DATA 608

Module #4 - New York City Tree Census

Author: John Mazon

Data: 10/24/21

In [1]:
```
import dash
import pandas as pd
import numpy as np
import json
import plotly.offline as py
import plotly.graph_objs as go
from plotly import tools
import dash_core_components as dcc
import dash_html_components as html
import colorlover as cl
```

In this Module we'll be looking at data from the New York City tree census

This data is collected by volunteers across the city, and is meant to catalog information about every single tree in the city.

Build a dash app for a arborist studying the health of various tree species(as defined by the variable 'spc_common') across each borough (defined by the variable 'borough'). This arborist would like to answer the following two questions for each species and in each borough:

Question 1: What proportion of tress are in good, fair, or poor health according to the 'health' variable?

In [2]:
```
import warnings
warnings.filterwarnings('ignore')
#warning have been take into consideration, for this section it will be used to reduce
```

In [3]:
```
#here in the first section we import our data from cityofnewyork site for trees data
url1 = 'https://data.cityofnewyork.us/resource/nwxe-4ae8.json'
trees = pd.read_json(url1)
trees_question1 = trees[['spc_common','health','boroname']]
trees_question1['spc_common'].fillna('Unknown',inplace = True)
trees_question1.dropna(inplace = True)

#here we desire to inquire for the health conditions
ourstatus = list(set(trees_question1['health']))
ourstatus2 = list(set(trees_question1['boroname']))
ourstatus3 = list(set(trees_question1['spc_common']))


print(ourstatus)
print(ourstatus2)
print(ourstatus3)
```

```
print(trees_question1.head(10))
#above allows use to view a glimpse of our entire data
#above we print our status

colors = ['rgb(49,130,189)','rgb(204,204,204)','rgba(222,45,38,0.8)']
```

```
['Fair', 'Good', 'Poor']
['Brooklyn', 'Queens', 'Manhattan', 'Staten Island', 'Bronx']
['red maple', 'Chinese fringetree', 'Sophora', 'Callery pear', 'honeylocust', 'London pl
anetree', 'sycamore maple', 'hedge maple', 'Amur maple', 'eastern redcedar', 'Atlantic w
hite cedar', 'mulberry', 'pin oak', 'Norway maple', 'black oak', 'willow oak', 'swamp wh
ite oak', 'Douglas-fir', 'silver maple', 'white oak', 'ash', 'tulip-poplar', 'Ohio bucke
ye', 'silver linden', 'Turkish hazelnut', 'northern red oak', 'American linden', 'pignut
hickory', 'southern magnolia', 'scarlet oak', 'sawtooth oak', 'black cherry', 'Kentucky
yellowwood', 'Japanese zelkova', 'crepe myrtle', 'American elm', 'sweetgum', 'ginkgo',
'crab apple']
         spc_common health       boroname
0         red maple   Fair         Queens
1           pin oak   Fair         Queens
2       honeylocust   Good       Brooklyn
3       honeylocust   Good       Brooklyn
4   American linden   Good       Brooklyn
5       honeylocust   Good      Manhattan
6       honeylocust   Good      Manhattan
7   American linden   Good      Manhattan
8       honeylocust   Good  Staten Island
9  London planetree   Fair       Brooklyn
```

In [4]:

```
#here we are able to create columns that will specify our tree health conditions
#in addition we experiment with our other statuses

for status in set(trees_question1['health']):
    trees_question1[status] = np.where(trees_question1['health']==status,1,0)

trees_question1 = pd.DataFrame(trees_question1.groupby(['boroname','spc_common']).sum()
trees_question1.head(20)
```

Out[4]:

		Fair	Good	Poor
boroname	**spc_common**			
Bronx	**American elm**	0	2	2
	Atlantic white cedar	0	1	0
	Callery pear	0	7	0
	Douglas-fir	0	1	0
	Japanese zelkova	0	2	0
	Kentucky yellowwood	0	1	0
	London planetree	3	0	1
	Norway maple	5	15	3
	Sophora	3	5	0
	crepe myrtle	0	1	0

		Fair	Good	Poor
boroname	**spc_common**			
	ginkgo	0	3	0
	honeylocust	3	17	0
	pin oak	0	2	0
	red maple	0	1	0
	sawtooth oak	1	1	0
	silver linden	0	1	0
	swamp white oak	0	2	0
	sweetgum	0	1	0
	white oak	1	2	0
Brooklyn	**American linden**	2	17	1

In [5]:
```
#here we are able to print out a list of our boroughs
boroughs = list(set(trees['boroname']))
print(boroughs)

#below enables us to initial categorize by health, boro, spc_common among other criteri
trees_question1['total'] = trees_question1.sum(axis=1)
for column in list(trees_question1.columns):
    trees_question1[column] = (trees_question1[column]/trees_question1['total'])*100
trees_question1.head()
```

```
['Brooklyn', 'Queens', 'Manhattan', 'Staten Island', 'Bronx']
```

Out[5]:

		Fair	Good	Poor	total
boroname	**spc_common**				
Bronx	**American elm**	0.0	50.0	50.0	100.0
	Atlantic white cedar	0.0	100.0	0.0	100.0
	Callery pear	0.0	100.0	0.0	100.0
	Douglas-fir	0.0	100.0	0.0	100.0
	Japanese zelkova	0.0	100.0	0.0	100.0

In [6]:
```
#we desire to create a list to store data for our boroughs
trace_list=[]

#below we wish to create plot titles
borough_list = list(map(lambda x: str(x), boroughs))
```

In [9]:
```
#below we wish to select number of columns
cols=len(boroughs)
#we want to calculate number of rows
```

```
rows=1
fig = tools.make_subplots(rows=rows, cols=cols, subplot_titles=tuple(borough_list))
```

In [10]:

```
#iterate through boroughs
for borough in boroughs:
        for i in range(0,len(ourstatus)):
            trace = go.Bar(
            x = list(trees_question1.loc[borough].index),
            y = list(trees_question1.loc[borough][ourstatus[i]]),
            name = ourstatus[i],
            marker=dict(color=colors[i])
            )
            trace_list += [trace]



row_i = []
col_j = []
for i in range(1,rows+1):
    for j in range (1,cols+1):
        for n in range (1,4):
            row_i.append(i)
            col_j.append(j)

for i in range(0,len(trace_list)):
     fig.append_trace(trace_list[i], row_i[i],col_j[i])


fig['layout'].update(showlegend=False,height=1000, width=900, title='The proportion of


app = dash.Dash()

colors = {
    'background': '#999999',
    'text': 'black'
}

app.layout = html.Div(style={'backgroundColor': colors['background']}, children=[
    html.H1(
        children='Question #1',
        style={
            'textAlign': 'center',
            'color': colors['text']
        }
    ),
    html.Div(children='Proportion of trees in Good, Fair and Poor conditions', style={
        'textAlign': 'center',
        'color': colors['text']
    }),


    html.Div([
        dcc.Graph(figure=fig, id='my-figure')])
    ])
```

```
if __name__ == '__main__':
    app.run_server(debug=True)
```

```
Dash is running on http://127.0.0.1:8050/

 * Serving Flask app "__main__" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
An exception has occurred, use %tb to see the full traceback.

SystemExit: 1
```

Question #2 Are stewards(steward activity measured by the 'steward' variable) having an impact on the health of trees ?

In [39]:
```
#import the libraries to solve question #2

import pandas as pd
import numpy as np
import json
import plotly.offline as py
import plotly.graph_objs as go
from plotly import tools
import dash
import dash_core_components as dcc
import dash_html_components as html
```

In [40]:
```
# we are able to pull in our json data into url and pass to trees
url = 'https://data.cityofnewyork.us/resource/nwxe-4ae8.json'
trees1 = pd.read_json(url)
trees_question1 = trees[['spc_common','status','boroname']]
trees_question1['spc_common'].fillna('Unknown',inplace = True)
```

In [44]:
```
#here we will create  the columns that specify tree status
for status in set(trees_question1['status']):
    trees_question1[status] = np.where(trees_question1['status']==status,1,0)

trees_question1 = pd.DataFrame(trees_question1.groupby(['boroname','spc_common']).sum()
trees_question1.head(19)
```

Out[44]:

		Stump	Alive	Dead
boroname	**spc_common**			
Bronx	**American elm**	0	4	0
	Atlantic white cedar	0	1	0
	Callery pear	0	7	0
	Douglas-fir	0	1	0
	Japanese zelkova	0	2	0

		Stump	Alive	Dead
boroname	**spc_common**			
	Kentucky yellowwood	0	1	0
	London planetree	0	4	0
	Norway maple	0	23	0
	Sophora	0	8	0
	Unknown	1	0	3
	crepe myrtle	0	1	0
	ginkgo	0	3	0
	honeylocust	0	20	0
	pin oak	0	2	0
	red maple	0	1	0
	sawtooth oak	0	2	0
	silver linden	0	1	0
	swamp white oak	0	2	0
	sweetgum	0	1	0

In [35]:
```
#we wish to find our boroughs
boroughs = list(set(trees['boroname']))

trace_list_question2 =[]
```

In [46]:
```
#below we wish to create our plot titles
borough_list = list(map(lambda x: str(x), boroughs))

trees_question2 = trees[['spc_common','health','boroname','steward']]

trees_question2['spc_common'].fillna('Unknown',inplace = True)
trees_question2.dropna(inplace = True)
trees_question2[['steward','health']] = trees_question2[['steward','health']].apply(lam
trees_question2_cor = pd.DataFrame(trees_question2.groupby(['boroname','spc_common']).c
fig_question2 = tools.make_subplots(rows=1, cols=len(boroughs), subplot_titles=tuple(bo



boroughs = list(set(trees_question2['boroname']))
plants = list(set(trees_question2['spc_common']))

for borough in boroughs:
    trace = go.Bar(
            x = list(trees_question1.loc[borough].index),
            y = list(trees_question2_cor.loc[borough]['steward'][::2])
            )
    trace_list_question2 += [trace]

for i in range(len(boroughs)):
```

```
    fig_question2.append_trace(trace_list_question2[i], 1, i+1)



fig_question2['layout'].update(showlegend=False,height=500, width=1400, title='The prop




app = dash.Dash()

colors = {
    'background': '#b2b2b2',
    'text': '#111111'
}



app.layout = html.Div(style={'backgroundColor': colors['background']}, children=[
    html.H1(
        children='Question #2',
        style={
            'textAlign': 'center',
            'color': colors['text']
        }
    ),
    html.Div(children='The correlation between stewards and health of trees', style={
        'textAlign': 'center',
        'color': colors['text']
    }),

    html.Div([
        dcc.Graph(figure=fig_question2, id='my-figure')
])
    ])


if __name__ == '__main__':
    app.run_server(debug=True)
```

```
Dash is running on http://127.0.0.1:8050/

Dash is running on http://127.0.0.1:8050/

Dash is running on http://127.0.0.1:8050/

 * Serving Flask app "__main__" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
An exception has occurred, use %tb to see the full traceback.

SystemExit: 1
```

In [ ]: