, 60, 1]],
    text=[['one', 'twenty', 'thirty'],
           ['twenty', 'one', 'sixty'],
           ['thirty', 'sixty', 'one']],
    texttemplate="%{text}",
    textfont={"size":20}))

fig.show()
```

a
:how
ects
:ls

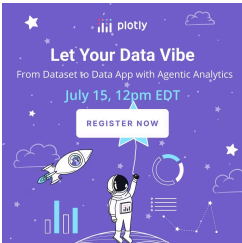


Heatmap and datashader

Arrays of rasterized values build by datashader can be visualized using plotly's heatmaps, as shown in the [plotly and datashader tutorial \(/python/datashader/\)](#).

Reference

See [function reference for px.imshow\(\)](#) (<https://plotly.com/python-api-reference/generated/plotly.express.imshow>) or <https://plotly.com/python/reference/heatmap/> (<https://plotly.com/python/reference/heatmap/>) for more information and chart attribute options!



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
```



Dash your way to interactive web apps.

No JavaScript required!

GET STARTED NOW

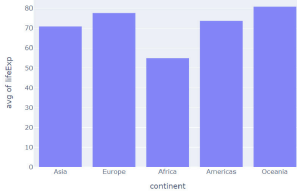
My First App with Data, Graph, and Controls

pop

lifeExp

gdpPerCap

country	pop	continent	lifeExp	gdpPerCap
Afghanistan	31889923	Asia	43.828	974.5883384
Albania	3600523	Europe	76.423	5937.829525999999
Algeria	33333216	Africa	72.381	6223.367465
Angola	12420476	Africa	42.731	4707.231267
Argentina	40301927	Americas	75.32	12779.37964
Australia	20434176	Oceania	81.235	34435.367439999995
Austria	8199783	Europe	79.829	36126.4927
Bahrain	706573	Asia	75.635	29796.04834
Bangladesh	150448339	Asia	64.062	1701.253792
Belgium	10391226	Europe	79.441	33062.04908
Benin	8878314	Africa	56.728	1441.284873
Bolivia	9119152	Americas	65.554	3821.137884



(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/>)

