Automotive Data Wrangling using Jupyter Notebooks, Python and Brunel Visualization

The purpose of this lab

By now you have used the data wrangling tools of Watson Studio and are familiar with a somewhat arduous, yet necessary task of cleansing, preparing and transforming your data. The activities below are typically done by the Data Engineer with close consultation with the Data journalist and the data scientist. In this lab, the Data Engineer performs many of the tasks that you completed in the prior exercises, except that it is done using Jupyter Notebooks.

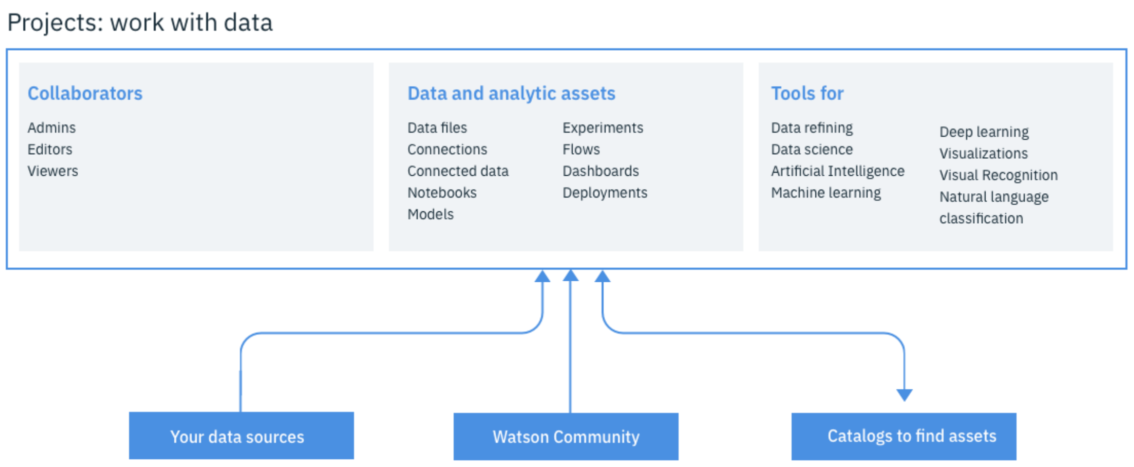
Objectives

The objective of this lab is to complete the data cleansing and transformation steps that you performed in the earlier labs, except using Jupyter Notebooks as your development environment.

- Create a new project in Watson Studio
- Import into Pandas dataframe the automobile data set from the prior exercise
- Use open source technologies, such as Brunel to visualize your data

Overview

Watson Studio provides you with the environment and tools to solve your business problems by collaboratively working with data. You can choose the tools you need to analyze and visualize data, to cleanse and shape data, to ingest streaming data, or to create, train, and deploy machine learning models. This illustration shows how the architecture of Watson Studio is centered around the project. A project is where you organize your resources and work with data.



Prerequisites

These are the most important resources in a project:

- Collaborators are the team who works with the data. Three roles provide [different permissions](#).
- Data assets point to your data. Here's what you can do to prepare your data:
    - Access data from connections to your cloud or on-premises data sources
    - Access data from your organization's catalogs
    - Upload files to the project's object storage
    - Cleanse and shape data with the Data Refinery tool

- Analytical assets and tools are how you derive insights from data. You customize your project with the tools you need. Here's what you can do to analyze your data:
    - Analyze data with Jupyter notebooks or RStudio.
    - Build, train, test, and deploy machine learning and deep learning models.
    - Run deep learning model experiments in parallel with neural networks.
    - Classify images by training deep learning models to recognize image content.
    - Create and share dashboards of data visualizations without coding.
    - Classify text by training a model to classify text according to classes you define.

You can also bring in data and analytic assets from the IBM Watson Community.

