What is Plotnine?

The `plotnine` python library brings the power of R's ggplot2 to Python. Gain access to functions like:

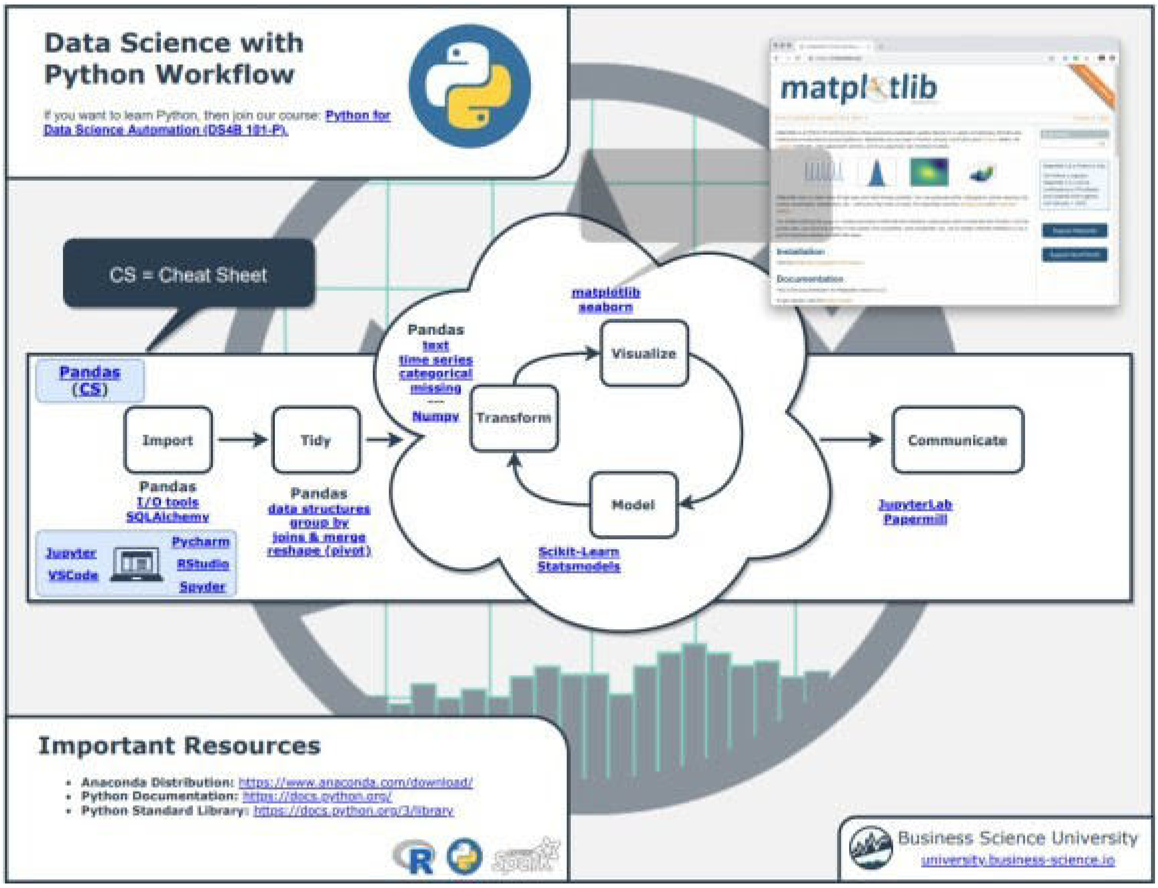
- `ggplot()` – Make the plot canvas (layout).
- `aes()` – Map `pandas` DataFrame columns to the plot aesthetics (x, y, color, fill, etc).
- **Geometries** – Add geometry layers including `geom_point()`, `geom_smooth()`.
- And more!

Before we get started, get the Python Cheat Sheet

`Plotnine` is great for data visualization in Python if you are coming from an R background. But, you might want to explore documentation for the entire Python Ecosystem (`pandas`, `plotnine`, `plotly`, and many more libraries). I'll use the Ultimate Python Cheat Sheet.

Ultimate Python Cheat Sheet:

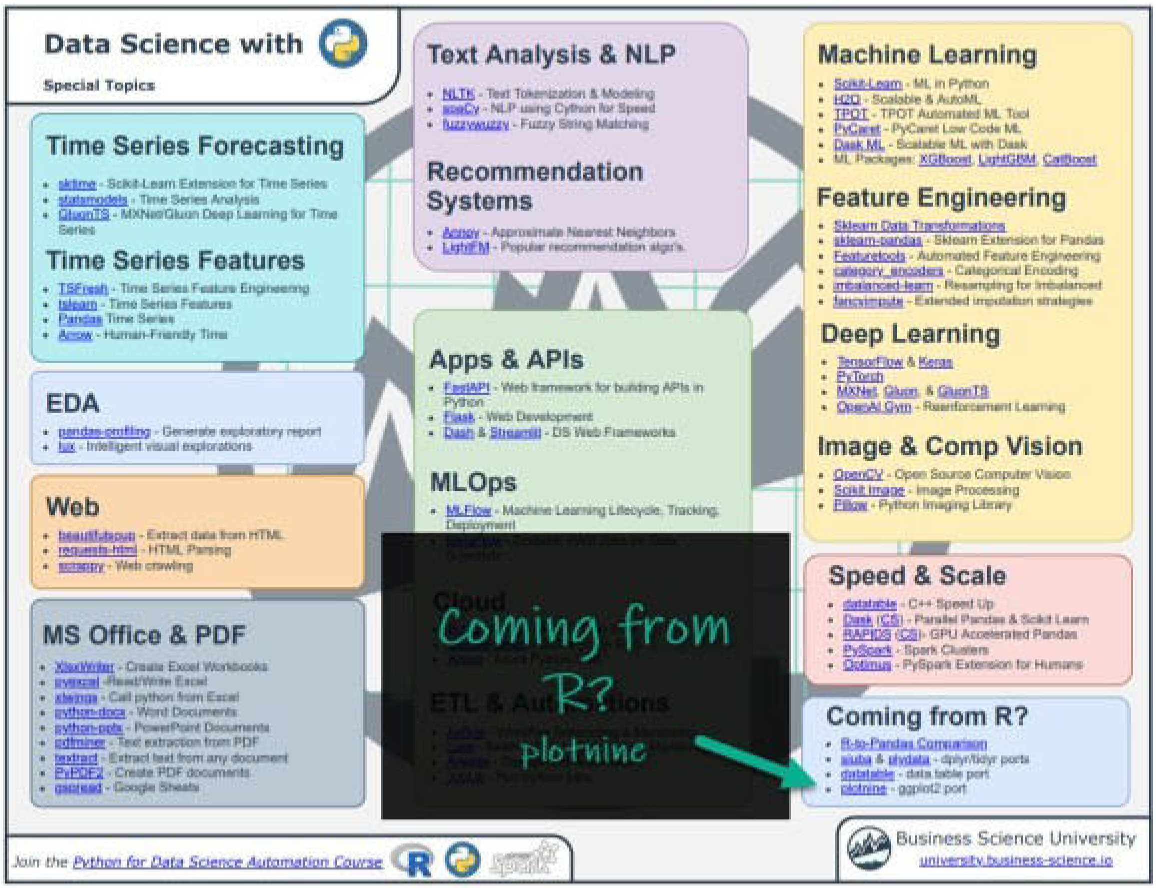
First, Download the Ultimate Python Cheat Sheet. This gives you access to the entire Python Ecosystem at your fingertips via hyperlinked documenation and cheat sheets.



