

Practice Assignment Part 2

Objectives

After completing the lab you will be able to:

- Create a dash board layout with a RadioItem and a Dropdown
- Add Pie chart and Bar chart

Estimated time needed: 45 minutes

About Skills Network Cloud IDE

This Skills Network Labs Cloud IDE (Integrated Development Environment) provides a hands-on environment in your web browser for completing course and project related labs. It utilizes Theia, an open-source IDE platform, that can be run on desktop or on the cloud. So far in the course you have been using Jupyter notebooks to run your python code. This IDE provides an alternative for editing and running your Python code. In this lab you will be using this alternative Python runtime to create and launch your Dash applications.

Important Notice about this lab environment

Please be aware that sessions for this lab environment are not persisted. When you launch the Cloud IDE, you are presented with a ‘dedicated computer on the cloud’ exclusively for you. This is available to you as long as you are actively working on the labs.

Once you close your session or it is timed out due to inactivity, you are logged off, and this cloud computer on the cloud is destroyed along with any files you may have created, downloaded or installed. The next time you launch this lab, a new environment is created for you.

If you finish only part of the lab and return later, you may have to start from the beginning. So, it is a good idea to plan to your time accordingly and finish your labs in a single session.

Components of Dashboard and Expected layout

Components of the Dashboard

1. Select Region
 2. Select Year
 3. Division to display
 - Pie Chart to display Monthly Average Estimated Fire Area for the selected Regions in the selected Year
 - Bar Chart to display Monthly Average Count of Pixels for Presumed Vegetation Fires for the selected Regions in the selected Year
-

Expected Layout

Requirements to create the expected result

- A dropdown [menu](#): For choosing year
- A radioitem for choosing the Region
- The layout will be designed as follows:
- An outer division with two inner divisions (as shown in the expected layout)
- One of the inner divisions will have information about the radioitem and dropdown (which are the input) and the other one is for adding graphs(the 2 output graphs).
- Callback function to compute data, create graph and return to the layout.

To do:

1. Import required libraries and read the dataset
 2. Create an application layout
 3. Add title to the dashboard using HTML H1 component
 4. Add a radioitem using dcc.RadioItems and dropdown using dcc.dropdown
 5. Add the pie chart and bar chart core graph components.
 6. Run the app
-

Get the tool ready

1. Open a new terminal, by clicking on the menu bar and selecting **Terminal->New Terminal**, as in the image below.
2. Install python packages required to run the application. Copy and paste the below command to the terminal.

```
pip3.8 install setuptools
```

```
python3.8 -m pip install packaging
```

```
python3.8 -m pip install pandas dash
```

```
pip3 install httpx==0.20 dash plotly
```

Let's create the application

- Create a new file called `Dash_wildfire.py`

Get the application skeleton

Note: Throughout this lab, you will see placeholder text like '.....' or `groupby(.....)`. These dots should be replaced with appropriate values such as column names ('Year', 'Vehicle_Type', etc.), grouping keys, or chart titles. Refer to the comments and hints above each task to determine what needs to be filled in.

You can use this as a base code to complete the task below.

Structure of the skeleton file

```
import pandas as pd
import dash
from dash import html, dcc
from dash.dependencies import Input, Output, State
import plotly.graph_objects as go
import plotly.express as px
from dash import no_update
import datetime as dt
#Create app
app = dash.Dash(__name__)
# Clear the layout and do not display exception till callback gets executed
app.config.suppress_callback_exceptions = True
# Read the wildfire data into pandas dataframe
df = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/Data%20Fi
#Extract year and month from the date column
df['Month'] = pd.to_datetime(df['Date']).dt.month_name() #used for the names of the months
df['Year'] = pd.to_datetime(df['Date']).dt.year
#Layout Section of Dash
#Task 2.1 Add the Title to the Dashboard
app.layout = html.Div(children=[html.H1(.....),
# TASK 2.2: Add the radio items and a dropdown right below the first inner division
#outer division starts
    html.Div([
        # First inner division for adding dropdown helper text for Selected Drive wheels
        html.Div([
            html.H2(.....),
            #Radio items to select the region
            #dcc.RadioItems(['NSW',.....], value = '...', id='...',inline=True)),
            dcc.RadioItems([{"label": "New South Wales", "value": "NSW"},
                {.....},
                {.....},
                {.....},
                {.....},
                {.....},
                {"label": "...", "value": ..}], value = "...", id='.....,inline=True))],
```

```

        #Dropdown to select year
        html.Div([
            html.H2('.....', style={.....}),
            dcc.Dropdown(.....)
        ]),
#Second Inner division for adding 2 inner divisions for 2 output graphs
#TASK 2.3: Add two empty divisions for output inside the next inner division.
        html.Div([
            html.Div([ ], id='.....'),
            html.Div([ ], id='.....')
        ], style={'.....'}),
    ])
#outer division ends
])
#layout ends
#TASK 2.4: Add the Ouput and input components inside the app.callback decorator.
#Place to add @app.callback Decorator
@app.callback([Output(component_id=....., component_property=.....),
               Output(component_id=....., component_property=.....)],
              [Input(component_id=....., component_property=.....),
               Input(component_id=....., component_property=.....)])

#TASK 2.5: Add the callback function.
#Place to define the callback function .
def reg_year_display(input_region,input_year):

    #data
    region_data = df[df['Region'] == input_region]
    y_r_data = region_data[region_data['Year']==input_year]
    #Plot one - Monthly Average Estimated Fire Area

    est_data = .....
    fig1 = px.pie(....., title="{} : Monthly Average Estimated Fire Area in year {}".format(input_region,input_year))

    #Plot two - Monthly Average Count of Pixels for Presumed Vegetation Fires
    veg_data = .....
    fig2 = px.bar(....., title='{} : Average Count of Pixels for Presumed Vegetation Fires in year {}'.format(input_region,input_year))

    return [.....,
            ..... ]
if __name__ == '__main__':
    app.run()

```

TASK 2.1: Add title to the dashboard

Update the `html.H1()` tag to hold the application title.

- Application title is Australia Wildfire Dashboard
- Use style parameter provided below to make the title center aligned, with color code #503D36, and font-size as 26

```

html.H1('Australia Wildfire Dashboard',
        style={'textAlign': 'center', 'color': '#503D36',
               'font-size': 26}),

```

After updating the `html.H1()` with the application title, the `app.layout` will look like:

Reference Links:

[H1 component](#)

[Dash HTML Components](#)

TASK 2.2: Add the radio items and a dropdown right below the first inner division.

Radio items to choose the Region

The radio items work similar to the dropdown, you need to call `dcc.RadioItems` and pass the list of items. Make use of `inline=True` property to display the radio items in a horizontal line

- You can extract the regions from the dataframe using `df.Region.unique()` or pass the list of all regions directly as `['NSW', 'QL', 'SA', 'TA', 'VI', 'WA', 'NT']` .
- Assign radioitems id as `region`

- Label as Select Region
- value as NSW

For your reference below are the abrivations used in the dataset for regions

NSW - New South Wales

NT - Northern Territory

QL - Queensland

SA - South Australia

TA - Tasmania

VI - Victoria

WA - Western Australia

Read more on [RadioItems](#)

```
html.Div([
    html.H2('Select Region:', style={'margin-right': '2em'}),
    #Radio items to select the region
    dcc.RadioItems(['NSW', 'QL', 'SA', 'TA', 'VI', 'WA'], 'NSW', id='region', inline=True)],
```

- or you can use labels:value pair a well in raioditems as below

```
#OR you can use labels:value pair a well in raioditems as below
#Radio items to select the region
dcc.RadioItems([{"label": "New South Wales", "value": "NSW"},
                 {"label": "Northern Territory", "value": "NT"},
                 {"label": "Queensland", "value": "QL"},
                 {"label": "South Australia", "value": "SA"},
                 {"label": "Tasmania", "value": "TA"},
                 {"label": "Victoria", "value": "VI"},
                 {"label": "Western Australia", "value": "WA"}], "NSW", id='region', inline=True)],
```

Dropdown to choose the Year

- The dropdown has an id as year.
- The label as Select Year
- The values allowed in the dropdown are years from 2005 to 2020
- The default value when the dropdown is displayed is 2005.

```
html.Div([
    html.H2('Select Year:', style={'margin-right': '2em'}),
    dcc.Dropdown(df.Year.unique(), value = 2005, id='year')
    #notice the use of unique() from pandas to fetch the values of year from the dataframe for dropdown
]),
```

Reference [link](#)

TASK 2.3: Add two empty divisions for output inside the next inner division.

- Use 2 `html.Div()` tags .
- Provide division ids as `plot1` and `plot2`.

```
html.Div([ ], id='plot1'),
html.Div([ ], id='plot2')
```

TASK 2.4: Add the Output and input components inside the app.callback decorator.

- The inputs and outputs of our application's interface are described declaratively as the arguments of `@app.callback` decorator.

-In Dash, the inputs and outputs of our application are simply the properties of a particular component.

- In this example, we have two inputs:-
 - input for Region is the `value` property of the component that has the ID `region`
 - input for Year is the `value` property of the component that has the ID `year`
- Our layout has 2 outputs so we need to create 2 output components.

It is a list with 2 output parameters with component id and property. Here, the component property will be `children` as we have created empty division and passing in `dcc.Graph` (figure) after computation.

Component ids will be `plot1` , `plot2`.

```
@app.callback([Output(component_id='plot1', component_property='children'),
                Output(component_id='plot2', component_property='children')],
              [Input(component_id='region', component_property='value'),
               Input(component_id='year', component_property='value')])
```

TASK 2.5: Add the callback function.

- Whenever an input property changes, the function that the callback decorator wraps will get called automatically.
- In this case let us define a function `reg_year_display()` which will be wrapped by our decorator.
- The function first filters our dataframe `df` by the selected value of the region from the radio items and year from the dropdown as follows
- `region_data = df[df['Region'] == input_region]`
- `y_r_data = region_data[region_data['Year']==input_year]`
- For pie chart on Monthly Average Estimated Fire Area: -
 - Next we will group by the Month and calculate the mean `Estimated_fire_area` of the dataframe.
 - Use the `px.pie()` function to plot the pie chart
- For bar chart on Monthly Average Count of Pixels for Presumed Vegetation Fires: -
 - Next we will group by the Month and calculate the mean `Count` of the dataframe.
 - Use the `px.bar()` function to plot the bar chart

```
def reg_year_display(input_region,input_year):

    #data
    region_data = df[df['Region'] == input_region]
    y_r_data = region_data[region_data['Year']==input_year]
    #Plot one - Monthly Average Estimated Fire Area

    est_data = y_r_data.groupby('Month')['Estimated_fire_area'].mean().reset_index()

    fig1 = px.pie(est_data, values='Estimated_fire_area', names='Month', title="{} : Monthly Average Estimated Fire Area in year {}".format(input_region, input_year))

    #Plot two - Monthly Average Count of Pixels for Presumed Vegetation Fires
    veg_data = y_r_data.groupby('Month')['Count'].mean().reset_index()
    fig2 = px.bar(veg_data, x='Month', y='Count', title="{} : Average Count of Pixels for Presumed Vegetation Fires in year {}".format(input_region, input_year))

    return [dcc.Graph.figure=fig1),
            dcc.Graph.figure=fig2) ]
```

- Finally we return the 2 figure objects `fig1` and `fig2` in `dcc.Graph` method.
- Once you have finished coding save your code.

Run the Application

- Next Run the python file using the command

```
python3.8 Dash_wildfire.py
```

- Observe the port number shown in the terminal.
- Click on the Launch Application option from the menu bar. Provide the port number and click OK

Refer to the complete code Dash_wildfire.py here

```
import pandas as pd
import dash
from dash import html, dcc
from dash.dependencies import Input, Output, State
import plotly.graph_objects as go
import plotly.express as px
from dash import no_update
import datetime as dt
#Create app
app = dash.Dash(__name__)
#Clear the layout and do not display exception till callback gets executed
app.config.suppress_callback_exceptions = True
# Read the wildfire data into pandas dataframe
df = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/Data%20Fi
#Extract year and month from the date column
df['Month'] = pd.to_datetime(df['Date']).dt.month_name() #used for the names of the months
df['Year'] = pd.to_datetime(df['Date']).dt.year
#Layout Section of Dash
#Task 1 Add the Title to the Dashboard
app.layout = html.Div(children=[html.H1('Australia Wildfire Dashboard',
style={'textAlign': 'center', 'color': '#503D36',
'font-size': 26}),
# TASK 2: Add the radio items and a dropdown right below the first inner division
#outer division starts
html.Div([
# First inner division for adding dropdown helper text for Selected Drive wheels
html.Div([
html.H2('Select Region:', style={'margin-right': '2em'}),
#Radio items to select the region
#dcc.RadioItems(['NSW','QL','SA','TA','VI','WA'], 'NSW', id='region',inline=True)),
dcc.RadioItems([{"label": "New South Wales", "value": "NSW"},
{"label": "Northern Territory", "value": "NT"},
{"label": "Queensland", "value": "QL"},
{"label": "South Australia", "value": "SA"},
{"label": "Tasmania", "value": "TA"},
{"label": "Victoria", "value": "VI"},
{"label": "Western Australia", "value": "WA"}], "NSW", id='region', inline=True)),
#Dropdown to select year
html.Div([
html.H2('Select Year:', style={'margin-right': '2em'}),
dcc.Dropdown(df.Year.unique(), value = 2005, id='year')
]),
#TASK 3: Add two empty divisions for output inside the next inner division.
#Second Inner division for adding 2 inner divisions for 2 output graphs
html.Div([
html.Div([ ], id='plot1'),
html.Div([ ], id='plot2')
], style={'display': 'flex'}),
]),
#outer division ends
])
#layout ends
#TASK 4: Add the Output and input components inside the app.callback decorator.
#Place to add @app.callback Decorator
@app.callback([Output(component_id='plot1', component_property='children'),
Output(component_id='plot2', component_property='children')],
[Input(component_id='region', component_property='value'),
Input(component_id='year', component_property='value')])
#TASK 5: Add the callback function.
#Place to define the callback function .
def reg_year_display(input_region, input_year):
#data
region_data = df[df['Region'] == input_region]
y_r_data = region_data[region_data['Year']==input_year]
#Plot one - Monthly Average Estimated Fire Area
est_data = y_r_data.groupby('Month')['Estimated_fire_area'].mean().reset_index()
fig1 = px.pie(est_data, values='Estimated_fire_area', names='Month', title='{ } : Monthly Average Estimated Fire Area in year {}'.format(input
#Plot two - Monthly Average Count of Pixels for Presumed Vegetation Fires
veg_data = y_r_data.groupby('Month')['Count'].mean().reset_index()
fig2 = px.bar(veg_data, x='Month', y='Count', title='{ } : Average Count of Pixels for Presumed Vegetation Fires in year {}'.format(input_regi
return [dcc.Graph(figure=fig1),
dcc.Graph(figure=fig2) ]
if __name__ == '__main__':
app.run()
```

Congratulations, you have successfully created dash application!

Author

[Dr. Pooja](#)

© IBM Corporation. All rights reserved.