Section 1. Getting Started with Watson Studio

By now you have already registered with IBM Cloud and applied your promo code. Let's begin our journey:

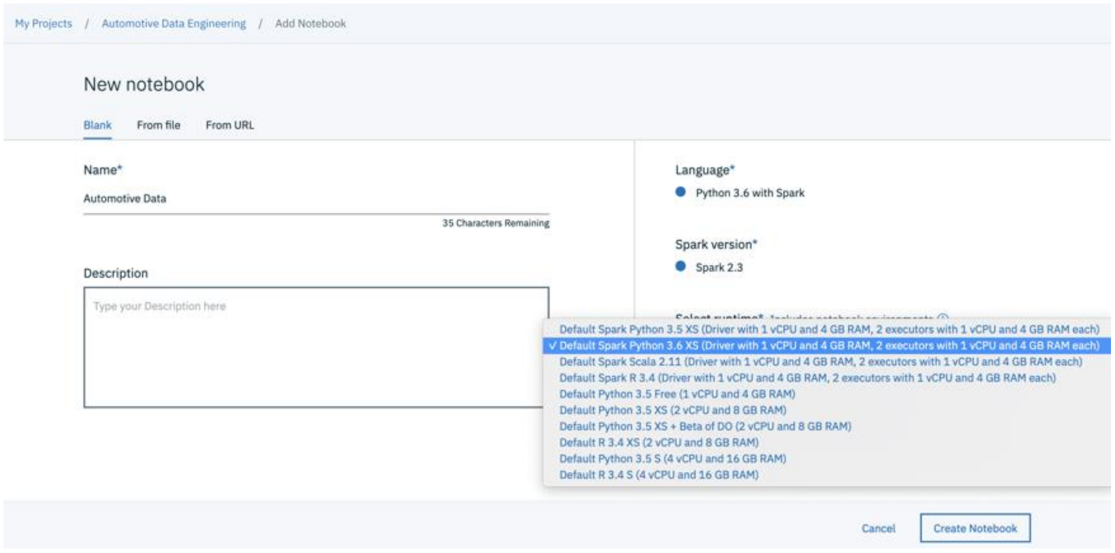
1. Login into IBM Cloud: https://console.bluemix.net/catalog/

   Note: Ensure that you have the promocode applied.

First time accessing Watson Studio	Returning to Watson Studio
Follow either one of the paths outlined below depending on whether you are new to Watson Studio or that you have already provisioned the service and are about to work with projects	
a. Click the **Catalog** tab and remove the **label:lite** filter. b. Search for the **Watson Studio** service and click that tile. c. Click **Create**. d. Click the **Get Started** button and when Done, click **Get Started**. e. Click the **New Project** tile. f. Select the **Complete** tile. g. Click **OK**. h. Specify a name. In this example, it is **Automotive data**. i. Click **Add** to add a storage. j. While the Lite plan is selected, click **Create**. k. In the Confirm Creation dialog, select the default region. l. Click **Refresh**. m. Now you have a cloud object storage available to you, click **Create**. n. Continue with the steps below.	a) Access the **Dashboard** from the menu icon in the top left corner b) Search for the previously created Watson Studio service and open it. c) Click the **Get Started** button. d) Open the latest existing project, for example: **Automotive Data engineering**. e) Continue with the steps below.

Section 2. Starting with the Jupyter books

1. From the **Add to project** drop-down list (top right), select **Notebook**.
2. Again, specify a new Notebook name; for example, Automotive Data and choose Python 3.6 from the drop-down list and click **Create Notebook**.



My Projects / Automotive Data Engineering / Add Notebook

New notebook

Blank | From file | From URL

Name*: Automotive Data

35 Characters Remaining

Description: Type your Description here

Language*: ● Python 3.6 with Spark

Spark version*: ● Spark 2.3

Select runtime*

- Default Spark Python 3.5 XS (Driver with 1 vCPU and 4 GB RAM, 2 executors with 1 vCPU and 4 GB RAM each)
- ✓ Default Spark Python 3.6 XS (Driver with 1 vCPU and 4 GB RAM, 2 executors with 1 vCPU and 4 GB RAM each)
- Default Spark Scala 2.11 (Driver with 1 vCPU and 4 GB RAM, 2 executors with 1 vCPU and 4 GB RAM each)
- Default Spark R 3.4 (Driver with 1 vCPU and 4 GB RAM, 2 executors with 1 vCPU and 4 GB RAM each)
- Default Python 3.5 Free (1 vCPU and 4 GB RAM)
- Default Python 3.5 XS (2 vCPU and 8 GB RAM)
- Default Python 3.5 XS + Beta of DO (2 vCPU and 8 GB RAM)
- Default R 3.4 XS (2 vCPU and 8 GB RAM)
- Default Python 3.5 S (4 vCPU and 16 GB RAM)
- Default R 3.4 S (4 vCPU and 16 GB RAM)

You are now in your notebook and ready to start working. When you use a notebook in Watson Studio, you can run a cell only by selecting it, then going to the toolbar and clicking on the Run Cell **(▸)** button. When a cell is running, an **[*]** is shown beside the cell. Once the cell has finished the asterisks is replaced by a number.

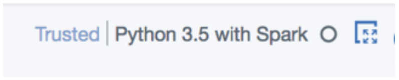
If you don’t see the Jupyter toolbar showing the Run Cell **(▸)** button and other notebook controls, you are not in edit mode. Go to the dark blue toolbar above the notebook and click the edit (pencil) icon.

The Brunel Visualization language makes it easy to build interactive charts and diagrams that you can deploy rapidly. This notebook contains the steps and code to get you started with visualizing data with Brunel.

Some familiarity with Python is recommended. This notebook runs on Python 3 with Spark.

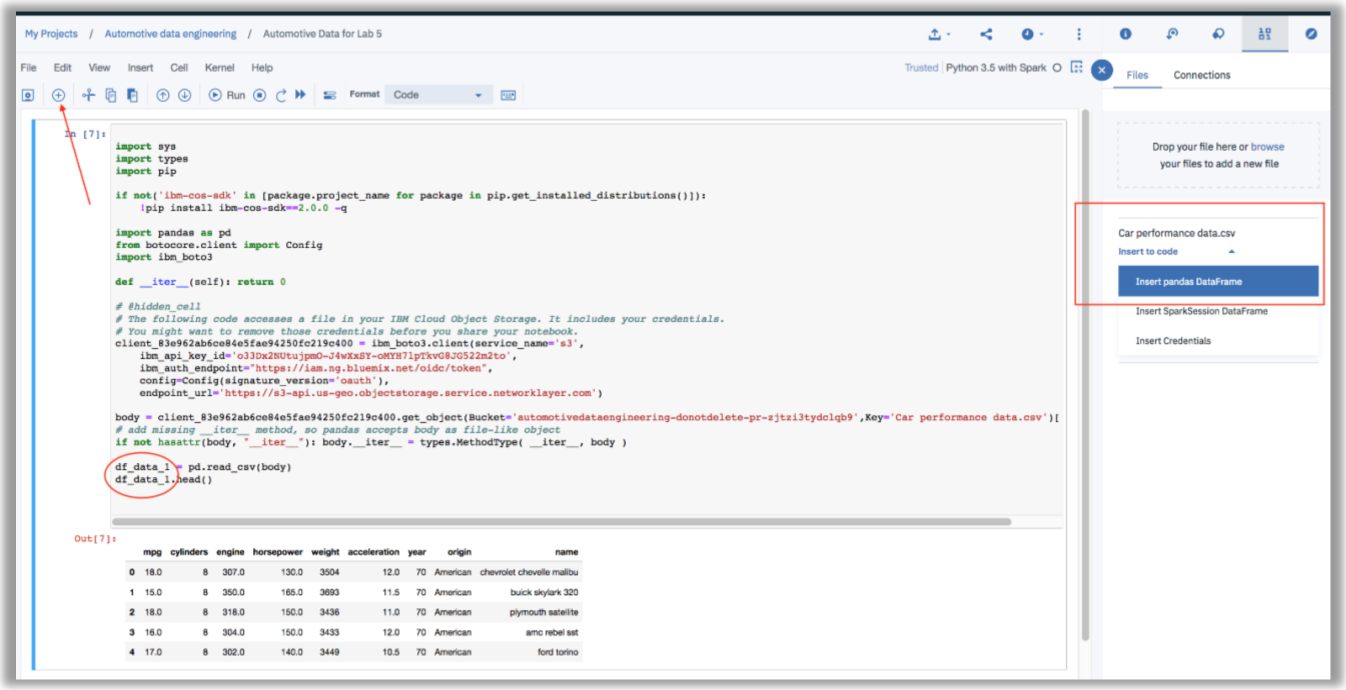
You will use data about cars to graph the relationships between various properties, for example, how horsepower affects gas mileage. The cars data set was used for the 1983 American Statistical Association Data Exposition. This data set was collected by Ernesto Ramos and David Donoho and obtained from StatLib.

3. Ensure that the kernel is trusted and running. Click the untrusted link and change it to trusted:



4. The first step is to obtain the data. Click **Find add data** .
5. From the **Files** section, select a csv file; for example, the **Car performance data.csv**.
6. Select **Insert pandas dataframe** option.

7. Click Run (▸). Allow enough time for the kernel to complete its function.


My Projects / Automotive data engineering / Automotive Data for Lab 5
File Edit View Insert Cell Kernel Help
Trusted Python 3.5 with Spark
Run
Format Code
Files Connections
Drop your file here or browse your files to add a new file
Car performance data.csv
Insert to code
Insert pandas DataFrame
Insert SparkSession DataFrame
Insert Credentials

```
In [7]:
import sys
import types
import pip

if not('ibm-cos-sdk' in [package.project_name for package in pip.get_installed_distributions()]):
    !pip install ibm-cos-sdk==2.0.0 -q

import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share your notebook.
client_83e962ab6ce84e5fae94250fc219c400 = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='o33Dx2NUtujpmO-J4wXxSY-oMYH7lpTkvG8JG522m2to',
    ibm_auth_endpoint="https://iam.ng.bluemix.net/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3-api.us-geo.objectstorage.service.networklayer.com')

body = client_83e962ab6ce84e5fae94250fc219c400.get_object(Bucket='automotivedataengineering-donotdelete-pr-zjtzi3tydclqb9',Key='Car performance data.csv')[
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

df_data_1 = pd.read_csv(body)
df_data_1.head()
```

```
Out[7]:
   mpg  cylinders  engine  horsepower  weight  acceleration  year  origin                      name
0  18.0         8   307.0       130.0    3504          12.0    70  American  chevrolet chevelle malibu
1  15.0         8   350.0       165.0    3693          11.5    70  American           buick skylark 320
2  18.0         8   318.0       150.0    3436          11.0    70  American          plymouth satellite
3  16.0         8   304.0       150.0    3433          12.0    70  American               amc rebel sst
4  17.0         8   302.0       140.0    3449          10.5    70  American                 ford torino
```

8. Click **Insert Cell Below**. Also note that the name of your dataframe is now: df_data_1. You will need this name (and yes it will change to df_data_2 and basically n+1 each time you run the import) to insert within some of the code snippets below.

9. Install and import Brunel. Enter the following command and click Run (▸). Allow for the star in the cell mark to turn into a number. This could take upwards of three to five minutes.

```
!pip install brunel==2.3
import brunel
```

Section 3. Visualize the data

You'll create some charts and diagrams with Brunel commands.
The basic format of each call to Brunel is simple. Whether the command is a single line or a set of lines, the commands are concatenated together, and the result interpreted as one command.
Here are some of the rules for using Brunel that you'll need in this notebook:

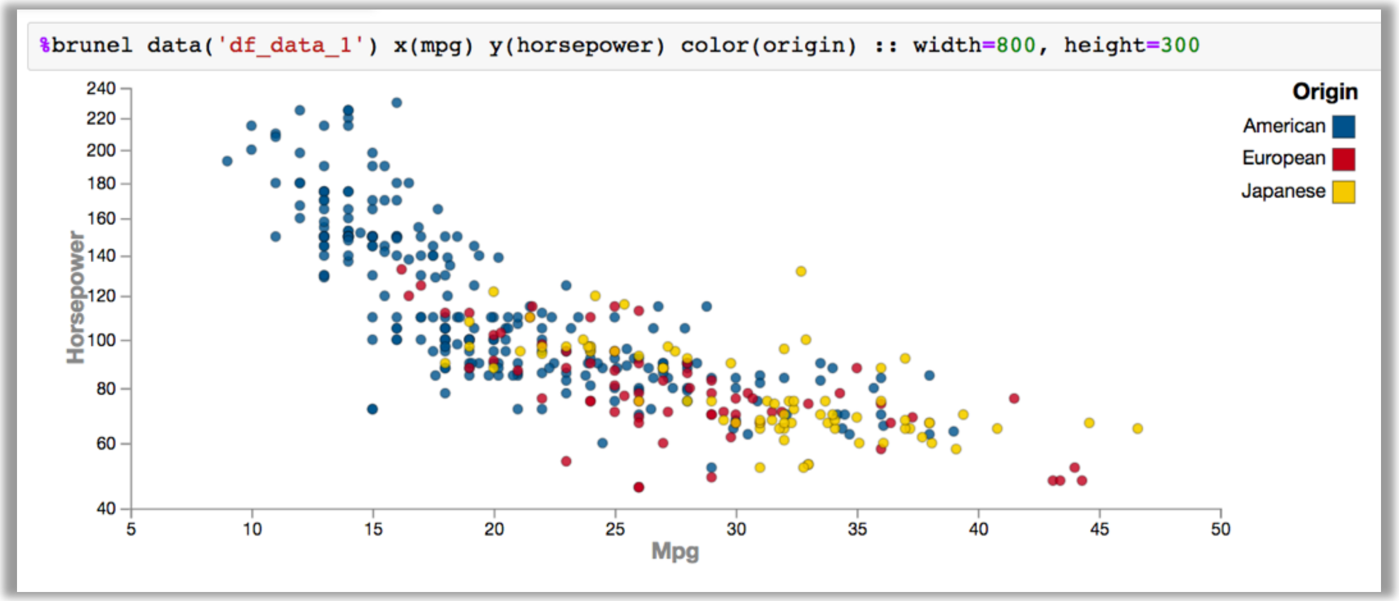
- **DataFrame**: Use the data command to specify the pandas DataFrame.
- **Chart type**: Use commands like chord and treemap to specify a chart type. If you don't specify a type, the default chart type is a scatterplot.
- **Chart definition**: Use the x and y commands to specify the data to include on the x-axis and the y-axis.
- **Styling**: Use commands like color, tooltip, and label to control the styling of the graph.
- **Size**: Use the width and height key-value pairs to specify the size of the graph. The key-value pairs must be preceded with two colons and separated with a comma, for example: :: width=800, height=300

See detailed documentation on the Brunel Visualization language at **Introduction to Brunel**.

Run the cells to show the relationship between the miles per gallon and the horsepower of the vehicles in a scatter plot. The color identifies the origin of the vehicles.

1. In a new cell enter the command below (copy/paste) and run the cell (one line). In this example, your dataframe name is: **df_data_1**

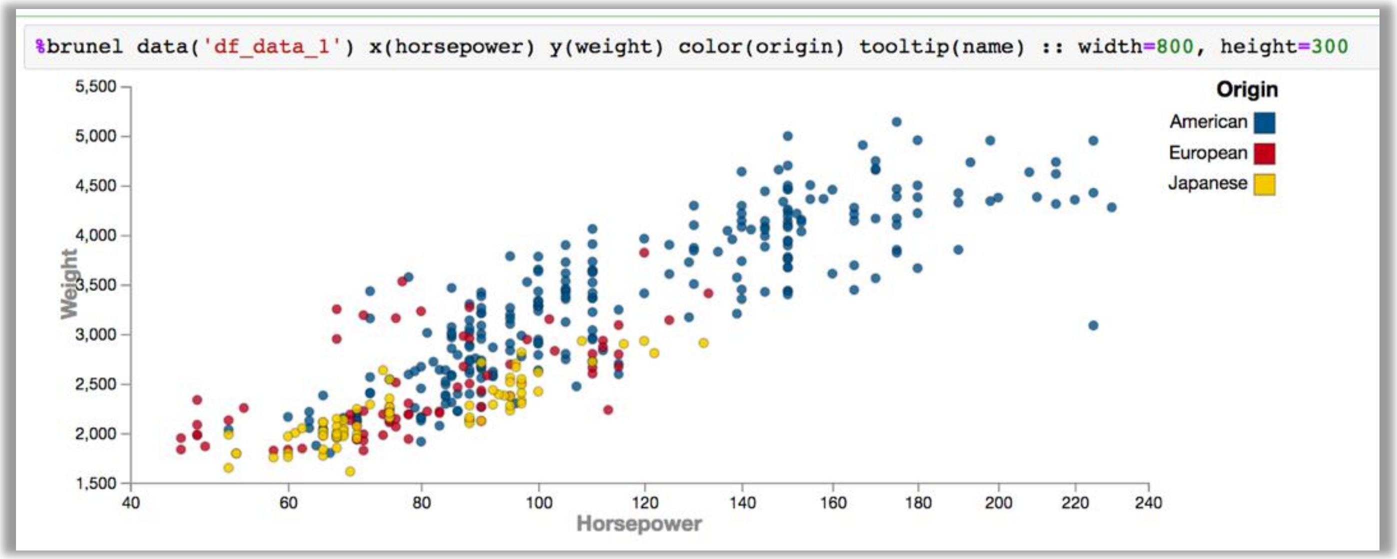
```
%brunel data('<enter the dataframe name here>') x(mpg) y(horsepower) color(origin) :: width=800,
height=300
```



```
%brunel data('df_data_1') x(mpg) y(horsepower) color(origin) :: width=800, height=300
```

