Python (/python) > Scientific Charts (/python/scientific-charts) > Annotated Heatmaps

Suggest an edit to this page (https://github.com/plotly/plotly.py/edit/doc-prod/doc/python/annotated-heatmap.md)

Annotated Heatmaps in Python

press

otations

How to make Annotated 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)

Annotated Heatmaps with Plotly Express

New in v5.5

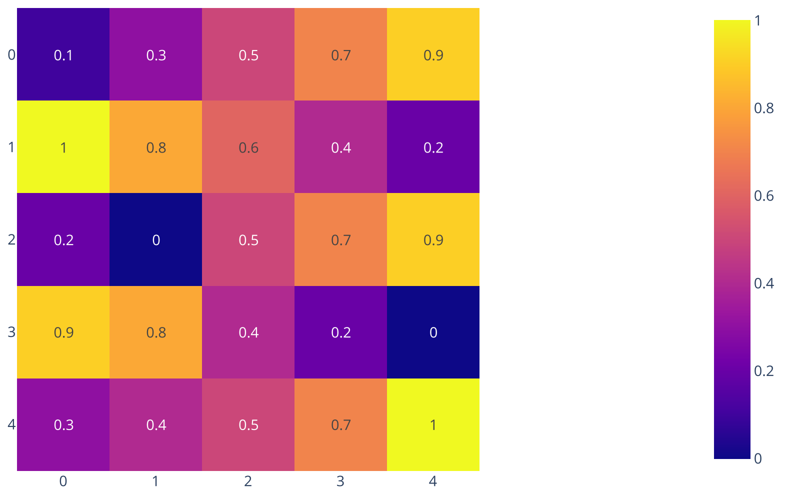
As of version 5.5.0 of plotly, the **recommended way to** display annotated heatmaps is to use px.imshow() (/python/heatmaps/) rather than the now-deprecated create_annotated_heatmap figure factory documented below for historical reasons.

Basic Annotated Heatmap for z-annotations

New in v5.5

After creating a figure with px.imshow, you can add z-annotations with .update_traces(texttemplate="%{z}").

```
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()
```



Deprecated Figure Factory

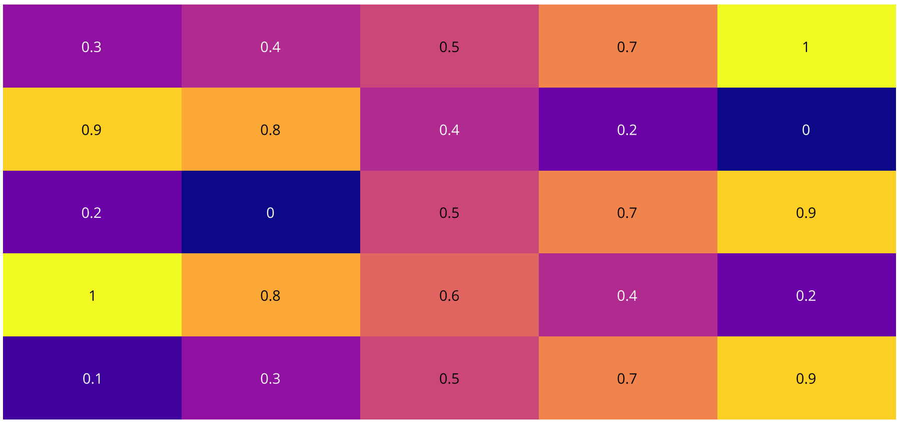
The remaining examples show how to create Annotated Heatmaps with the deprecated create_annotated_heatmap figure factory (/python/figure-factories/).

