2023-07-07_GrammarOfGraphics_002-MorePlotly

July 11, 2023

1 Some observations on PlotLy

While the API of Plotly may not follow the same formal structure as ggplot, it still adheres to a very clear grammar that allows fast high-level figure manipulation. The clearest sign for the existence of this grammar is that any plotly figure can always be serialized as a JSON object, which is in fact how the python front end sends the figure to the JavaScript rendering library. We will briefly look at this grammar and how it is reflected in the PlotLy API to get a more general impression of "structured" figure generation beyond the sole example of ggplot2 / plotnine.

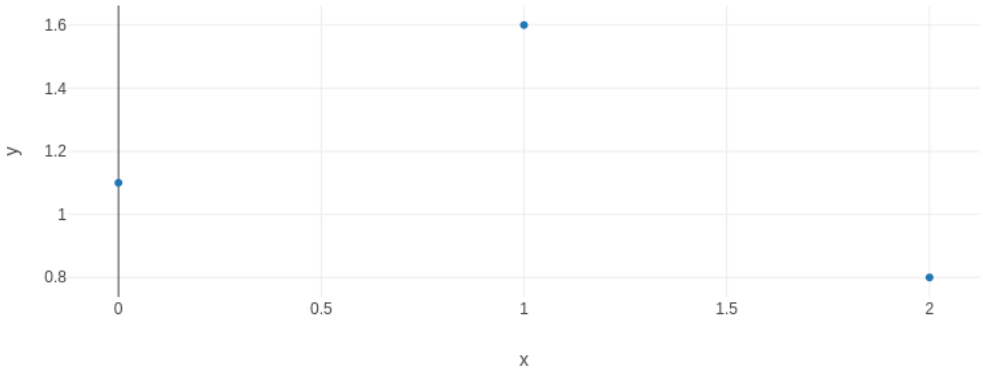
[1]:
```
# basic libraries
import numpy as np
import scipy
import pandas as pd

# plotly
import plotly.express as px
import plotly.graph_objects as go
import plotly.io
```

1.1 A very simple example and the JSON rep

[2]:
```
data=pd.DataFrame({"x":[0.,1.,2.],"y":[1.1,1.6,0.8],"z":["a","a","b"]})
```

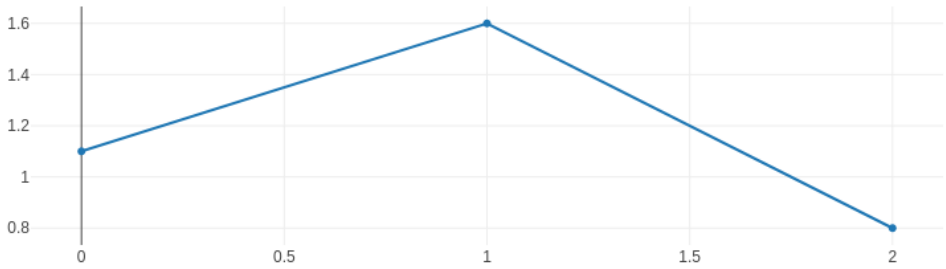
[6]:
```
fig=px.scatter(data,"x","y",template="none")
fig.show()
```



[7]:
```
# the following gives the serialized json object
# as we can see, it still contains some "clutter" from the px interface
plotly.io.to_json(fig)
```

[7]:
```
'{"data":[{"hovertemplate":"x=%{x}<br>y=%{y}<extra></extra>","legendgroup":"","m
arker":{"color":"#1F77B4","symbol":"circle"},"mode":"markers","name":"","orienta
tion":"v","showlegend":false,"x":[0.0,1.0,2.0],"xaxis":"x","y":[1.1,1.6,0.8],"ya
xis":"y","type":"scatter"}],"layout":{"template":{"data":{"scatter":[{"type":"sc
atter"}]}},"xaxis":{"anchor":"y","domain":[0.0,1.0],"title":{"text":"x"}},"yaxis
":{"anchor":"x","domain":[0.0,1.0],"title":{"text":"y"}},"legend":{"tracegroupga
p":0},"margin":{"t":60}}}'
```

[10]:
```
# this generates more or less the most bare-bones figure representation
fig = go.Figure()
fig.add_trace(go.Scatter(x=data["x"],y=data["y"]))
fig.update_layout(template="none")
fig.show()
plotly.io.to_json(fig)
```



