
Website (https://www.coleridgeinitiative.org)

source to be updated when notebook added to GitHub

Table of Contents

JupyterLab contains a dynamic table of contents that can be accessed by clicking the last of the six icons on the left-hand sidebar.

Introduction to Data Visualization

As discussed in the lecutre, visualization is very effective at conveying information about big, complex datasets. In this module, you will learn to quickly and flexibly make a wide series of visualizations for exploratory data analysis and communicating to your audience. This module contains a practical introduction to data visualization in Python and covers important rules that any data visualizer should follow.

Learning Objectives

- Become familiar with a core base of data visualization tools in Python - specifically `matplotlib` and `seaborn`
- Begin exploring what visualizations are going to best reveal various types of patterns in your data
- Learn more about our primary datasets data with exploratory analyses and visualizations

Python Setup

In this notebook we will use the following Python packages:

- `pyathenajdbc` : for interfacing with the Athena data query service to pull our data from the servers
- `pandas` : for loading and transforming our data into formats suitable for visualization
- `matplotlib` : Python's widely-used, standard visualization library
- `seaborn` : Python's higher-level visualization library with some good built-ins
- `calendar` : Python's libary for the use of functions related to the calendar
- `numpy` : Python's widely-used library for mathematical operations on matrices and arrays

In [ ]:
```
# pandas-related imports
import pandas as pd

# Athena interaction imports
from pyathenajdbc import connect

# visualization packages
import matplotlib.pyplot as plt
import seaborn as sn

#numpy-related imports
import numpy as np

#date-time related packages
import calendar

# so images get plotted in the notebook
%matplotlib inline
```

Load the Data

The first parameter is the connection to the database. To create a connection we will use the `Pyathenajdbc` package and tell it which database we want to connect to, just like in DBeaver.

Establish a Database Connection

In [ ]:
```
#Establish a connection with the iri_usda database to access the data
conn = connect(s3_staging_dir = 's3://usda-iri-2019-queryresults/',
               region_name = 'us-gov-west-1',
               LogLevel = '0',
               workgroup = 'workgroup-iri_usda')
```

Steps for Producing Visualizations

- Collect the necessary data
- Process and clean for visualization purposes
- Determine the cohort you wish to visualize
- Determine the appropriate visualization
- Visuzalize using python tools

For the purpose of these visualization excersizes, we will be using some code cells from the sample project templates. In particular, we will use visualizations to break down the cost of 100% whole wheat bread in different ways.

In the first two visualizations, we will analyze temporal changes of the average price of 100% whole wheat bread. But first, we will need to follow the prior steps before we produce this visualization. Thus, we will utiliize the `pd_pos_all` table to collect the upc codes, descriptions and flavors of all bread products. From there, we can identify upc codes pertaining to just 100% whole wheat bread to filter the trip data.

> `pd_pos_all` contains a higher level of information on products, as opposed to the in-depth descriptions in the `pd_master_all` table.

In [ ]:
```
#Develop a query to get data related to all bread products
bread_query = """
SELECT distinct upc, flavor, upcdesc
from iri_usda.pd_pos_all
where product in ('FRESH BREAD','HAMBURGER AND HOT DOG BUNS','PITA BREA
D',
    'BAGELS/BIALYS','BREAD','ROLL','BUN'
    ,'BAGEL')
    and category = 'FRESH BREAD & ROLLS'
    AND year = '2017'
"""

#Write out the query as a pandas df
bread_df = pd.read_sql(bread_query, conn)
```

In [ ]:
```
#Perform the filter for just whole wheat products
ww_df = bread_df[bread_df.flavor.str.contains('100% WHOLE WHEAT')]
```

In [ ]:
```
#Create a list of the distinct whole wheat product UPCs (will save for l
ater)
ww_upc_list = ww_df.upc.unique().tolist()
```

In the cell below, wic households are being indentified from the `project_q2_cohort` table. `project_q2_cohort` is a table that contains all demographic data on wic eligible and participating households who have sufficient purchasing data in 2017.

Please note that you can access self-created tables from the `iri_usda_2019_db` database.

In [ ]:
```
#Develop a query to get data related to exclusively WIC participating ho
useholds

wic_hh_query = """
SELECT distinct panid
from iri_usda_2019_db.project_q2_cohort
where wic_june = 1;"""

#Write out the query as a pandas df
wic_hh_df = pd.read_sql(wic_hh_query, conn)
```

In [ ]:
```
#Create a list of the distinct WIC participating household IDs (will sav
e for later)
wic_hh_list = wic_hh_df.panid.unique().tolist()
```

In the cell below, trips for 2017 where an individual `panid` using are being indentified from the `trip_all` table.