Simple Annotated Heatmap

press

otations

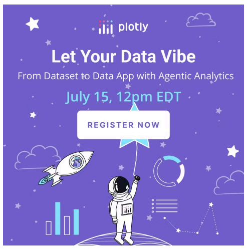
```
import plotly.figure_factory as ff

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 = ff.create_annotated_heatmap(z)
fig.show()
```



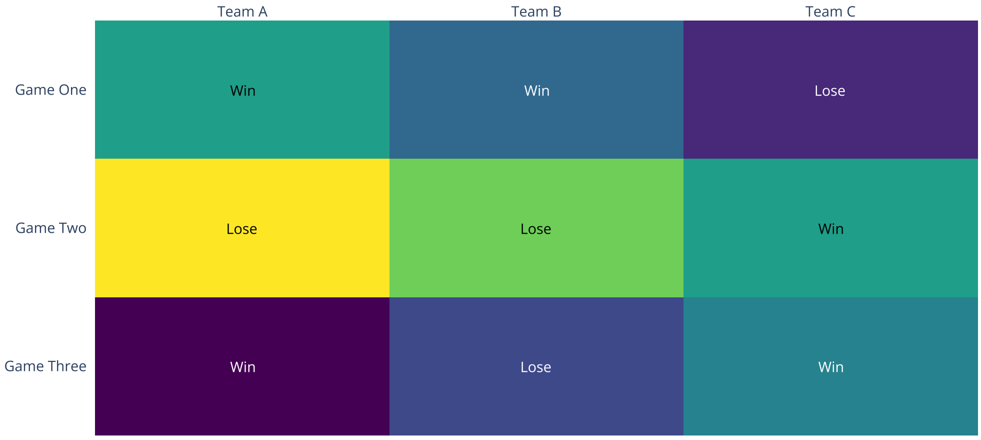
Custom Text and X & Y Labels

set annotation_text to a matrix with the same dimensions as z

> WARNING: this legacy figure factory requires the y array to be provided in reverse order, and will map the z_text to the z values in reverse order. **The use of the** px.imshow() **version below is highly recommended**

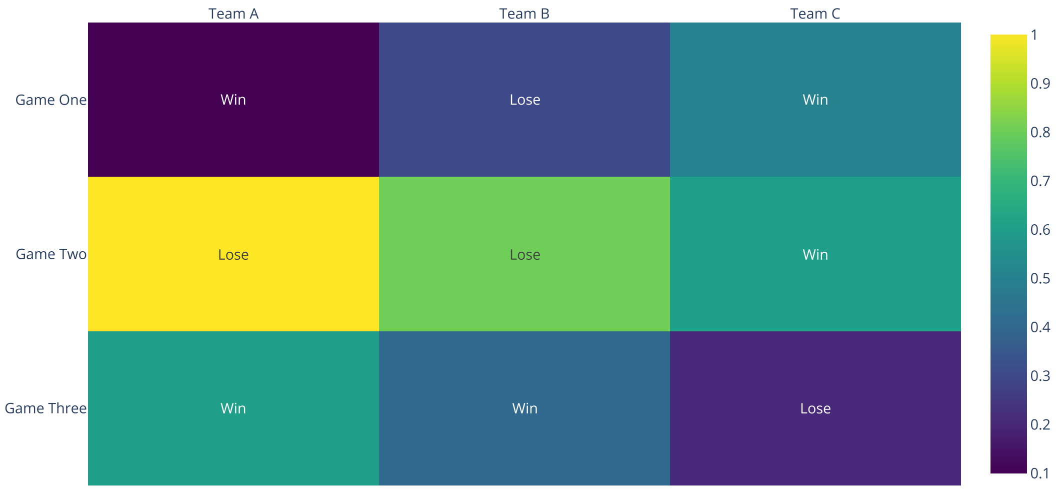


```
import plotly.figure_factory as ff

z = [[.1, .3, .5],
     [1.0, .8, .6],
     [.6, .4, .2]]

x = ['Team A', 'Team B', 'Team C']
y = ['Game Three', 'Game Two', 'Game One']

z_text = [['Win', 'Lose', 'Win'],
          ['Lose', 'Lose', 'Win'],
          ['Win', 'Win', 'Lose']]

fig = ff.create_annotated_heatmap(z, x=x, y=y, annotation_text=z_text, colorscale='Viridis')
fig.show()
```

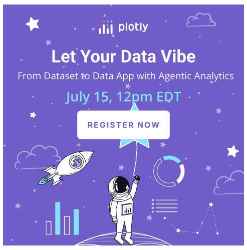


Here is the same figure using px.imshow()

```
import plotly.express as px

x = ['Team A', 'Team B', 'Team C']
y = ['Game One', 'Game Two', 'Game Three']

z = [[.1, .3, .5],
     [1.0, .8, .6],
     [.6, .4, .2]]

z_text = [['Win', 'Lose', 'Win'],
          ['Lose', 'Lose', 'Win'],
          ['Win', 'Win', 'Lose']]

fig = px.imshow(z, x=x, y=y, color_continuous_scale='Viridis', aspect="auto")
fig.update_traces(text=z_text, texttemplate="%{text}")
fig.update_xaxes(side="top")
fig.show()
```