(Click image to download)

If you’re coming from R, navigate to “Coming From R?” Section

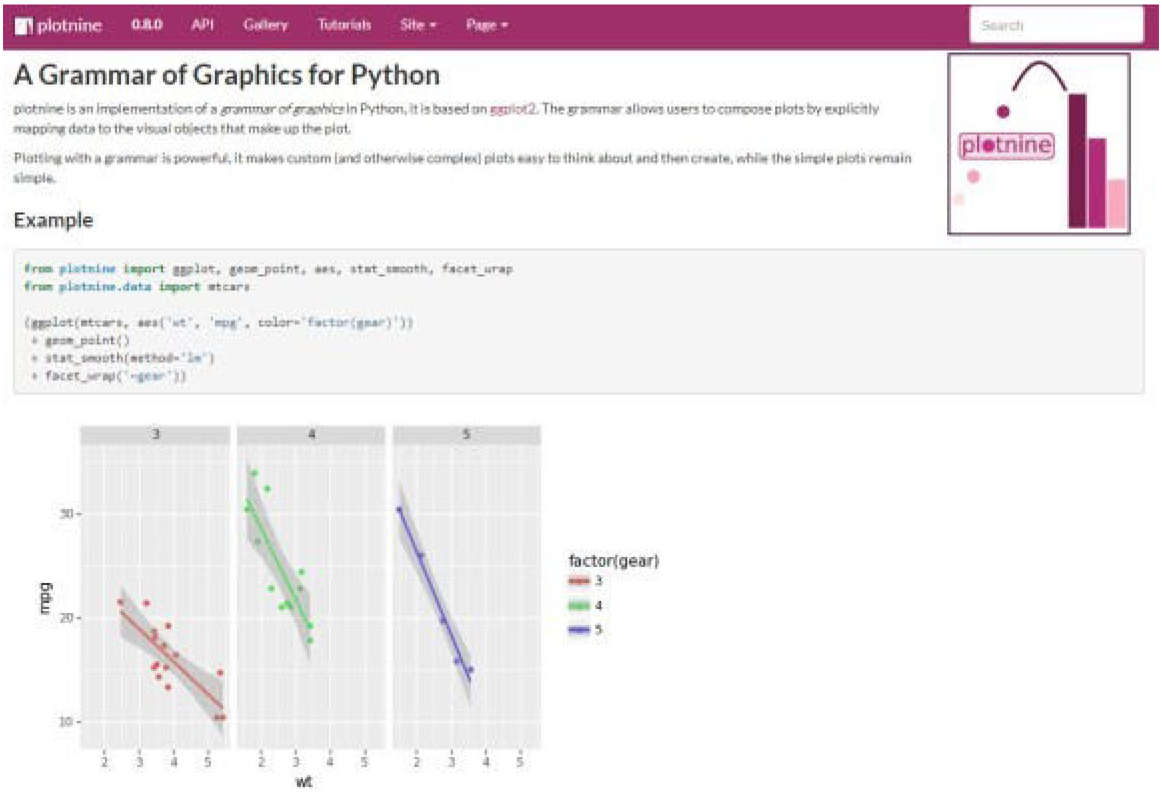
Next, go to the section, “Coming from R?”. You can quickly get to the `Plotnine` Documentation.



(Click image to download)

Explore Plotnine

You have access to the Plotnine Documentation at your fingertips.



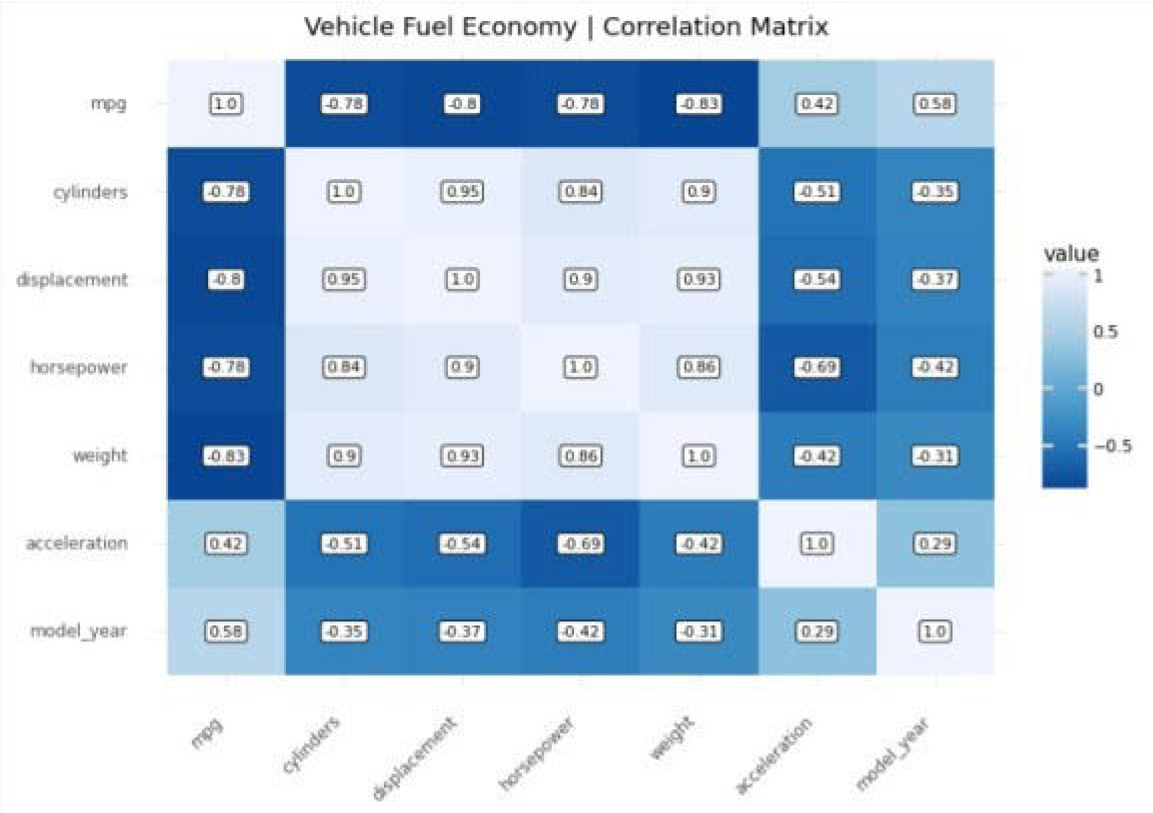
Onto the tutorial.

How Plotnine Works

From the *Plotnine Documentation*, you can see that the grammar of graphics from `ggplot` is used to add layers that control geometries, facets, themes, and more.

Making a Correlation Matrix Plot

Let's check out how to make a professional **correlation matrix plot** with `plotnine`.



Get the code.

Step 1: Load Libraries and Data

First, let's load the libraries and data. From the libraries, we'll import `numpy` and `pandas` to start out. We'll also load the `mpg` dataset.

```
# LIBRARIES ----
import numpy as np
import pandas as pd

# DATASET ----
mpg_df = pd.read_csv("https://raw.githubusercontent.com/mwaskom/seaborn-data/master/mpg.csv")
mpg_df
```

Get the code.

We'll also load the `mpg_df` data set.

```
[26] mpg_df
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	name
0	18.0	8	307.0	130.0	3504	12.0	70	usa	chevrolet chevelle malibu
1	15.0	8	350.0	165.0	3693	11.5	70	usa	buick skylark 320
2	18.0	8	318.0	150.0	3436	11.0	70	usa	plymouth satellite
3	16.0	8	304.0	150.0	3433	12.0	70	usa	amc rebel sst
4	17.0	8	302.0	140.0	3449	10.5	70	usa	ford torino
...	...	...	...	...	...	...	...	...	...
393	27.0	4	140.0	86.0	2790	15.6	82	usa	ford mustang gl
394	44.0	4	97.0	52.0	2130	24.6	82	europe	vw pickup
395	32.0	4	135.0	84.0	2295	11.6	82	usa	dodge rampage
396	28.0	4	120.0	79.0	2625	18.6	82	usa	ford ranger
397	31.0	4	119.0	82.0	2720	19.4	82	usa	chevy s-10

398 rows × 9 columns

Step 2: Expose Relationships with Correlation

> Goal: Understand Relationships to Fuel Economy (`mpg`) versus vehicle attributes like weight, cylinders, and model year.

The correlation matrix is a square (n-by-n) matrix that shows the relationships between each feature. The correlation values range from -1 to +1 indicating both the strength (magnitude) and direction (positive/negative) of the relationship.

Code

We'll use the `corr()` method from `Pandas` to make a **correlation matrix** as a Pandas DataFrame.

```
[27] mpg_df.corr()
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year
mpg	1.000000	-0.775396	-0.804203	-0.778427	-0.831741	0.420289	0.579267
cylinders	-0.775396	1.000000	0.950721	0.842983	0.896017	-0.505419	-0.348746
displacement	-0.804203	0.950721	1.000000	0.897257	0.932824	-0.543684	-0.370164
horsepower	-0.778427	0.842983	0.897257	1.000000	0.864538	-0.689196	-0.416361
weight	-0.831741	0.896017	0.932824	0.864538	1.000000	-0.417457	-0.306564
acceleration	0.420289	-0.505419	-0.543684	-0.689196	-0.417457	1.000000	0.288137
model_year	0.579267	-0.348746	-0.370164	-0.416361	-0.306564	0.288137	1.000000

Step 3: Wrangle the Data into Tidy Format

> Goal: Prepare the data for visualization with `plotnine` by formatting in "long" ("tidy") format

The `plotnine` data visualization API requires data to be in the "tidy" or long format where each row is an observation. In this case, we need each row to contain the first variable, the second variable, and the value of the correlation. We can do this with `pandas`. Pandas can be a challenge for beginners. I teach `pandas` in-depth with 5-hours of data wrangling training in Module 3 of my Python for Data Science Automation Course.

```python
import plotnine as p9
import plydata.cat_tools as cat

tidy_corr = mpg_df \
    .corr() \
    .melt(
        ignore_index=False,
    ) \
    .reset_index() \
    .set_axis(
        labels = ["var1", "var2", "value"],
        axis   = 1
    ) \
    .assign(lab_text = lambda x: np.round(x['value'], 2)) \
    .assign(
        var1 = lambda x: cat.cat_inorder(x['var1']),
        var2 = lambda x:
            cat.cat_rev(
                cat.cat_inorder(x['var2'])
            )
    )
```

Get the code.

The trick here is to use:

- Import `plotnine` and `plydate.cat_tools` to use ggplot functionality next and to more easily work with categorical data
- `melt()` to pivot the data longer
- `assign()` to add label text columns for the heatmap labels
- `assign()` and `cat_inorder()` to organize the categorical columns as categories in the correct order.

This outputs the data in Tidy format.

```
[28] tidy_corr
```

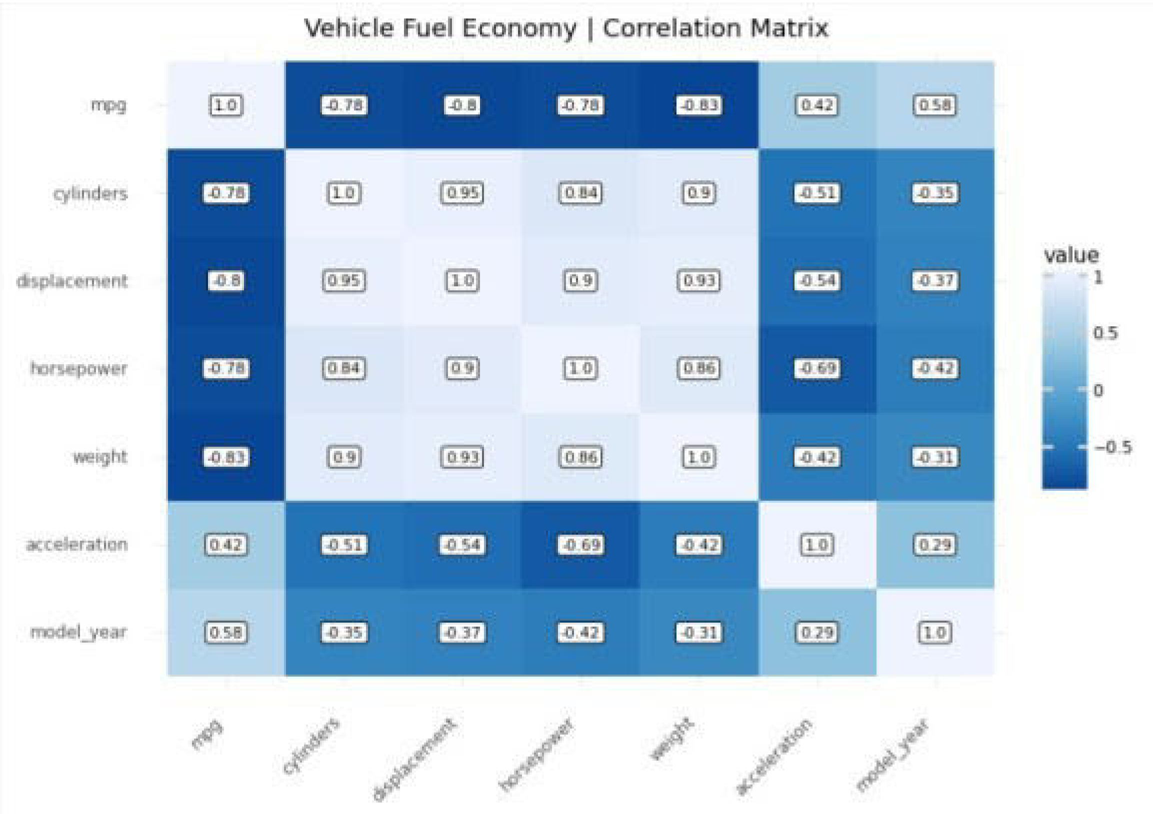
	var1	var2	value	lab_text
0	mpg	mpg	1.000000	1.00
1	cylinders	mpg	-0.775396	-0.78
2	displacement	mpg	-0.804203	-0.80
3	horsepower	mpg	-0.778427	-0.78
4	weight	mpg	-0.831741	-0.83
5	acceleration	mpg	0.420289	0.42
6	model_year	mpg	0.579267	0.58
7	mpg	cylinders	-0.775396	-0.78
8	cylinders	cylinders	1.000000	1.00
9	displacement	cylinders	0.950721	0.95
10	horsepower	cylinders	0.842983	0.84
11	weight	cylinders	0.896017	0.90

Step 4: Make the correlation visualization with `plotnine`

> Goal: Make a professional-looking correlation plot that could be used in a business report to highlight key relationships to management.

Correlation visualizations are very powerful for business reporting as they can highlight key relationships for management. The problem is that many data scientists don't know how to make them look professional, which can detract from your message to business stakeholders. Thankfully, `plotnine` solves this challenge. I teach `plotnine` in-depth with 4-hours of data visualization training in Module 7 of my Python for Data Science Automation Course.

First, here's the correlation matrix heatmap visualization. We can clearly see that as cylinders increase (bigger engine) and weight increases (larger vehicles), fuel economy (mpg) tends to decrease. Conversely, as acceleration increases (possibly due to lower weight) and model year increases (newer vehicles), fuel economy tends to increase.



Next, here's the code used to generate the visual.

```
p9.ggplot(
    mapping = p9.aes("var1", "var2", fill = "value"),
    data    = tidy_corr
) + \
    p9.geom_tile() + \
    p9.geom_label(
        p9.aes(label = "lab_text"),
        fill = "white",
        size = 8
    ) + \
    p9.scale_fill_distiller() + \
    p9.theme_minimal() + \
    p9.labs(
        title = "Vehicle Fuel Economy | Correlation Matrix",
        x = "", y = ""
    ) + \
    p9.theme(
        axis_text_x= p9.element_text(rotation=45, hjust = 1),
        figure_size=(8,6)
    )
```

The trick here is to use:

- `geom_tile()` to make the heat map.
- `geom_label()` to add label text for the correlation values.
- `scale_fill_distiller()` to add a nice fill to the tile to give a professional

appearance.

Summary

This was a short introduction to `plotnine`, which brings `ggplot2` to python. If you're coming from R, `plotnine` is a great package to make professional plots in `Python`.

With that said, you're eventually going to want to learn `pandas`, the most widely used data wrangling tool in Python. Why?

- Our data wrangling code was written in Pandas
- Most data science teams use Pandas
- Pandas plays nicely with Plotnine

So, it makes sense to eventually learn Pandas and Plotnine to help with communication and working on R/Python teams.

If you'd like to learn data science for business with `Python`, `Pandas`, and `Plotnine` from an R-programmers guidance, then read on.…