2. In a new cell enter the command below (copy/paste) and run the cell (one line). In this example, your dataframe name is: df_data_1

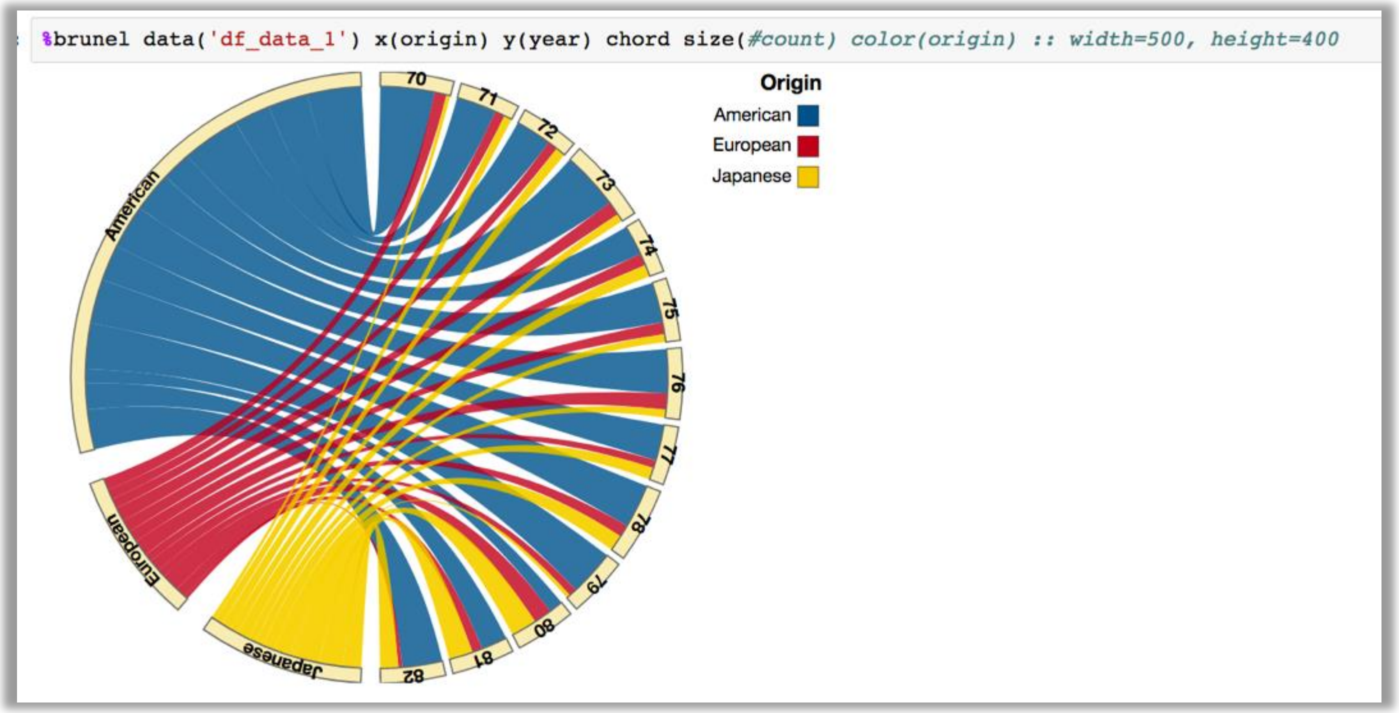
```
%brunel data('df_data_1') x(horsepower) y(weight) color(origin) tooltip(name) :: width=800, height=300
```



```
%brunel data('df_data_1') x(horsepower) y(weight) color(origin) tooltip(name) :: width=800, height=300
```