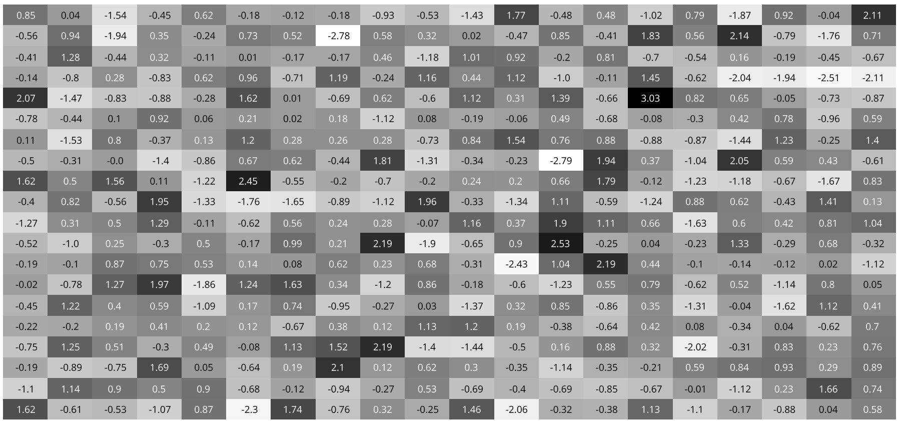
Annotated Heatmap with numpy



```
import plotly.figure_factory as ff
import numpy as np
np.random.seed(1)

z = np.random.randn(20, 20)
z_text = np.around(z, decimals=2) # Only show rounded value (full value on hover)

fig = ff.create_annotated_heatmap(z, annotation_text=z_text, colorscale='Greys',
                                  hoverinfo='z')

# Make text size smaller
for i in range(len(fig.layout.annotations)):
    fig.layout.annotations[i].font.size = 8

fig.show()
```

