2%20-%20Dash%20-%20Getting%20started

April 12, 2023

Topics

1 What is Dash?

- Dash is a 'low-code' framework (set of tools and procedures) for building dashboard applications using Python, Julia or R.
- It is tightly integrated with plotly, the data visualization library.
- The dashboard can be rendered as part of the Jupyter notebook, in an HTML file on a local machine or as an HTML file on a web server.

2 Building blocks

3 Layouts and Callbacks

- Dash apps are composed of two parts.
- The first part is the "layout" of the app and it describes what the application looks like.

- Callbacks, the second part, describes the interactivity of the application

4 HTML components and dash core components

- HTML components are the dash equivalant of html tags
- Core components (dcc) are graphs, markdown blocks and interactivity components like sliders, dropdowns, etc.

[ ]:
```
# The general structure of a very simple dashboard application.


imports .....

get the data....

create a figure(plot)...

app = JupyterDash(__name__)     # This is the start of the application
```

```
app.layout = html.Div(          # Describe what the page will look like
    ↪

    layout code

    dcc.Graph()                  # What plot will be included

)

app.run_server(mode='inline')    # .run_server() is the method to run the code
```

5 A very simple dashboard app

The pip install below is only required because we are running in the Google CoLab environment. Remove the hashtag (#) to uncomment the line.

[ ]:
```
# The general structure of a dashboard application:

imports .....

app = JupyterDash(__name__)      # This is the start of the application

get the data....

create a figure(plot)...

app.layout =                     # Describe what the page will look like

    layout code

    dcc.Graph()                  # What plot will be included

@app.callback(
    what are the inputs?
    what are the outputs?

    resusable component )        # This processes the input and creates the
    ↪output

app.run_server(mode='inline')    # .run_server() is the method to run the code
```

6 Layout example

[ ]:
```
#!pip install jupyter-dash
```

[ ]:
```
from jupyter_dash import JupyterDash
from dash.dependencies import Output, Input
from dash import no_update
from dash import dcc
from dash import html

import pandas as pd
import plotly.graph_objects as go
import plotly.express as px
import math

ob = pd.read_csv('https://raw.githubusercontent.com/jimcody2014/Python-Data/
 ↪main/outbreaks-dashboard.csv')
ob_month = ob.groupby('Month')[['Illnesses','Hospitalizations', 'Fatalities']].
 ↪sum().reset_index()
oby = ob.groupby('Year')[['Illnesses','Hospitalizations', 'Fatalities']].sum().
 ↪reset_index()
obs = ob.groupby('State')[['Illnesses','Hospitalizations', 'Fatalities']].sum().
 ↪reset_index()


# Create an empty figure here
fig = go.Figure()

app = JupyterDash(__name__)

# Layout the dashboard

# app.layout - html.Div( something goes in the .Div )
# app.layout - html.Div( [sometimes a list of things go into the .Div] )
# app.layout - html.Div( [sometimes other .Divs go in the .Div] )

app.layout = html.Div([       # passing in a list of 'things' to Div
    html.H1('Hello Jim'),     # This line generates <h1>Hello Jim</h1>

    html.Div('''
        An Empty Dashboard
    '''),

    dcc.Graph(
        id='example-graph',
        figure=fig
    )
```

```
])
#app.run_server(mode='inline')
app.run_server(mode='external', port = 8060)
```

7 Callback example

[ ]:
```
# An example of a callback from documentation
# Just changes the text that appears - no plotting

from jupyter_dash import JupyterDash
from dash.dependencies import Output, Input
from dash import dcc
from dash import html

app = JupyterDash(__name__)

app.layout = html.Div([
    html.H6("Change the value in the text box to see 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 'Output: {}'.format(input_value)

#app.run_server(mode='inline')
app.run_server(mode='external', port = 8071)
```

[ ]:
```

```