[10]:
```
'{"data":[{"x":[0.0,1.0,2.0],"y":[1.1,1.6,0.8],"type":"scatter"}],"layout":{"tem
plate":{"data":{"scatter":[{"type":"scatter"}]}}}}'
```

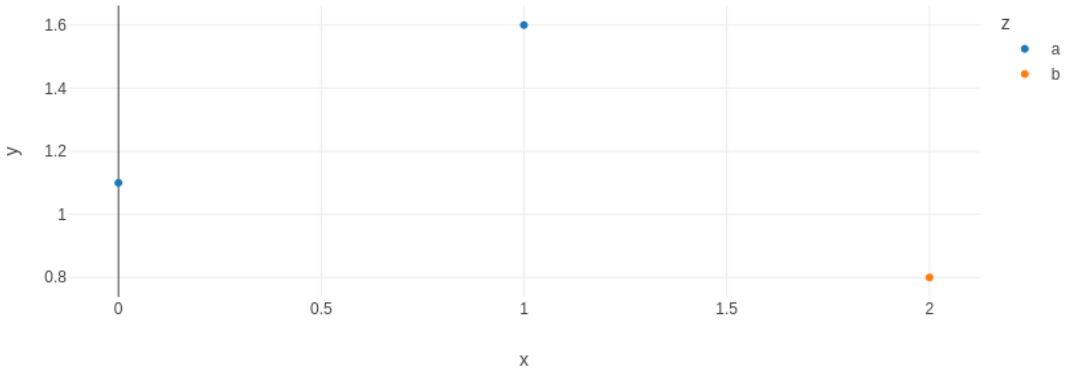
[11]:
```
# merely replacing scatter with bar will do just the same in the json␣
 ↪representation:
# to the grammar is quite straight-forward in this sense
fig = go.Figure()
fig.add_trace(go.Bar(x=data["x"],y=data["y"]))
fig.update_layout(template="none")
# fig.show()
plotly.io.to_json(fig)
```

[11]:
```
'{"data":[{"x":[0.0,1.0,2.0],"y":[1.1,1.6,0.8],"type":"bar"}],"layout":{"templat
e":{"data":{"scatter":[{"type":"scatter"}]}}}}'
```

1.2 Where API and JSON rep begin to differ

The above might suggest that API and JSON are very much aligned, but with more complicated high-level functions we find that a simple function generates quite complex JSON code with lots of ”boilerplate” code. So the internal grammar is much more low-level than the one we saw in ggplot.

[12]:
```
# now we proceed as above, but we map another variable to the "aesthetic" color␣
 ↪(using ggplots nomenclature)
# this can only be done at the high level interface
# by the way: note that above the go-functions do not directly interact with␣
 ↪the dataframe!
fig=px.scatter(data,"x","y",color="z",template="none")
fig.show()
```



[13]:
```
# looking now at the json code, we find that internally the single trace has␣
 ↪been split into two
# and that the legend information is already explicitly contained in the json
# (instead of being inferred form the data, as it should be potentially␣
 ↪possible)
print(plotly.io.to_json(fig))
```

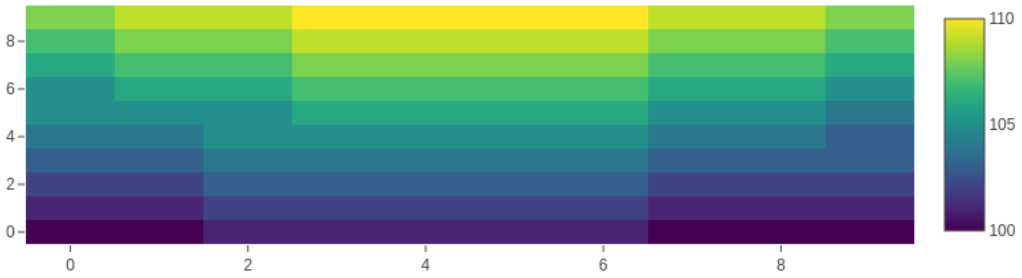
{"data":[{"hovertemplate":"z=a
x=%{x}
y=%{y}<extra></extra>","legendgroup"
:"a","marker":{"color":"#1F77B4","symbol":"circle"},"mode":"markers","name":"a",
"orientation":"v","showlegend":true,"x":[0.0,1.0],"xaxis":"x","y":[1.1,1.6],"yax
is":"y","type":"scatter"},{"hovertemplate":"z=b
x=%{x}
y=%{y}<extra></extr
a>","legendgroup":"b","marker":{"color":"#FF7F0E","symbol":"circle"},"mode":"mar
kers","name":"b","orientation":"v","showlegend":true,"x":[2.0],"xaxis":"x","y":[
0.8],"yaxis":"y","type":"scatter"}],"layout":{"template":{"data":{"scatter":[{"t
ype":"scatter"}]}},"xaxis":{"anchor":"y","domain":[0.0,1.0],"title":{"text":"x"}
},"yaxis":{"anchor":"x","domain":[0.0,1.0],"title":{"text":"y"}},"legend":{"titl
e":{"text":"z"},"tracegroupgap":0},"margin":{"t":60}}}