ɔress

otations



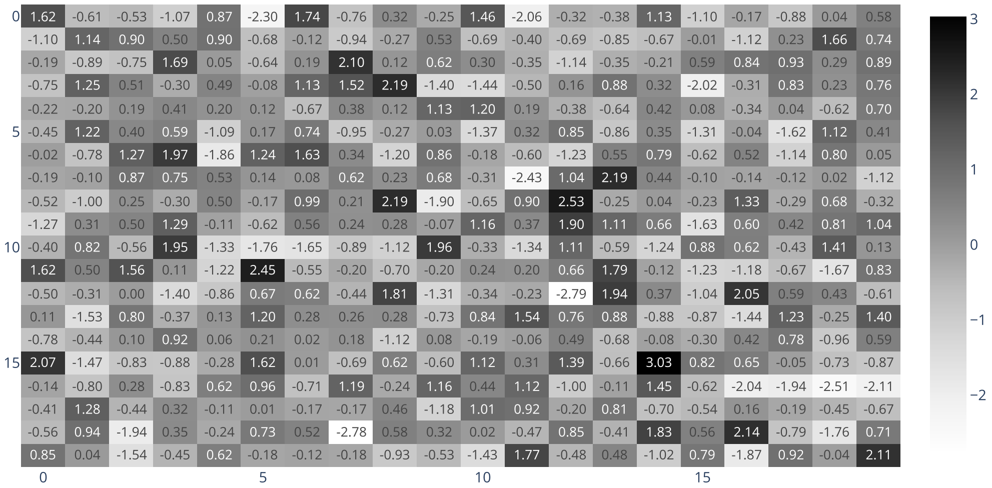
0.85	0.04	-1.54	-0.45	0.62	-0.18	-0.12	-0.18	-0.93	-0.53	-1.43	1.77	-0.48	0.48	-1.02	0.79	-1.87	0.92	-0.04	2.11
-0.56	0.94	-1.94	0.35	-0.24	0.73	0.52	-2.78	0.58	0.32	0.02	-0.47	0.85	-0.41	1.83	0.56	2.14	-0.79	-1.76	0.71
-0.41	1.28	-0.44	0.32	-0.11	0.01	-0.17	-0.17	0.46	-1.18	1.01	0.92	-0.2	0.81	-0.7	-0.54	0.16	-0.19	-0.45	-0.67
-0.14	-0.8	0.28	-0.83	0.62	0.96	-0.71	1.19	-0.24	1.16	0.44	1.12	-1.0	-0.11	1.45	-0.62	-2.04	-1.94	-2.51	-2.11
2.07	-1.47	-0.83	-0.88	-0.28	1.62	0.01	-0.69	0.62	-0.6	1.12	0.31	1.39	-0.66	3.03	0.82	0.65	-0.05	-0.73	-0.87
-0.78	-0.44	0.1	0.92	0.06	0.21	0.02	0.18	-1.12	0.08	-0.19	-0.06	0.49	-0.68	-0.08	-0.3	0.42	0.78	-0.96	0.59
0.11	-1.53	0.8	-0.37	0.13	1.2	0.28	0.26	0.28	-0.73	0.84	1.54	0.76	0.88	-0.88	-0.87	-1.44	1.23	-0.25	1.4
-0.5	-0.31	-0.0	-1.4	-0.86	0.67	0.62	-0.44	1.81	-1.31	-0.34	-0.23	-2.79	1.94	0.37	-1.04	2.05	0.59	0.43	-0.61
1.62	0.5	1.56	0.11	-1.22	2.45	-0.55	-0.2	-0.7	-0.2	0.24	0.2	0.66	1.79	-0.12	-1.23	-1.18	-0.67	-1.67	0.83
-0.4	0.82	-0.56	1.95	-1.33	-1.76	-1.65	-0.89	-1.12	1.96	-0.33	-1.34	1.11	-0.59	-1.24	0.88	0.62	-0.43	1.41	0.13
-1.27	0.31	0.5	1.29	-0.11	-0.62	0.56	0.24	0.28	-0.07	1.16	0.37	1.9	1.11	0.66	-1.63	0.6	0.42	0.81	1.04
-0.52	-1.0	0.25	-0.3	0.5	-0.17	0.99	0.21	2.19	-1.9	-0.65	0.9	2.53	-0.25	0.04	-0.23	1.33	-0.29	0.68	-0.32
-0.19	-0.1	0.87	0.75	0.53	0.14	0.08	0.62	0.23	0.68	-0.31	-2.43	1.04	2.19	0.44	-0.1	-0.14	-0.12	0.02	-1.12
-0.02	-0.78	1.27	1.97	-1.86	1.24	1.63	0.34	-1.2	0.86	-0.18	-0.6	-1.23	0.55	0.79	-0.62	0.52	-1.14	0.8	0.05
-0.45	1.22	0.4	0.59	-1.09	0.17	0.74	-0.95	-0.27	0.03	-1.37	0.32	0.85	-0.86	0.35	-1.31	-0.04	-1.62	1.12	0.41
-0.22	-0.2	0.19	0.41	0.2	0.12	-0.67	0.38	0.12	1.13	1.2	0.19	-0.38	-0.64	0.42	0.08	-0.34	0.04	-0.62	0.7
-0.75	1.25	0.51	-0.3	0.49	-0.08	1.13	1.52	2.19	-1.4	-1.44	-0.5	0.16	0.88	0.32	-2.02	-0.31	0.83	0.23	0.76
-0.19	-0.89	-0.75	1.69	0.05	-0.64	0.19	2.1	0.12	0.62	0.3	-0.35	-1.14	-0.35	-0.21	0.59	0.84	0.93	0.29	0.89
-1.1	1.14	0.9	0.5	0.9	-0.68	-0.12	-0.94	-0.27	0.53	-0.69	-0.4	-0.69	-0.85	-0.67	-0.01	-1.12	0.23	1.66	0.74
1.62	-0.61	-0.53	-1.07	0.87	-2.3	1.74	-0.76	0.32	-0.25	1.46	-2.06	-0.32	-0.38	1.13	-1.1	-0.17	-0.88	0.04	0.58

