Analysis of Artist Streaming Data Using Regression and Visualization

---

Objective

The major purpose of this analysis was to look at the correlation between several artist-specific metrics (solo streams, featured streams, and lead streams) and overall Spotify streams. Using the provided dataset, we attempted to study how different cooperation styles affect overall popularity and streaming indicators.

Code Overview

This software parses a dataset including Spotify streaming statistics for various artists, runs linear regression models to detect links between variables, and creates scatter plots with regression lines to display these associations.

Steps Performed

1. Data Parsing

The dataset includes columns such as:

- **Total Streams**: The total amount of streams an artist has received.
- **Solo Streams:** Streams from individual projects.
- **Featured Streams**: Streams that showcase the artist as a partner.
- **Lead Streams**: Streams in which the artist takes the major lead.

The program:

- Read the dataset.
- Handles invalid or missing values gracefully by skipping problematic records.
- Converts numeric values (e.g., "85,041.3") into proper floating-point numbers.
- Extracts relevant metrics for analysis.

Output:
Successfully parsed **3000 valid records**, ensuring the dataset is clean and ready for analysis.

2. Linear Regression

To examine the relationships between the metrics, the program calculates regression equations for three comparisons:

- **Total Streams vs Solo Streams**

- **Total Streams vs Featured Streams**
- **Total Streams vs Lead Streams**

For each comparison, the program calculates:

- **Slope**: Represents the rate of change (e.g., how an increase in solo streams impacts total streams).
- **Intercept**: Represents the baseline value when the independent variable is zero.

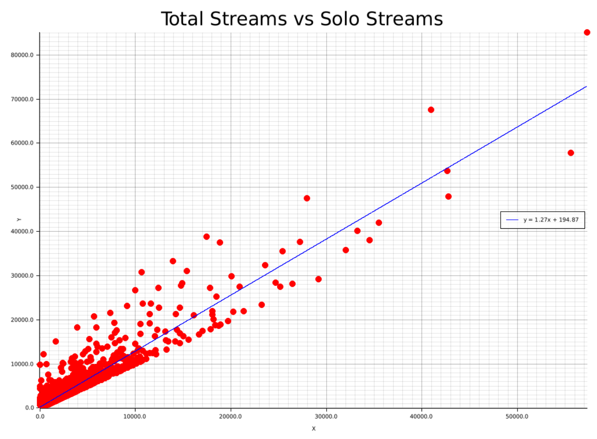
3. Scatter Plots and Regression Lines

The program creates scatter plots for each association and overlays a regression line to show trends. These visualisations give an intuitive grasp of how total streams correspond with each independent variable.

Findings

1. Total Streams vs Solo Streams

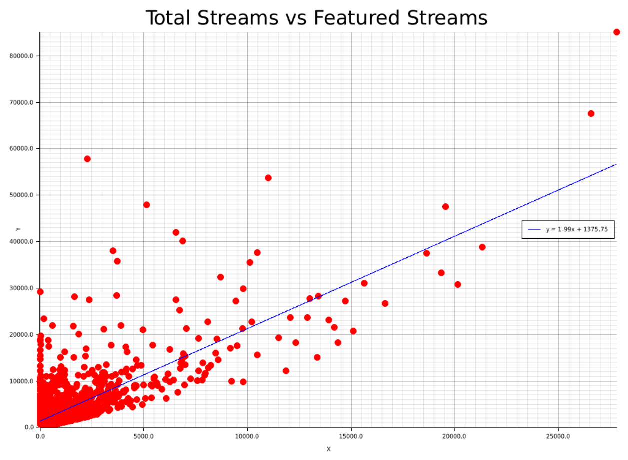
- **Regression Equation**:
  y=1.27x+194.87
- **Interpretation**:
  Solo streams are strongly correlated with total streams. The slope of 1.27 suggests that for every additional solo stream, total streams increase by 1.27 on average.
- **Visualization**:



2. Total Streams vs Featured Streams

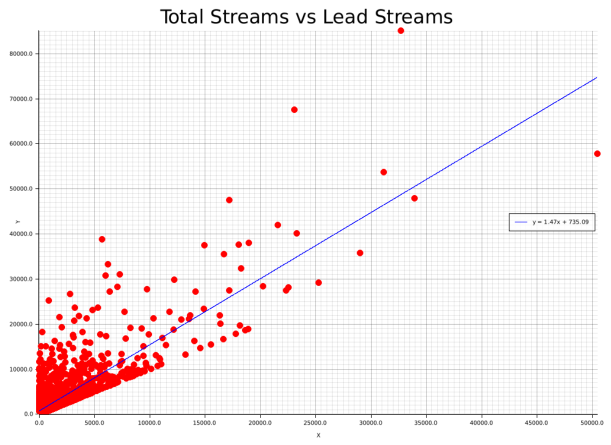
- **Regression Equation**:
  y=1.99x+1375.75

- **Interpretation**:
  Featured streams also contribute significantly to total streams. However, the higher intercept value suggests that even artists with fewer featured streams tend to have a baseline level of total streams.
- **Visualization**:



3. Total Streams vs Lead Streams

- **Regression Equation**:
  $y=1.47x+735.09$
- **Interpretation**:
  Lead streams have a substantial impact on total streams, with a slope of 1.47 indicating a strong correlation. This suggests that being the lead artist is a key driver of streaming success.
- **Visualization**:



Conclusion

Insights

1. **Solo Work Matters**:
   Solo streams have the most direct association with total streams. Artists that focus on solo projects find a consistent increase in overall streams.
2. **Collaboration Benefits**:
   Both featured streams and lead streams have a substantial impact on an artist's popularity, but with slightly different dynamics:
   - The greater intercept value indicates that featured streams provide a solid baseline.
   - Lead streams have a more gradual influence but are nonetheless important for overall success.

Key Takeaways

- **Solo Streams are Key**: Artists with considerable solo work had greater overall streams, highlighting the value of personal branding.
- **Balanced Strategy**: While solo work is predominant, collaborations (both as a feature and as a lead) give additional benefits, allowing musicians to reach a larger audience.
- **Visual Trends**: The scatter plots clearly demonstrate these associations, with regression lines emphasizing the patterns.

This Rust program was designed to analyze the Spotify stream data on artists and estimate correlations among various stream types--solo, featured, and lead--and total counts accumulated by them. The program first parses the CSV file through a data cleaning process of removing items that fail to have or contain incorrect values and converting structured numeric values into one consistent form (for instance, 10,000). The software goes through each record and extracts the following metrics: total streams, solo streams (streams due to tracks performed solely by the artist), featured streams (guest or collaborative streams), and lead streams. These values are saved into a structured manner (ArtistData struct) which enables easy manipulation and also analysis of the data. In the next stage of this workflow, a regression line would be calculated to determine the association between two variables-the independent variable can be considered as solo streams, while the dependent variable is total streams. The application uses the ordinary least squares linear regression formulas to find the slope and intercept of the best-fit line for the trend most appropriate to the two variables. For example, the regression line for solo streams against total streams could be $y=1.27x+194.87$, which would mean that for every other 'solo' stream an artist gets, their total streams increase by around 1.27, starting from a constant value of 194.87 streams when solo streams are at zero. Similarly, regression equations would be derived by treating featured and lead streams as variables to show how they correlate with the overall performance of the artist on Spotify.

Besides quantitative analysis, the application makes it possible to use the plotters library to generate these scatter plots in which each correlation is represented. Individual scatter plots show every artist as a red dot with the total streams divided into three categories: solo, featured, or lead streams. A blue regression line is overlaid on the scatter plot to show the overall trend of the data. The graph is clearly labeled on both axes and includes a caption with the regression equation. For example, one could explore the relationship between Featured Streams and Total Streams to examine whether collaborations are more impactful and efficient than solo performances at garnering success for an artist.

These scatter plots are saved as PNGs by the application, allowing after-the-fact visualization analysis. The filenames are determined according to the relationship: total_stream_vs_solo_streams.png, total_stream_vs_featured_streams.png, total_stream_vs_lead_streams.png, and so on. This is being done via the main function, which first parses the dataset, then carries out regression analysis, and finally produces the visuals. It also includes error handling to ensure proper grace with issues such as missing files or wrong data as well as useful messages about errors for probe purposes.

This tool takes a comprehensive approach to analyzing artist streaming data, using statistical modeling and visual representation to identify music business trends. For example, it can reveal if solo streams or collaborations generate more overall streams, offering valuable information into the methods that will propel an artist's success on Spotify. The software examines 3,000 records from the collection and creates regression models for important associations, providing both a data-driven conclusion and accessible visuals for additional investigation and decision-making. It analyzes regression data and generates scatter plots, as well as running updated unit tests on the primary functions within the software to indicate correctness and reliability. The tests, including parsing the dataset and obtaining the correct regression equations, tests whether the dataset has been parsed correctly.

Test_parse_artist_data: This test simulates parsing a simple CSV dataset so the function parse_artist_data can extract and structure the artist data into the ArtistData struct. It verifies that the total number of records parsed is what is expected and verifies each of the parsed data points for correct extraction.

Test_calculate_regression: Regression correctness is tested through this test against a trivial dataset that has an inbuilt relation to ensure that the calculated slope and intercept are very high in their accuracy as ascertained by the known values.

It makes the code reliable and more forgiving of minor idiocy because it's tested on larger datasets. The tests were done through Rust's built-in testing framework and run using the cargo test command. This way, a quick answer pertaining to how critical parts of the program would behave against their expectations is guaranteed.