3. In a new cell enter the command below (copy/paste) and run the cell (one line). In this example, your dataframe name is: df_data_1

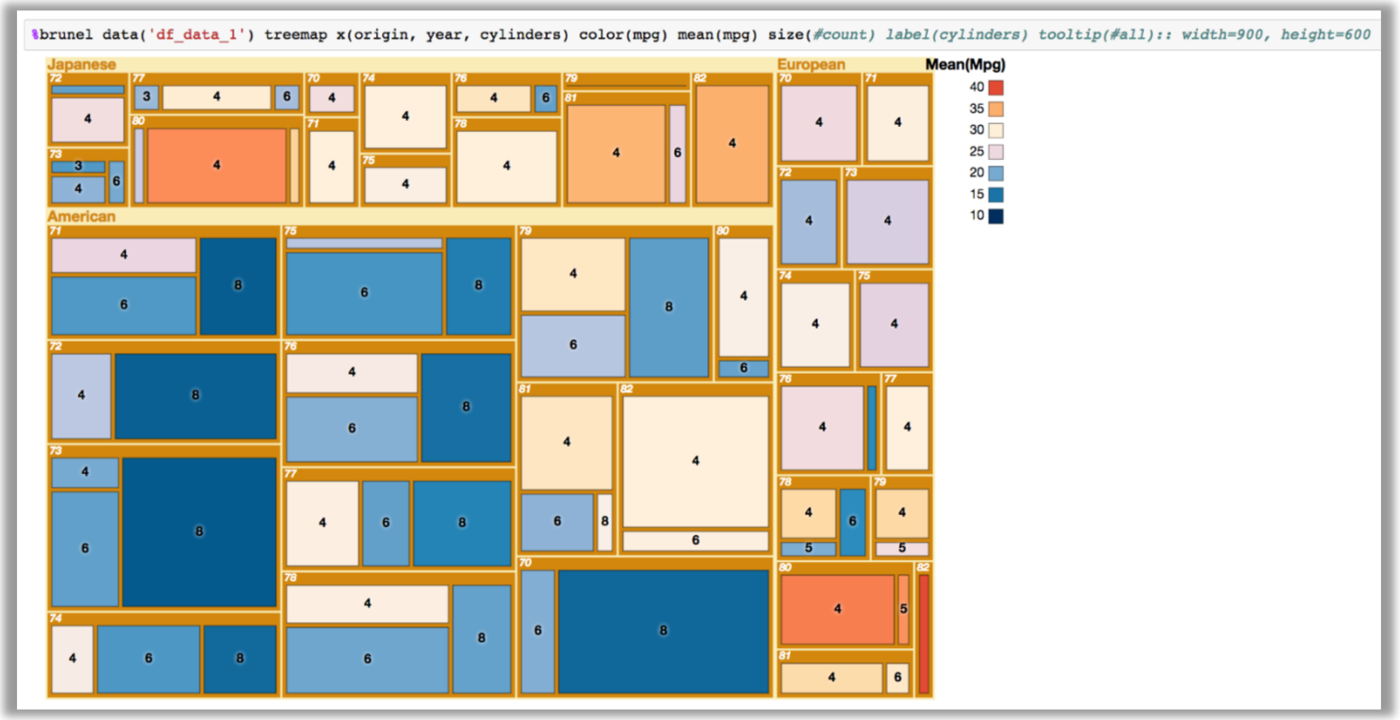
```
%brunel data('df_data_1') x(origin) y(year) chord size(#count) color(origin) :: width=500, height=400
```



```
%brunel data('df_data_1') x(origin) y(year) chord size(#count) color(origin) :: width=500, height=400
```

4. In a new cell enter the command below (copy/paste) and run the cell (one line). In this example, your dataframe name is: df_data_1

```
%brunel data('df_data_1') treemap x(origin, year, cylinders) color(mpg) mean(mpg) size(#count)
label(cylinders) tooltip(#all):: width=900, height=600
```



```
%brunel data('df_data_1') treemap x(origin, year, cylinders) color(mpg) mean(mpg) size(#count) label(cylinders) tooltip(#all):: width=900, height=600
```

You explored different types of charts and formatting and learned how you can use the pandas DataFrame to refine your charts. Try changing the formatting of these charts, or creating your own.