Here is the same figure using px.imshow()

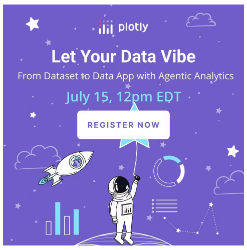
```
import plotly.express as px
import numpy as np
np.random.seed(1)

z = np.random.randn(20, 20)

fig = px.imshow(z, text_auto=".2f", color_continuous_scale='Greys', aspect="auto")
fig.show()
```

ɔress

otations



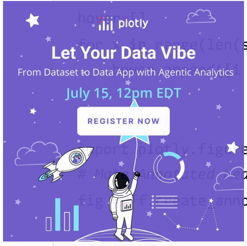
Here is a fairly contrived example showing how one can display a periodic table with custom text and hover using ff.create_annotated_heatmap() (scroll below to see the px.imshow() equivalent).



press

otations

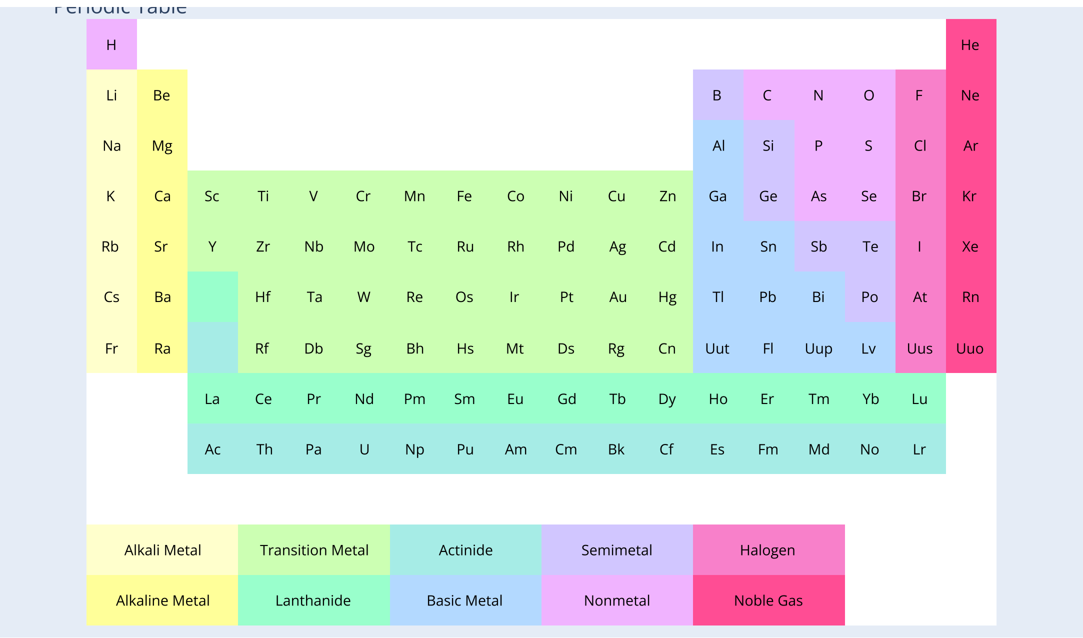
```
# Periodic Table Data
symbol = [['H', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', 'He'],
         ['Li', 'Be', '', '', '', '', '', '', '', '', '', '', 'B', 'C', 'N', 'O', 'F', 'Ne'],
         ['Na', 'Mg', '', '', '', '', '', '', '', '', '', '', 'Al', 'Si', 'P', 'S', 'Cl', 'Ar'],
         ['K', 'Ca', 'Sc', 'Ti', 'V', 'Cr', 'Mn', 'Fe', 'Co', 'Ni', 'Cu', 'Zn', 'Ga', 'Ge', 'As', 'Se', 'Br', 'Kr'],
         ['Rb ', 'Sr', 'Y', 'Zr', 'Nb', 'Mo', 'Tc', 'Ru', 'Rh', 'Pd', 'Ag', 'Cd', 'In', 'Sn', 'Sb', 'Te', 'I', 'Xe' ],
         ['Cs', 'Ba', '', 'Hf', 'Ta', 'W', 'Re', 'Os', 'Ir', 'Pt', 'Au', 'Hg', 'Tl', 'Pb', 'Bi', 'Po', 'At', 'Rn' ],
         ['Fr', 'Ra', '', 'Rf', 'Db', 'Sg', 'Bh', 'Hs', 'Mt', 'Ds', 'Rg', 'Cn', 'Uut', 'Fl', 'Uup', 'Lv', 'Uus', 'Uuo'],
         ['', '', 'La', 'Ce', 'Pr', 'Nd', 'Pm', 'Sm', 'Eu', 'Gd', 'Tb', 'Dy', 'Ho', 'Er', 'Tm', 'Yb', 'Lu', ''],
         ['', '', 'Ac', 'Th', 'Pa', 'U', 'Np', 'Pu', 'Am', 'Cm', 'Bk', 'Cf', 'Es', 'Fm', 'Md', 'No', 'Lr', '' ],
         ['', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', ''],
         ['', 'Alkali Metal', '', '', 'Transition Metal', '', '', 'Actinide', '', '', 'Semimetal', '', '', 'Halogen', '', '', '', ''],
         ['', 'Alkaline Metal', '', '', 'Lanthanide', '', '', 'Basic Metal', '', '', 'Nonmetal', '', '', 'Noble Gas', '', '', '', '']]

element = [['Hydrogen', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', 'Helium'],
           ['Lithium', 'Beryllium', '', '', '', '', '', '', '', '', '', '', 'Boron', 'Carbon', 'Nitrogen', 'Oxygen', 'Fluorine', 'Neon'],
           ['Sodium', 'Magnesium', '', '', '', '', '', '', '', '', '', '', 'Aluminium', 'Silicon', 'Phosphorus', 'Sulfur', 'Chlorine', 'Argon'],
           ['Potassium', 'Calcium', 'Scandium', 'Titanium', 'Vanadium', 'Chromium',  'Manganese', 'Iron', 'Cobalt', 'Nickel', 'Copper', 'Zinc', 'Ga
llium', 'Germanium', 'Arsenic', 'Selenium', 'Bromine', 'Krypton'],
           ['Rubidium', 'Strontium', 'Yttrium', 'Zirconium', 'Niobium', 'Molybdenum', 'Technetium', 'Ruthenium', 'Rhodium', 'Palladium', 'Silver',
'Cadmium', 'Indium', 'Tin', 'Antimony', 'Tellurium', 'Iodine', 'Xenon'],
           ['Cesium', 'Barium', '',  'Hafnium', 'Tantalum', 'Tungsten', 'Rhenium', 'Osmium', 'Iridium', 'Platinum', 'Gold', 'Mercury', 'Thallium',
'Lead', 'Bismuth', 'Polonium', 'Astatine', 'Radon'],
           ['Francium', 'Radium', '', 'Rutherfordium','Dubnium','Seaborgium','Bohrium','Hassium','Meitnerium','Darmstadtium','Roentgenium','Coperni
cium','Ununtrium','Ununquadium','Ununpentium','Ununhexium','Ununseptium','Ununoctium'],
           ['', '',  'Lanthanum', 'Cerium', 'Praseodymium', 'Neodymium', 'Promethium', 'Samarium', 'Europium', 'Gadolinium', 'Terbium', 'Dysprosiu
m', 'Holmium', 'Erbium', 'Thulium', 'Ytterbium', 'Lutetium', ''],
           ['', '', 'Actinium', 'Thorium', 'Protactinium', 'Uranium', 'Neptunium', 'Plutonium', 'Americium', 'Curium', 'Berkelium', 'Californium',
'Einsteinium','Fermium' ,'Mendelevium', 'Nobelium', 'Lawrencium', '' ],
           ['', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', ''],
           ['', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', ''],
           ['', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '']]

atomic_mass = [[ 1.00794, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0,  4.002602],
     [ 6.941, 9.012182, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0,  10.811, 12.0107, 14.0067, 15.9994, 18.9984032, 20.1797],
     [ 22.98976928, 24.3050, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0,  26.9815386, 28.0855, 30.973762, 32.065, 35.453, 39.948],
     [ 39.0983, 40.078, 44.955912, 47.867, 50.9415, 51.9961, 54.938045, 55.845, 58.933195, 58.6934, 63.546, 65.38, 69.723, 72.64, 74.92160, 78.96,
79.904, 83.798],
     [ 85.4678, 87.62, 88.90585, 91.224, 92.90638, 95.96, 98, 101.07, 102.90550, 106.42, 107.8682, 112.411, 114.818, 118.710, 121.760, 127.60, 126.
90447, 131.293],
     [ 132.9054519, 137.327, .0, 178.49, 180.94788, 183.84, 186.207, 190.23, 192.217, 195.084, 196.966569, 200.59, 204.3833, 207.2, 208.98040, 209,
210, 222],
     [223, 226, .0, 267, 268, 271, 272, 270, 276, 281, 280, 285, 284, 289, 288, 293, 'unknown', 294],
     [.0, .0, 138.90547, 140.116, 140.90765, 144.242, 145, 150.36, 151.964, 157.25, 158.92535, 162.500, 164.93032, 167.259, 168.93421, 173.054, 17
4.9668, .0],
     [.0, .0, 227, 232.03806, 231.03588, 238.02891, 237, 244, 243, 247, 247, 251, 252, 257, 258, 259, 262, .0],
     [.0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0],
     [.0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0],
     [.0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0, .0]]

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

# Set Colorscale
colorscale=[[0.0, 'rgb(255,255,255)'], [.2, 'rgb(255, 255, 153)'],
            [.4, 'rgb(153, 255, 204)'], [.6, 'rgb(179, 217, 255)'],
            [.8, 'rgb(240, 179, 255)'],[1.0, 'rgb(255, 77, 148)']]

# Display element name and atomic mass on hover
hover=[]
for x in range(len(symbol)):
    hover.append([i + '<br>' + 'Atomic Mass: ' + str(j) if i else ''
                      for i, j in zip(element[x], atomic_mass[x])])

import plotly.figure_factory as ff
# Make Annotated Heatmap
fig = ff.create_annotated_heatmap(color[::-1], annotation_text=symbol[::-1], text=hover[::-1],
                                 colorscale=colorscale, font_colors=['black'], hoverinfo='text')
```



```
fig.update_layout(
     title_text='Periodic Table',
     margin=dict(l=10, r=10, t=10, b=10, pad=10),
     xaxis=dict(zeroline=False, showgrid=False),
     yaxis=dict(zeroline=False, showgrid=False, scaleanchor="x"),
)
fig.show()
```

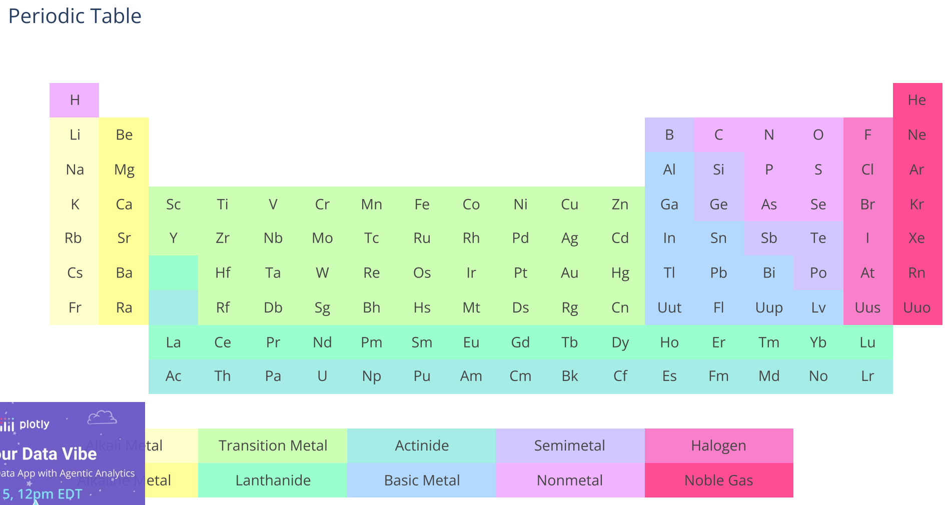
ɔress

otations



Here is the same output using px.imshow() with much less array manipulation:

```
import plotly.express as px
import numpy as np

fig = px.imshow(color, color_continuous_scale=colorscale, aspect="auto",
               title='Periodic Table')
fig.update_traces(
    text=symbol, texttemplate="%{text}", textfont_size=12,
    customdata=np.moveaxis([element, atomic_mass], 0,-1),
    hovertemplate="%{customdata[0]}<br>Atomic Mass: %{customdata[1]:.2f}<extra></extra>"
)
fig.update_xaxes(visible=False)
fig.update_yaxes(visible=False)
fig.update_coloraxes(showscale=False)
fig.show()
```



Reference

For more info on Plotly heatmaps, see: https://plotly.com/python/reference/heatmap/ (https://plotly.com/python/reference/heatmap/).
For more info on using colorscales with Plotly see: https://plotly.com/python/heatmap-and-contour-colorscales/ (https://plotly.com/python/heatmap-and-contour-colorscales/)
For more info on ff.create_annotated_heatmap(), see the full function reference (https://plotly.com/python-api-reference/generated/plotly.figure_factory.create_annotated_heatmap.html#plotly.figure_factory.create_annotated_heatmap)

ɔress
otations

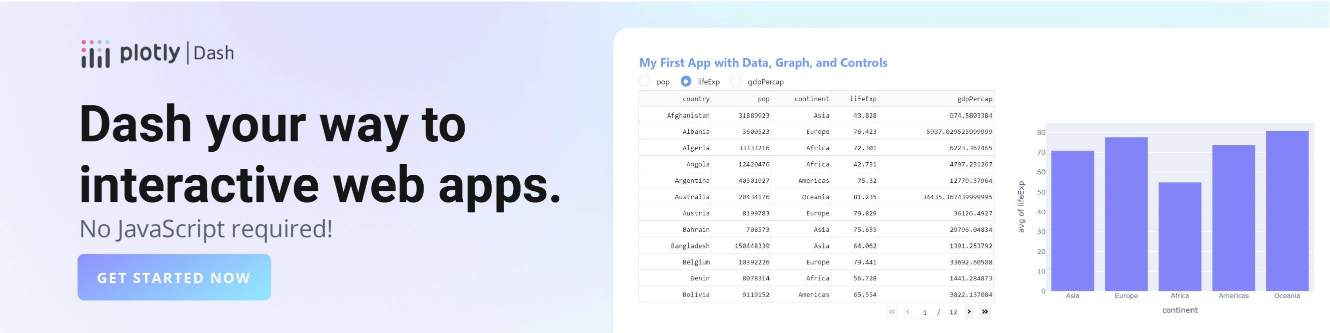
What About Dash?

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
```



My First App with Data, Graph, and Controls

○ pop ● lifeExp ○ gdpPercap

country	pop	continent	lifeExp	gdpPercap
Afghanistan	31889923	Asia	43.828	974.5803384
Albania	3600523	Europe	76.423	5937.029525999999
Algeria	33333216	Africa	72.301	6223.367465
Angola	12420476	Africa	42.731	4797.231267
Argentina	40301927	Americas	75.32	12779.37964
Australia	20434176	Oceania	81.235	34435.367439999995
Austria	8199783	Europe	79.829	36126.4927
Bahrain	708573	Asia	75.635	29796.04834
Bangladesh	150448339	Asia	64.062	1391.253792
Belgium	10392226	Europe	79.441	33692.60508
Benin	8078314	Africa	56.728	1441.284873
Bolivia	9119152	Americas	65.554	3822.137084

« ‹ 1 / 12 › »

(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)

Products

Dash (https://plotly.com/dash/)
Consulting and Training (https://plotly.com/consulting-and-oem/)

Pricing

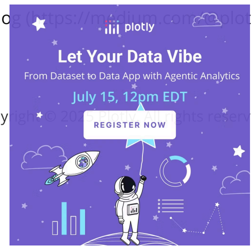
Enterprise Pricing (https://plotly.com/get-pricing/)

About Us

Careers (https://plotly.com/careers)
Resources (https://plotly.com/resources/)
Blog (https://medium.com/@plotlygraphs)

Support

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



Terms of Service (https://community.plotly.com/tos) Privacy Policy (https://plotly.com/privacy/)