1.3 But API can still operate as if a high-level grammar (almost) existed

[15]:
```
# import some useful sample datasets
# taken from https://raw.githubusercontent.com/plotly/datasets/master/volcano.
 ↪csv
volcanodf=pd.read_csv("volcano.csv")
volcanoFull=np.array(volcanodf)
volcano=volcanoFull[:10,:10]
```

[16]:
```
fig = go.Figure()
fig.add_trace(go.Heatmap(z=volcano, colorscale="Viridis"))
fig.update_layout(template="none")
```

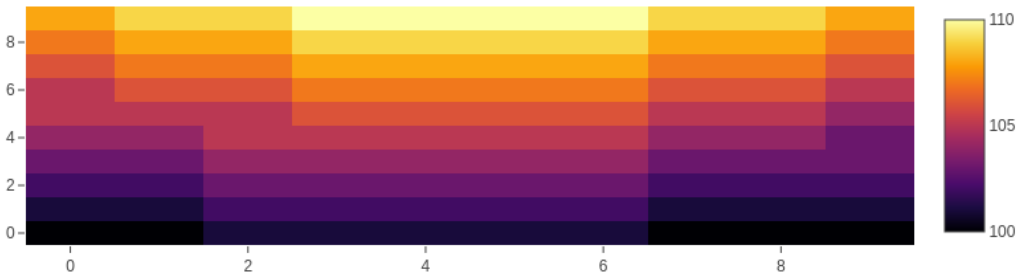
```
fig.show()
```



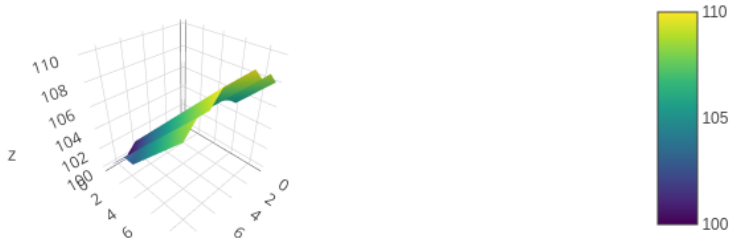
[82]:
```
print(plotly.io.to_json(fig))
```


```
{"data":[{"colorscale":[[0.0,"#440154"],[0.1111111111111111,"#482878"],[0.222222
2222222222,"#3e4989"],[0.3333333333333333,"#31688e"],[0.4444444444444444,"#26828
e"],[0.5555555555555556,"#1f9e89"],[0.6666666666666666,"#35b779"],[0.77777777777
77778,"#6ece58"],[0.8888888888888888,"#b5de2b"],[1.0,"#fde725"]],"z":[[100,100,1
01,101,101,101,101,100,100,100],[101,101,102,102,102,102,102,101,101,101],[102,1
02,103,103,103,103,103,102,102,102],[103,103,104,104,104,104,104,103,103,103],[1
04,104,105,105,105,105,105,104,104,103],[105,105,105,106,106,106,106,105,105,104
],[105,106,106,107,107,107,107,106,106,105],[106,107,107,108,108,108,108,107,107
,106],[107,108,108,109,109,109,109,108,108,107],[108,109,109,110,110,110,110,109
,109,108]],"type":"heatmap"}],"layout":{"template":{"data":{"scatter":[{"type":"
scatter"}]}}}}
```


[17]:
```
fig.update_traces(colorscale="Inferno")
```



[18]:
```
# and we only need to change one keyword and the whole plot changes,
# like using a different "geom"
fig = go.Figure()
fig.add_trace(go.Surface(z=volcano, colorscale="Viridis"))
fig.update_layout(template="none")
fig.show()
```

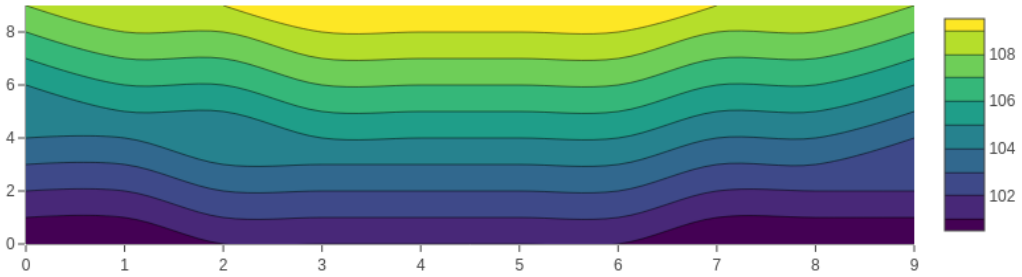


[19]:
```
# this json serialization only differs by the above in that "heatmap" is␣
 ↪replaced by "surface"
print(plotly.io.to_json(fig))
```


```
{"data":[{"colorscale":[[0.0,"#440154"],[0.1111111111111111,"#482878"],[0.222222
2222222222,"#3e4989"],[0.3333333333333333,"#31688e"],[0.4444444444444444,"#26828
e"],[0.5555555555555556,"#1f9e89"],[0.6666666666666666,"#35b779"],[0.77777777777
77778,"#6ece58"],[0.8888888888888888,"#b5de2b"],[1.0,"#fde725"]],"z":[[100,100,1
01,101,101,101,101,100,100,100],[101,101,102,102,102,102,102,101,101,101],[102,1
02,103,103,103,103,103,102,102,102],[103,103,104,104,104,104,104,103,103,103],[1
04,104,105,105,105,105,105,104,104,103],[105,105,105,106,106,106,106,105,105,104
],[105,106,106,107,107,107,107,106,106,105],[106,107,107,108,108,108,108,107,107
,106],[107,108,108,109,109,109,109,108,108,107],[108,109,109,110,110,110,110,109
,109,108]],"type":"surface"}],"layout":{"template":{"data":{"scatter":[{"type":"
scatter"}]}}}}
```


[21]:
```
# still! such a change is apparently too fundamental and would not always work
# so plotly will not allow the following, showing again some deviation from an
# actual high-level formal grammar
#fig.update_traces(type="Surface")
```

[22]:
```
# here is another nice example
fig = go.Figure()
fig.add_trace(go.Contour(z=volcano, colorscale="Viridis"))
fig.update_layout(template="none")
fig.show()
```



[96]:
```
# this json serialization only differs by the above in that "heatmap" is␣
 ↪replaced by "surface"
print(plotly.io.to_json(fig))
```


```
{"data":[{"colorscale":[[0.0,"#440154"],[0.1111111111111111,"#482878"],[0.222222
2222222222,"#3e4989"],[0.3333333333333333,"#31688e"],[0.4444444444444444,"#26828
e"],[0.5555555555555556,"#1f9e89"],[0.6666666666666666,"#35b779"],[0.77777777777
77778,"#6ece58"],[0.8888888888888888,"#b5de2b"],[1.0,"#fde725"]],"z":[[100,100,1
01,101,101,101,101,101,100,100,100],[101,101,102,102,102,102,102,101,101,101],[102,1
02,103,103,103,103,103,102,102,102],[103,103,104,104,104,104,104,103,103,103],[1
04,104,105,105,105,105,105,104,104,103],[105,105,105,106,106,106,106,105,105,104
],[105,106,106,107,107,107,107,106,106,105],[106,107,107,108,108,108,108,107,107
,106],[107,108,108,109,109,109,109,108,108,107],[108,109,109,110,110,110,110,109
,109,108]],"type":"contour"}],"layout":{"template":{"data":{"scatter":[{"type":"
scatter"}]}}}}
```


[ ]: