# Fuel costs and other anlaysis by county

September 30, 2018

Fleet Fuel Management

```
<h2 align = 'Center'> -Rishabh Mulani, Krzysztof Dutka, Nanyi Yang</h2></html>
```

### 0.0.1 1.1 Package & file imports

```
In [35]: #Standard & special imports

         import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         import seaborn as sns
         import os
         pd.set_option('display.max_columns', 500)
         import plotly.plotly as py
         import plotly.figure_factory as ff
         import plotly
         plotly.tools.set_credentials_file(username='rmulani2', api_key='swfkuJXx1ywUCh6fWYBz')
         import ipywidgets
         from IPython.display import clear_output
         import bqplot.pyplot as bqplt
         from bqplot import *
         import plotly.graph_objs as go
         !jupyter nbextension enable --py widgetsnbextension
```

```
Enabling notebook extension jupyter-js-widgets/extension...
      - Validating: OK
```

### 0.0.2 1.2 Reading in the data

```
In [15]: #read in the csv files
         df_fuel_card = pd.read_csv('fuel_card_export.csv')
         df_sample = pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/master/minor
         df_fips_dict = pd.read_csv('Fips Codes.csv')
```

### 0.0.3 1.3 Initializing widgets

In [16]:
```
#Initialize widgets

pbar = ipywidgets.FloatProgress(total = len(df_fips_dict['CountyName']), value = 0)
lbl_load = ipywidgets.Label('Loading progess:')
lbl_load_percent = ipywidgets.Label('%')
hbox_loading = ipywidgets.HBox([lbl_load, pbar, lbl_load_percent])
```

### 0.0.4 1.4 Automating Functions for making the plot for a selected parameter on the counties map

In [17]:
```
#checking the shpae of the data
df_fuel_card.shape
```

Out[17]: (6570, 77)

In [18]:
```
def add_fips(df_aggregated):
    '''

        This is a function to add the FIPS column to any dataframe with the 'Merchant S
        columns FIPS codes are directly linked to counties and are required by the plot
        is being referenced

        params: df_aggregated - A dataframe which has been aggregated for a specific pa
    '''

    pbar.value = 0
    lbl_load_percent.value = '%'
    try:
        for county, state in df_fips_dict[['CountyName', 'State']].values:
            df_aggregated.loc[(df_aggregated['Merchant County'] == county) & (df_aggreg
            pbar.value+=1
            lbl_load_percent.value = str(int(pbar.value))+'%'
    #if the dataframe has County and State as indices, reset indices and try the functi
    except KeyError:
        df_aggregated.reset_index(inplace = True)
        #clear_output()

        display(hbox_loading)
        add_fips(df_aggregated)

    return df_aggregated
```

In [19]:
```
def build_plotting_df(selected_col, agg_func):
    '''
    A function that builds the converts the original dataframe 'df_fuel_card'
    '''
    if agg_func == 'Count':
```

```
            df_selected_param = pd.DataFrame(df_fuel_card[['Merchant State / Province', 'Me
        elif agg_func == 'Average':
            df_selected_param = pd.DataFrame(df_fuel_card[['Merchant State / Province', 'Me
        elif agg_func == 'Total':
            df_selected_param = pd.DataFrame(df_fuel_card[['Merchant State / Province', 'Me
        elif agg_func == 'Max':
            df_selected_param = pd.DataFrame(df_fuel_card[['Merchant State / Province', 'Me
        elif agg_func == 'Min':
            df_selected_param = pd.DataFrame(df_fuel_card[['Merchant State / Province', 'Me

        #Adding the 'FIPS' column to df_unit_cost
        df_selected_param = add_fips(df_selected_param)
        return df_selected_param
```

In [50]:
```
#Choose a parameter in place of 'Unit Cost' to plot on the counties map
def make_plot(param_to_plot, agg_func):
    plotting_df = build_plotting_df(param_to_plot, agg_func)

    #dropping those rows where the county value was incorrect
    plotting_df = plotting_df[np.isfinite(plotting_df['FIPS'])]
    #plotting_df.shape
    values = plotting_df[param_to_plot].round(decimals = 4)
    fips = plotting_df['FIPS'].tolist()

    fig = ff.create_choropleth(
        #scope of the map varies by the states mentioned in the dataset
        fips=fips, values=values, scope=plotting_df['Merchant State / Province'].unique
        state_outline = {'color': 'rgb(0,0,0)', 'width': 0.5},
        county_outline={'color': 'rgb(0,0,0)', 'width': 0.5}, round_legend_values=True,
        legend_title='', title= agg_func+' of '+param_to_plot+' ''by County: Illinois &
    )
    #clear_output()
    display(py.iplot(fig, filename='choropleth_california_and_surr_states_outlines'))
```

In [21]:
```
def onclick(event):
    clear_output()
    display(vBox)
    make_plot(dd_param.value, dd_aggfunc.value)

btn = ipywidgets.Button(description = 'Visualize!')
btn.on_click(onclick)
dd_param = ipywidgets.Dropdown()
dd_param.options = df_fuel_card.columns.tolist()
dd_aggfunc =  ipywidgets.Dropdown()
dd_aggfunc.options = ['Count', 'Min', 'Max', 'Total', 'Average']
vBox = ipywidgets.VBox([dd_param, dd_aggfunc, btn])
```

In [22]:
```
display(vBox)
```

```
VBox(children=(Dropdown(options=('Transaction Id', 'Card Number ID', 'Emboss Line 2 ID', 'Author
```

```
HBox(children=(Label(value='Loading progess:'), FloatProgress(value=0.0), Label(value='%')))
```

```
<plotly.tools.PlotlyDisplay object>
```

```
In [23]: # @ipywidgets.interact(Parameter = df_fuel_card.columns.tolist(), Aggregate_by = ['Coun
         # def make_selected_plotly(Parameter, Aggregate_by):
         #     make_plot(Parameter, Aggregate_by)
```

The count of transaction id by county shows us that our data is mainly about Champaign with 6306 transactions pertaining to Champaign county. So it makes sense to focus on the Champaign county only for further analysis.

## 0.1 Unit cost v/s transaction count

```
In [24]: pbar2 = ipywidgets.IntProgress(max = len(df_fuel_card['Merchant County'].unique()), val
         lbl_load2 = ipywidgets.Label('Loading progess:')
         lbl_load_percent2 = ipywidgets.Label('%')
         hbox_loading2 = ipywidgets.HBox([lbl_load2, pbar2, lbl_load_percent2])
```

### 0.1.1 Creating a list of dataftames by county

```
In [28]: df_dict = {}
         pbar2.value = 0
         display(hbox_loading2)
         for county in df_fuel_card['Merchant County'].unique():
             data = pd.DataFrame(df_fuel_card.loc[df_fuel_card['Merchant County'] == county])
             df_dict[county] = data
             pbar2.value+=1
             lbl_load_percent2.value = str((pbar2.value/pbar2.max)*100)+'%'
```

```
HBox(children=(Label(value='Loading progess:'), IntProgress(value=0, max=40), Label(value='100.0
```

```
In [47]: @ipywidgets.interact(County = df_dict.keys())
         def makeplot(County):
         #     bqplt.clear()
             df = None
             df = df_dict[County][['Transaction Id', 'Unit Cost', 'Merchant Brand']].groupby('Me
             df.columns = df.columns.droplevel(1)
             df.columns = ['Number of transactions', 'Average Unit Cost']

         #     fig = bqplt.figure
         #     def_tt = Tooltip(labels = df.index.tolist())
         #     bqplt.scatter(df['Number of transactions'],df['Average Unit Cost'], colors=['red'
```

```
#     def_tt.show_labels = False
#     bqplt.title(County+' Trend between transaction count and unit cost')
#     bqplt.xlabel('Transaction count')
#     bqplt.ylabel('Unit Cost')
#     bqplt.show()

    trace = go.Scatter(
    x = df['Number of transactions'],
    y = df['Average Unit Cost'],
    mode = 'markers',
    name = 'Transactions',
        marker = dict(
        size = 10,
        color = df['Average Unit Cost'], #set color equal to a variable
        colorscale='Viridis',
        showscale=True
        )
    )
    data = [trace]
    layout = dict(title = County+'- Average Unit costs by transaction count',
              yaxis = dict(zeroline = False),
              xaxis = dict(zeroline = False)
             )

    fig = dict(data=data, layout=layout)
    display(py.iplot(fig, filename='styled-scatter'))
```

```
interactive(children=(Dropdown(description='County', options=('CHAMPAIGN', 'EDGAR', 'FAYETTE', '
```

In [52]:
```
#Choose a parameter in place of 'Unit Cost' to plot on the counties map
def make_plot_2(param_to_plot, agg_func):
    plotting_df = build_plotting_df(param_to_plot, agg_func)

    #dropping those rows where the county value was incorrect
    plotting_df = plotting_df[np.isfinite(plotting_df['FIPS'])]
    #plotting_df.shape
    values = plotting_df[param_to_plot].round(decimals = 4)
    fips = plotting_df['FIPS'].tolist()

    fig = ff.create_choropleth(
        #scope of the map varies by the states mentioned in the dataset
        fips=fips, values=values, scope=plotting_df['Merchant State / Province'].unique
        state_outline = {'color': 'rgb(0,0,0)', 'width': 0.5},
        county_outline={'color': 'rgb(0,0,1)', 'width': 0.5}, round_legend_values=True,
        legend_title='', title= agg_func+' of '+param_to_plot+' ''by County: Illinois &
    )
    #clear_output()
    display(py.iplot(fig, filename='average unit cost by county'))
```

```
In [53]: make_plot_2('Unit Cost', 'Average')
```

```
HBox(children=(Label(value='Loading progess:'), FloatProgress(value=0.0), Label(value='%')))
```

```
<plotly.tools.PlotlyDisplay object>
```