Introduction to Polars (and Plotly) in Python

J. Hathaway - Data Science Program Chair at BYU-I

Disclaimers

I am focusing on modern tools for data science in Python

Polars over Pandas and Plotly over Matplotlib.

These modern tools reflect the best of

- declarative programming (task focused programming)
- clean grammar (language abstraction that allows intuitive but complex actions)
- industry respect (these tools are very popular for their quality and have rapid growth)

Agenda

Exemplify the data science process - Extract,
Transform, Load, Analyze

1. Introduction and Set-up (10 minutes)
2. Polars for data munging (40 minutes)
3. Break (5 minutes)
4. Plotly for data visualization (35 minutes)

J. Hathaway

Data Scientists with ~20 years of industry experience and 8 years in Academia. Undergraduate degree in Economics (University of Utah) and a graduate degree in Statistics (BYU).

Checking our installation

1. [Python Installed](#)
2. [VS Code Installed](#)
3. [Python VS Code Extension Installed](#)

4. Python packages installed.

```
pip install polars plotly statsmodels pyarrow
```

```
pip3 install polars plotly statsmodels pyarrow
```

Introduction to Polars and Data Munging (speed)

Polars is a lightning fast DataFrame library/in-memory query engine. Its embarrassingly parallel execution, cache efficient algorithms and expressive API makes it perfect for efficient data wrangling, data pipelines, snappy APIs and so much more. Polars is about as fast as it gets, see the results in the [H2O.ai benchmark](#).
[Polars Website](#)

What is Polars?

The goal of `Polars` is to provide a lightning fast `DataFrame` library that:

- Utilizes all available cores on your machine.
- Optimizes queries to reduce unneeded work/memory allocations.
- Handles datasets much larger than your available RAM.
- Has an API that is consistent and predictable.
- Has a strict schema (data-types should be known before running the query).

Polars is written in Rust which gives it C/C++ performance and allows it to fully control performance critical parts in the query engine.

As such `Polars` goes to great lengths to:

- Reduce redundant copies.
- Traverse memory cache efficiently.
- Minimize contention in parallelism.
- Process data in chunks.
- Reuse memory allocations.

[Polars Documentation](#)

Their documenation compares Polars to some other Python data science tools

Introduction to Polars and Data Munging (declarative API)

Polars functions (Contexts & Expressions) are human readable and they align with standard SQL methods as well as the very popular big data package PySpark.

Contexts

- **Selection:** `df.select([..])`, `df.with_columns([..])`
- **Filtering:** `df.filter()`
- **Groupby/Aggregation:** `df.groupby(..).agg([..])`

Expressions

```python
new_df = df.select(
    pl.col("names").n_unique().alias("unique"),
    pl.approx_unique("names").alias("unique_approx"),
    (pl.col("nrs").sum() + 5).alias("nrs + 5"),
    pl.col("integers").cast(pl.Float32).alias("integers_as_floats"),
    pl.col("animal").str.n_chars().alias("letter_count")
)
```

Polars programming

Now let's practice using Polars with our installation of Python

1. Read data, melt data, and save as `.parquet` (01_read.py)
2. Munge data for visualization (02_munge.py)

Introduction to Parquet files and Arrow

When users ask for data or when many institutions share data the files are usually some type of text file (`.txt`, `.csv`, `.tsv`) or an Excel file. The most ubiquitous format is `.csv`. However, these formats limit effective data handling. We want a format that stores small, reads fast, and maintains variable types. The [Apache Arrow project](#) facilitates the goal with the `.parquet` and `.feather` formats and their respective packages for leveraging those formats.

Arrow is primarily an in-memory format, whereas Parquet is a storage format.

The following table from the [openbridge blog](#) provides a strong example of the benefits of these new formats.

Dataset	Size on Amazon S3	Query Run time	Data Scanned	Cost
Data stored as CSV files	1 TB	236 seconds	1.15 TB	$5.75
Data stored in Apache Parquet format*	130 GB	6.78 seconds	2.51 GB	$0.01
Savings / Speedup	87% less with Parquet	34x faster	99% less data scanned	99.7% savings

Introduction to Data Visualization

The world is full of [visual communication tools](#)

Our eyes are drawn to [colors and patterns](#). We can quickly identify red from blue, and squares from circles. Our culture is visual, including everything from art and advertisements to TV and movies. Data visualization is another form of visual art that grabs our interest and keeps our eyes on the message.

[Tableau Reference](#)

Advantages of data visualization:

- Easily sharing information.
- Interactively explore opportunities.
- Visualize patterns and relationships.

Disadvantages:

- Biased or inaccurate information.
- Correlation doesn't always mean causation.
- Core messages can get lost in translation.

Introduction to Plotly and Lets-Plot

Plotly:

Plotly Express is a built-in part of the plotly library, which makes interactive, publication-quality graphs.

```
pip3 install plotly #mac
pip install plotly #windows
```

```
import plotly.express as px
```

Lets-Plot:

Lets-Plot is a multiplatform plotting library based on the Grammar of Graphics. We provide ggplot2-like plotting API for Python

```
pip3 install lets-plot #mac
pip install lets-plot #windows
```

```
from lets_plot import *
LetsPlot.setup_html()
```

Introduction to **Plotly** for Data Visualization

The Plotly Python package leverages the plotly.js JavaScript library to enables Python users to create beautiful interactive web-based visualizations. Plotly.js is built on top of d3.js and stack.gl, Plotly.js is a high-level, declarative charting library. plotly.js ships with over 40 chart types, including 3D charts, statistical graphs, and SVG maps.
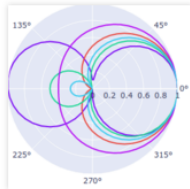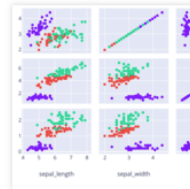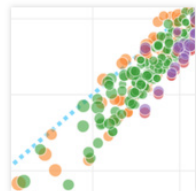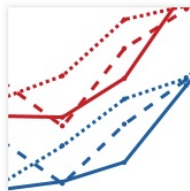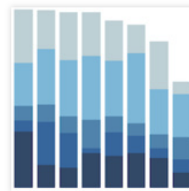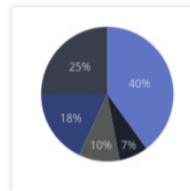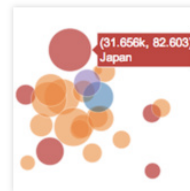
Plotly programming

Now let's practice using [Plotly](#) with our installation of Python

1. Plotly practice (03_begin_plotly.py)
2. Data visualization with our munged data (03_explore.py)

```python
import plotly.express as px
df = px.data.iris() # iris is a pandas DataFrame
fig = px.scatter(df, x="sepal_width", y="sepal_length")
fig.show()
```

```python
import plotly.express as px
df = px.data.gapminder().query("continent == 'Oceania'")
fig = px.line(df, x='year', y='lifeExp', color='country', markers=True)
fig.show()
```

Lets-Plot programming

Now let's practice using [Lets-Plot](#) with our installation of Python

1. Data visualization with our munged data (03_explore.py)

```python
import numpy as np
from lets_plot import *
LetsPlot.setup_html()

np.random.seed(12)
data = dict(
    cond=np.repeat(['A', 'B'], 200),
    rating=np.concatenate((np.random.normal(0, 1, 200), np.random.normal(1, 1.5, 200)))
)

ggplot(data, aes(x='rating', fill='cond')) + \
  ggsize(700, 300) + \
  geom_density(color='dark_green', alpha=.7) + \
  scale_fill_brewer(type='seq') + \
  theme(panel_grid_major_x='blank')
```

Next Steps

1. Program your day (incorporate programming into your daily work)
2. Display your talent -- Github.com
3. Offer your services -- Find small projects to do for friends and contacts
4. Apply for jobs -- LinkedIn