Dash for data visualization

mishima.syk #18

@iwatobipen

Who am I

● Twitter	https://twitter.com/iwatobipen
● 駄文	https://iwatobipen.wordpress.com/
● Github	https://github.com/iwatobipen

● 某製薬企業ケモインフォチームの中の人

Today’s topic

- Dashで簡単な可視化アプリを作ってみる

from Dash Introduction

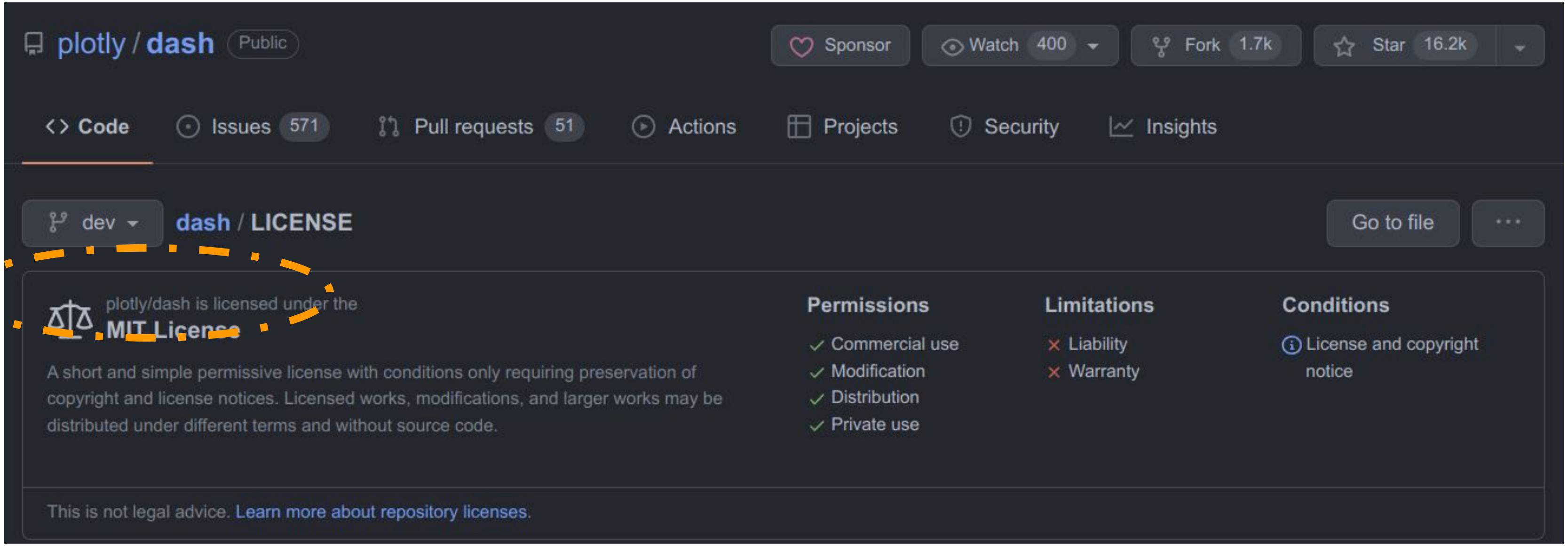
- Dash is the original low-code framework for rapidly building data apps in Python, R, Julia, and F# (experimental).
- Written on top of **Plotly.js** and **React.js,** Dash is ideal for **building and deploying data apps with customized user interfaces**. It's particularly suited for anyone who works with data.
- Dash is simple enough that you can bind a user interface to your code in less than 10 minutes.

$\Downarrow$

- Dashはイケてるデータ可視化パッケージ（意訳）

https://dash.plotly.com/introduction

License MIT (Enterpriseもある)



dash / LICENSE

plotly/dash is licensed under the
MIT License

A short and simple permissive license with conditions only requiring preservation of copyright and license notices. Licensed works, modifications, and larger works may be distributed under different terms and without source code.

Permissions	Limitations	Conditions
✓ Commercial use	✕ Liability	ⓘ License and copyright notice
✓ Modification	✕ Warranty	
✓ Distribution		
✓ Private use		

This is not legal advice. Learn more about repository licenses.

https://github.com/plotly/dash/blob/dev/LICENSE

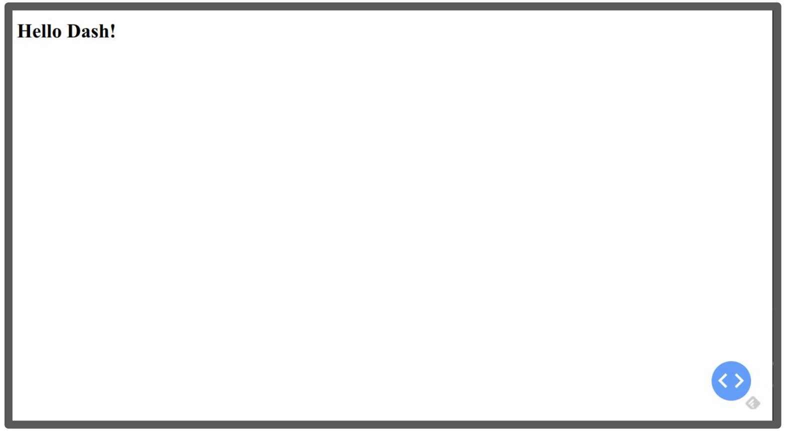
Hello Dash!

```
#hello_world.py
from dash import Dash, html, dcc
app = Dash(__name__)

app.layout = html.Div(
    children=[html.H1("Hello Dash!")]
)
if __name__=="__main__":
    app.run_server(debug=True)
```

Flaskと同じお作法

パーツをどんどん組み込む

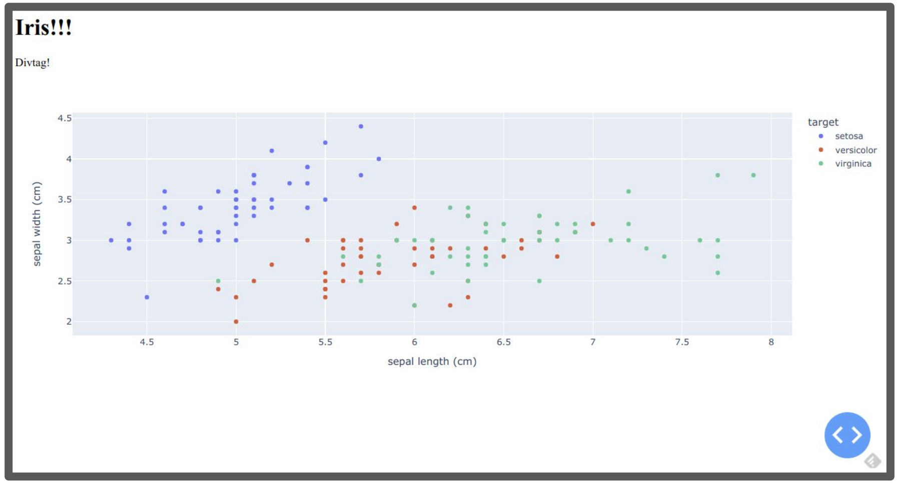
http://localhost:8050/


Hello Dash!

Plot iris !!

```
#hello_plot.py

## snip

fig = px.scatter(df, x=data.feature_names[0],

    y=data.feature_names[1], color='target')

app = Dash(__name__)

app.layout = html.Div(children=[

    html.H1(children='Iris!!!'),

    html.Div(children="Divtag!"),

    dcc.Graph(  id="iris!", figure=fig)])

## snip
```

Plotlyできれいなプロット♪

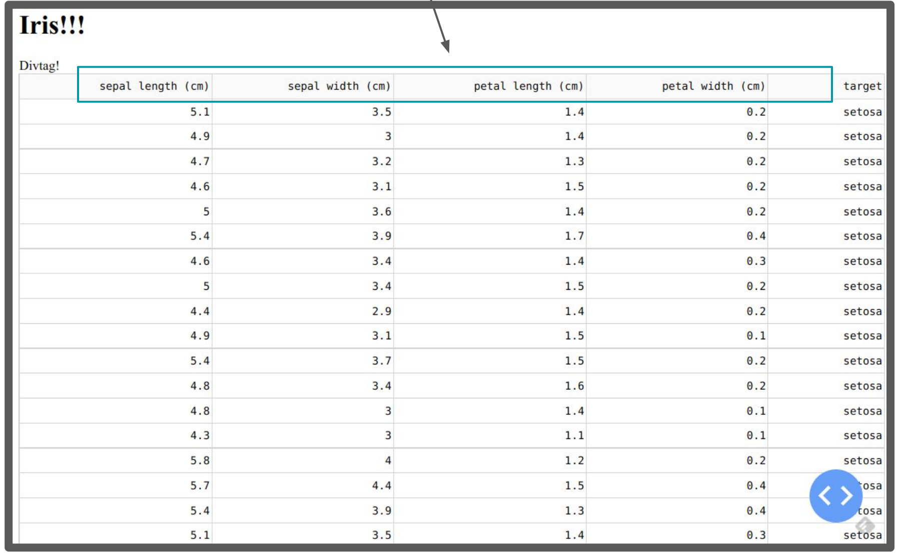

Iris!!!

Divtag!

Iris dataset has 4 features…

```
#hello_table.py

## snip

from dash import dash_table

app = Dash(__name__)

app.layout = html.Div(children=[

    html.H1(children='Iris!!!'),

    html.Div(children="Divtag!"),

    dash_table.DataTable(df.to_dict('records'),
[{"name": i, "id": i} for i in df.columns])

    ])
```

この4つを任意に選んでプロットしたいよね



Iris!!!

Divtag!

sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
5.1	3.5	1.4	0.2	setosa
4.9	3	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa
5.4	3.7	1.5	0.2	setosa
4.8	3.4	1.6	0.2	setosa
4.8	3	1.4	0.1	setosa
4.3	3	1.1	0.1	setosa
5.8	4	1.2	0.2	setosa
5.7	4.4	1.5	0.4	tosa
5.4	3.9	1.3	0.4	tosa
5.1	3.5	1.4	0.3	setosa

Let's use callback

```
from dash import Dash, dcc, html, Input, Output
app = Dash(__name__)
app.layout = html.Div([
    html.H6("Callbacks in action!"),
    html.Div([
        "Input: ",
        dcc.Input(id='my-input', value='initial value', type='text')
    ]),
    html.Br(),
    html.Div(id='my-output'),
])

@app.callback(
   Output(component_id='my-output',
   component_property='children'),
   Input(component_id='my-input', component_property='value'))
def update_output_div(input_value):
    return f'Output: {input_value}'
if __name__ == '__main__':
    app.run_server(debug=True)
```

dcc以下に様々な入力用のクラスがある

入出力をCallBack側で制御。

https://dash.plotly.com/basic-callbacks

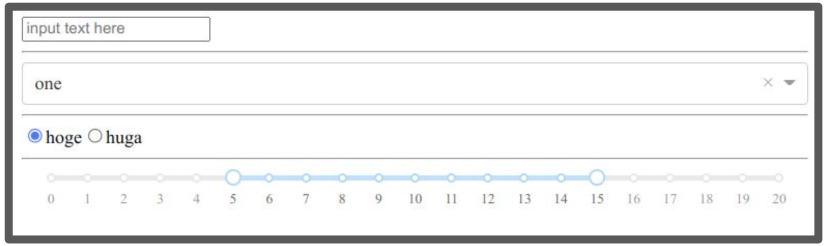
There are lots of parts in dcc ;)

dcc.Dropdown	ドロップダウンリストの実装に	散布図の軸選択
dcc.Input	テキスト入力に	
dcc.Slider	スライダーで数値指定	
dcc.Upload	ファイルアップロード	SDFをアップロードとか
dcc.RadioItems	ラジオボタン	項目選択に
dcc.RangeSlider	x〜yのようなレンジでの選択	分子物性の選択などに
などなど。。。		

https://dash.plotly.com/dash-core-components

dcc examples

```
# dcc_sample.py

from dash import Dash, dcc, html

app = Dash(__name__)

app.layout = html.Div(children=[

    dcc.Input(id='input', placeholder='input text here'),

    dcc.Dropdown(['one', 'two', 'three'], 'one', id='dropdown'),

    dcc.RadioItems(['hoge', 'huga'], 'hoge'),

    dcc.RangeSlider(0, 20, 1, value=[5,15],id='rangeslider')

])

if __name__=="__main__":

    app.run_server(debug=True)
```

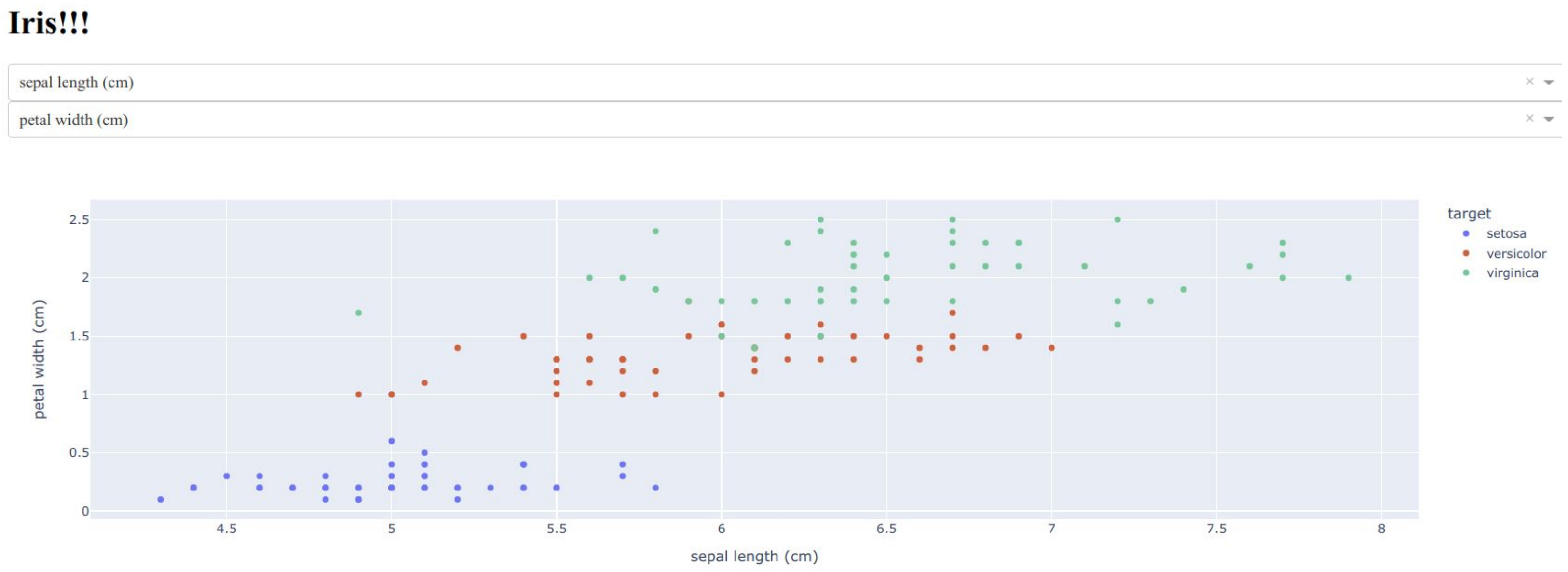


Interactive plot of Iris ;-)

app.layoutをCallbackの方で制御する
＞id属性でデータを紐付け

```
# hello_plot_v2.py
app = Dash(__name__)
app.layout = html.Div(children=[html.H1(children='Iris!!!'),
    dcc.Dropdown(data.feature_names, id='x-axis'),
    dcc.Dropdown(data.feature_names, id='y-axis'),
    dcc.Graph(id="iris")])
@app.callback(Output('iris', 'figure'), Input('x-axis', 'value'), Input('y-axis', 'value'))
def updatefig(xval, yval):
    fig = px.scatter(df,  x=xval, y=yval, color='target')
    return fig
```

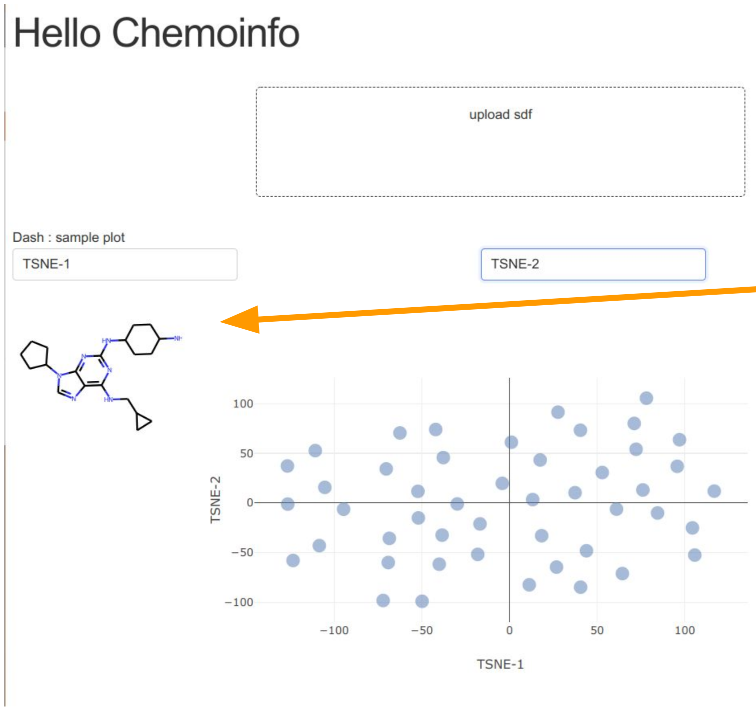
Interactive plot of Iris ;-)

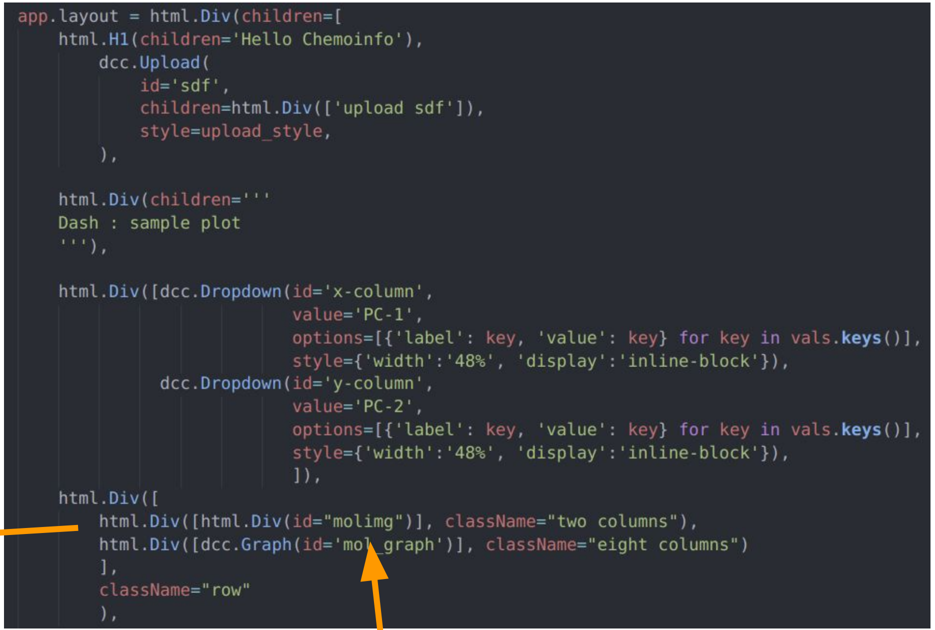

Iris!!!

sepal length (cm)

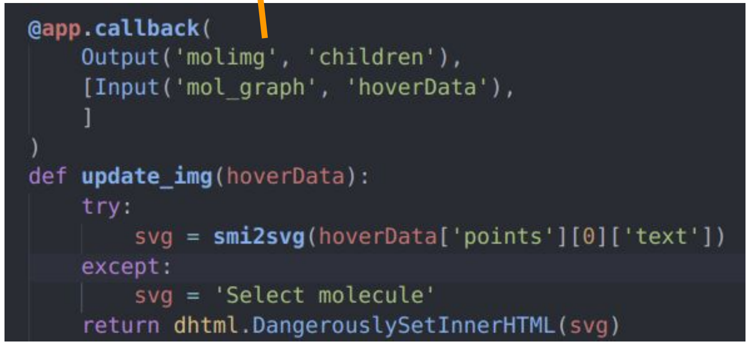
petal width (cm)

Chemoinfo appも作れるよ



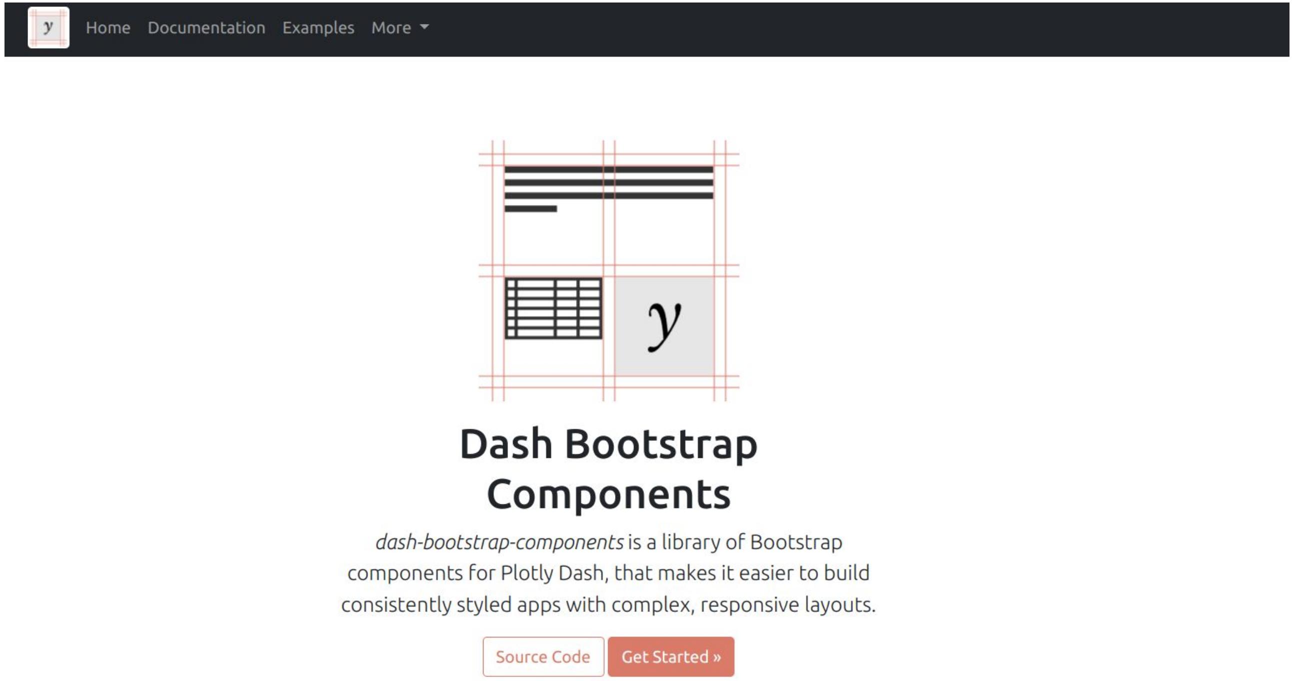


```
app.layout = html.Div(children=[
    html.H1(children='Hello Chemoinfo'),
        dcc.Upload(
            id='sdf',
            children=html.Div(['upload sdf']),
            style=upload_style,
        ),

    html.Div(children='''
    Dash : sample plot
    '''),

    html.Div([dcc.Dropdown(id='x-column',
                           value='PC-1',
                           options=[{'label': key, 'value': key} for key in vals.keys()],
                           style={'width':'48%', 'display':'inline-block'}),
              dcc.Dropdown(id='y-column',
                           value='PC-2',
                           options=[{'label': key, 'value': key} for key in vals.keys()],
                           style={'width':'48%', 'display':'inline-block'}),
                           ]),
    html.Div([
        html.Div([html.Div(id="molimg")], className="two columns"),
        html.Div([dcc.Graph(id='mol_graph')], className="eight columns")
        ],
        className="row"
        ),
```



```
@app.callback(
    Output('molimg', 'children'),
    [Input('mol_graph', 'hoverData'),
    ]
)
def update_img(hoverData):
    try:
        svg = smi2svg(hoverData['points'][0]['text'])
    except:
        svg = 'Select molecule'
    return dhtml.DangerouslySetInnerHTML(svg)
```

もう少しレイアウトをという方にはこちら！



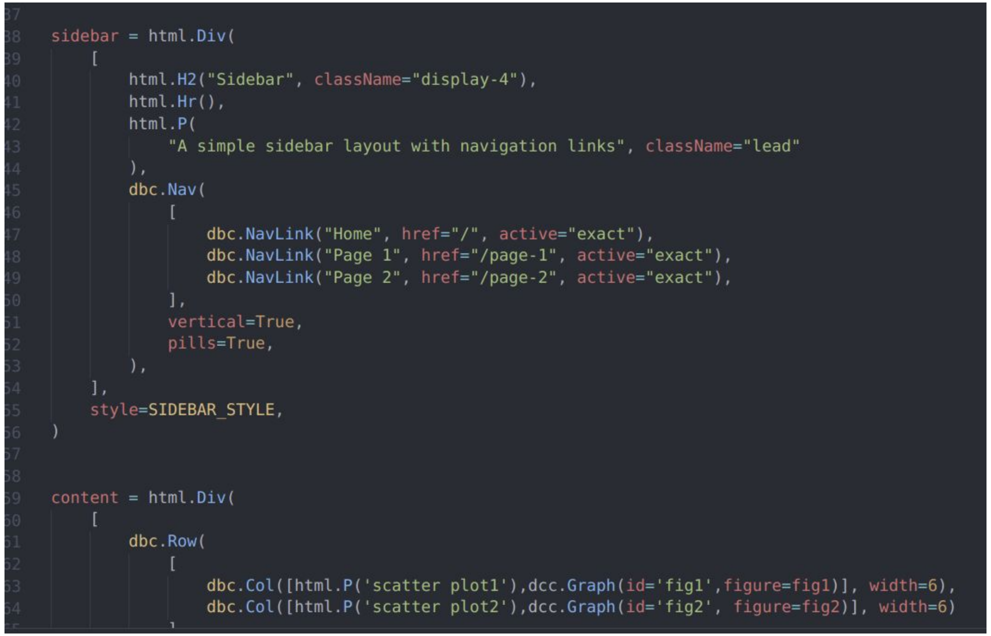
Dash Bootstrap Components

dash-bootstrap-components is a library of Bootstrap components for Plotly Dash, that makes it easier to build consistently styled apps with complex, responsive layouts.

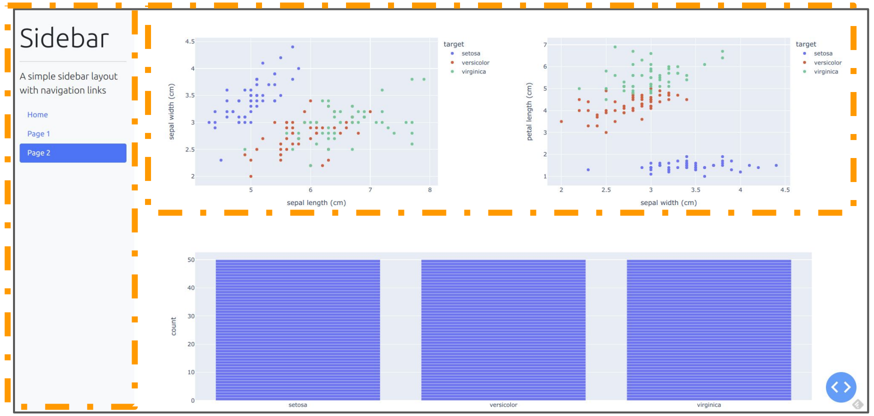
https://dash-bootstrap-components.opensource.faculty.ai/

#sidebar_with_graph.py

DBCを使ってレイアウトを簡単にコントロール



```
sidebar = html.Div(
    [
        html.H2("Sidebar", className="display-4"),
        html.Hr(),
        html.P(
            "A simple sidebar layout with navigation links", className="lead"
        ),
        dbc.Nav(
            [
                dbc.NavLink("Home", href="/", active="exact"),
                dbc.NavLink("Page 1", href="/page-1", active="exact"),
                dbc.NavLink("Page 2", href="/page-2", active="exact"),
            ],
            vertical=True,
            pills=True,
        ),
    ],
    style=SIDEBAR_STYLE,
)


content = html.Div(
    [
        dbc.Row(
            [
                dbc.Col([html.P('scatter plot1'),dcc.Graph(id='fig1',figure=fig1)], width=6),
                dbc.Col([html.P('scatter plot2'),dcc.Graph(id='fig2', figure=fig2)], width=6)
```



12のColumnで構成されるGridレイアウト

ところで、、、、

- 既存のFlask Appがもうあるんだけど、Dashをそこに組み込めたりしないの？
- サービスが全部独立してしまうのはちょっと、、、

という貴方に！！！

```
from dash import Dash
from werkzeug.middleware.dispatcher import DispatcherMiddleware
from werkzeug.serving import run_simple
import flask
from flask import Flask
from dash import html
server = Flask(__name__)

dash_app1 = Dash(__name__, server = server, url_base_pathname='/dashboard/' )
dash_app2 = Dash(__name__, server = server, url_base_pathname='/reports/')
dash_app1.layout = html.Div([html.H1('Hi there, I am app1 for dashboards')])
dash_app2.layout = html.Div([html.H1('Hi there, I am app2 for reports')])

@server.route('/dashboard/')
def render_dashboard():
    return flask.redirect('/dash1')

@server.route('/reports/')
def render_reports():
    return flask.redirect('/dash2')
app = DispatcherMiddleware(server, {
    '/dash1': dash_app1.server,
    '/dash2': dash_app2.server,})

run_simple('localhost', 8080, app, use_reloader=True, use_debugger=True)
```

その他参考情報
☆PatWalterさんのBlogは要チェック！

- https://practicalcheminformatics.blogspot.com/2019/11/interactive-plots-with-chemical.html
- https://iwatobipen.wordpress.com/2022/02/20/integration-of-molplotly-and-flask-for-developing-chemoinformatics-web-app-rdkit-molplotly-flask/
- https://github.com/wjm41/molplotly
-

まとめ

- Dashを使って良さげな可視化を提供しよう！
- リッチな可視化を提供する場合Call backやLayoutなどそこそこコード書く必要はあるが、フルスクラッチで開発するよりはコストが低い
- Plotlyのプロットきれいで動的なの良いですね
- 企業であれば便利なBIツールを使うかな
- 今日のコードと資料はMishima-sykリポジトリをチェケ！

https://github.com/Mishima-syk/18/tree/main/iwatobipen

おしまい