In [ ]:
```
# Develop a query to get data related to purchasing information of
# iri participants in 2017 using just WIC funds
trip_query = """
SELECT distinct purdate, panid, mop, upc, dollarspaid, quantity
from iri_usda.trip_all
where year = '2017'
and mop = '7'
limit 5000000;"""
trip_df = pd.read_sql(trip_query,conn)
```

Now that we have our data collected, we can now merge the datasets to get a data frame with wic receiving households and the date they have purchased whole wheat products using WIC funds

In [ ]:
```
#Join our purchases df to our wic household df to get purchase informati
on for just wic households
wic_trips_df = pd.merge(wic_hh_df,trip_df, on = 'panid')
ww_trip_df = pd.merge(wic_trips_df, ww_df[['upc','flavor','upcdesc']], o
n='upc')
ww_trip_df.head()
```

In [ ]:
```
#Create a month column by selecting just the month from the purchase dat
a column
ww_trip_df['month'] = ww_trip_df['purdate'].apply(lambda x: x.month)

#Create a dollars_per_product column by dividing the dollars paid by the
quantity of the product purchased
ww_trip_df['dollars_per_product'] = ww_trip_df['dollarspaid']/ww_trip_df
['quantity']
```

In [ ]:
```
#verify month and dollars_per_product columns are created
ww_trip_df.columns
```

In [ ]:
```
#subset our data to just month and dollarspaid
ww_purchase_sub = ww_trip_df[['month','dollars_per_product']]

#group our data to get the average price of 100% whole wheat products pe
r each month
ww_purchases_mean = ww_purchase_sub.groupby(['month']).mean()
ww_purchases_mean.head()
```

In [ ]:
```
#Reset the index to use the month column (was initially the index due to
the group by month clause in the cell above)
ww_pur_mean_cleaned = ww_purchases_mean.reset_index()

#convert month number to month name
ww_pur_mean_cleaned['month'] =  ww_pur_mean_cleaned['month'].apply(lambd
a x: calendar.month_name[x])
```

It is time to visualize. We will be using bar and line plots to visualize average cost of whole wheat products purchased through WIC. This is due to both being able to visualize data over time.

In [ ]:
```
# Create our bar plot
# start each figure with fig = plt.figure() to create the space
# for your visualization
# <text>\n<text> as seen in 'Bread\namong' in
# the title indidcates a line break so your title can
# fit above your graph
fig = plt.figure()
ax = ww_pur_mean_cleaned.plot.bar(x='month',y='dollars_per_product',
     title = 'Average Monthly Spending on Whole Wheat Bread\namong WIC H
ouseholds per Trip',
     legend = False)
#set the x-axis label
ax.set_xlabel('Month')

#set the y-axis label
ax.set_ylabel('Dollars per Whole Wheat Product Purchased');
```

Bar plots are great for visualizing over time. However, seeing the difference in values over time may be difficult if values are so similar, so we can visualize the same data using a line plot as well.

In [ ]:
```
#create our line plot
fig = plt.figure()
ax = ww_pur_mean_cleaned.plot(x='month',y='dollars_per_product',
                              title = 'Average Monthly Spending on Who
le Wheat Bread\namong WIC Households per Trip',
                             legend = False, rot = 90)
ax.set_xlabel('Month')
ax.set_ylabel('Dollars per Whole Wheat Product Purchased')
```

We will now look average amount spent, per trip, on both 100% whole wheat and other bread products. To do that, we plan to use box and whisker plots. These plots are great for visualizing descriptive stats about the data.

The box have the upper and lower quartile bounds and a median line found within the box. The median line indicates the median value found within the data. They have 'whiskers' that indicate the upper and lower bounds of the data (excluding statistical outliers).

In [ ]:
```
#Create a df of bread purchases
bread_trip_df = pd.merge(wic_trips_df, bread_df[['upc','flavor','upcdes
c']], on='upc')

#Create boolean (1 meaning yes, 0 meaning no) for products that are 100%
whole wheat
bread_trip_df['whole_wheat_flag'] = bread_trip_df.flavor.str.contains("1
00% WHOLE WHEAT")
```

In [ ]:
```
#Create a dollars_per_product column by dividing the dollars paid by the
quantity of the product purchased
bread_trip_df['dollars_per_product'] = bread_trip_df['dollarspaid']/brea
d_trip_df['quantity']
```

In [ ]:
```
bread_trip_df['product'] = bread_trip_df['whole_wheat_flag'].replace(Tru
e,'100% Whole Wheat Product').replace(False,'Other Bread Product')
```

In [ ]:
```
#subset to just the flag and the dollars spents on the bread products
bread_trip_df_sub =  bread_trip_df[['product','dollars_per_product']]
```

In [ ]:
```
#create a boxplot
fig = plt.figure()
ax = bread_trip_df_sub.boxplot(by='product')

#set the title
ax.set_title('Boxplots of Prices of Bread Poducts\nwith and without the
 Whole Wheat Flag')

#Create axes same as before
ax.set_xlabel("Product")
ax.set_ylabel("Dollars per Whole Wheat Product Purchased")

#removes the subtitle
plt.suptitle("")
```

The boxplot above is descriptive. However, there is plenty of clutter that can be distracting. Let's remove outliers and grid lines that appear in the plot to make our results more clear.

In [ ]:
```
#Create the same boxplot without the grid or outliers
fig = plt.figure()
ax = bread_trip_df_sub.boxplot(by='product', grid=False, showfliers=Fals
e)
ax.set_title('Boxplots of Prices of Bread Poducts\nwith and without the
 Whole Wheat Flag\nWithout Outliers')
ax.set_xlabel("Product")
ax.set_ylabel("Dollars per Whole Wheat Product Purchased")
plt.suptitle("")
```

We have now gone through some examples of visualizing values over time and descriptive statistics. One last thing we should cover is choosing the best visualization for numeric data.

Visualize average expenditures by household size

Here we will compare two visualizations of the average amount of money spent on 100% whole wheat bread in 2017 across families of different sizes.

In [ ]:
```
# reminder of the DataFrame of UPC codes for 100% whole wheat bread
# we created above
ww_df.head()
```

In [ ]:
```
# put all the UPC codes as a single Python string
# (but long list of values) to use in our SQL query
ww_upcs = ','.join(["'"+upc+"'" for upc in ww_df['upc'].unique()])
```

In [ ]:
```
# print what that "object" looks like
ww_upcs
```

In [ ]:
```
# summarize bread purchases of our cohort in 2017

query = '''
select demo.hhsize, demo.panid,
    sum(trip.dollarspaid - trip.coupon) as total_spent_ww
FROM iri_usda.demo_all demo
join iri_usda.trip_all trip
on trip.panid = demo.panid
where demo.year = '2017'
    AND demo.panid IN (SELECT panid FROM iri_usda_2019_db.panid_expense)
    AND trip.year = '2017'
    AND trip.panid IN (SELECT panid FROM iri_usda_2019_db.panid_expense)
    AND trip.upc IN ({})
group by demo.hhsize, demo.panid
'''.format(ww_upcs)
df = pd.read_sql(query, conn)
```

Key points in above query:

1. get `hhsize` from the demographic table for just our WIC subset cohort
2. subset the `trip` table to just the year, list of panid in our cohort, and UPCs that are 100% WHOLE WHEAT
3. Sum total purchases across the year for each household

In [ ]:
```
# view what that data looks like
df.head()
```

In [ ]:
```
# calculate expenditure per person
df['ww_spent_person'] = df['total_spent_ww']/df['hhsize']
```

In [ ]:
```
# summary stats of 100% whole wheat bread expenditures by HH size
df.groupby('hhsize')['ww_spent_person'].describe()
```

In [ ]:
```
# Perform a groupby to get the mean amount spent on
# 100% whole wheat bread by household size
hh_ww_pur_agg = df.groupby('hhsize').mean()
```

In [ ]:
```
#reset the index after the groupby, as done previously
hh_ww_pur_cleaned = hh_ww_pur_agg.reset_index()
```

In [ ]:
```
#plot the pie plot
fig = plt.figure()
pie_plot = hh_ww_pur_agg.plot.pie(y='ww_spent_person', figsize =(10,10))
pie_plot.set_title('Average Amount of Money Spent per Person on 100% Who
le Wheat Bread by Household Size in 2017')
pie_plot.set_ylabel('Dollars per Person')
```

As we can see, the visualization above is not all that telling. It will be difficult to determine how much a person, on average, in an eight-person household spends a 100% whole wheat bread by interpreting the arc length or angle size. A bar plot is a much more appropriate visualization for this.

In [ ]:
```
#Create the bar plot
fig = plt.figure()
ax = hh_ww_pur_cleaned.plot.bar(x='hhsize',y='ww_spent_person',
     title = 'Average Amount Spent on 100% Whole Wheat Bread\nby Househo
ld Size in 2017',
                                     legend = False, rot = 90)
ax.set_ylabel('Average Dollars Spent per Person')
ax.set_xlabel('Household Size')
```