SLEI Research Analyst Data Task

June 7, 2023

1 SLEI Research Analyst Data Task

Author: Mingyuan Hua Date: June 2, 2023 Last Updated: June 7, 2023

Here are the data. It is Sean Lahman's Baseball Databank, with information like salary, players who have been admitted into the Hall of Fame, player weight, and of course, baseball performance.

1.1 Q1. Evaluate the Hall of Fame by Country of Origin

You've been asked to evaluate the Hall of Fame status of players as a result of country of origin for SLEI Faculty Advisor, Paul Oyer (not likely, but he does love baseball). You decide to make a plot of players in the hall of fame over time, grouped by each country.

1.1.1 Load the data

Import the required libraries and load the necessary CSV files(HallOfFame.csv, Master.csv) from the Lahman Baseball Database.

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objects as go
import plotly.express as px
from IPython.display import Image

# load the data
hof_data = pd.read_csv('core/HallOfFame.csv')
master_data = pd.read_csv('core/Master.csv')
# master_data.head()
```

1.1.2 Clean and merge the data

Clean the data and merge the necessary columns from both datasets based on common player IDs. For this analysis, we need the player's country of origin, induction year into the Hall of Fame, and player ID.

We need to filter for players who are inducted into the Hall of Fame, group the data by country and year, and count the number of players, pivot the data to have each country as a column and fill missing values with zeros, and then calculate cumulative sum for each country.

[2]:
```
# merge the data
merged_data = pd.merge(hof_data[['playerID', 'yearid', 'inducted']],␣
 ↪master_data[['playerID', 'birthCountry']], on='playerID', how='left')
# filter for players who are inducted into the Hall of Fame
hof_players = merged_data[merged_data['inducted'] == 'Y']
# group the data by country and year and count the number of players
grouped_data = hof_players.groupby(['birthCountry', 'yearid']).size().
 ↪reset_index(name='count')
# pivot the data to have each country as a column and fill missing values with␣
 ↪zeros
pivot_data = grouped_data.pivot_table(index='yearid', columns='birthCountry',␣
 ↪values='count', fill_value=0).reset_index()
# calculate cumulative sum for each country
cumulative_data = pivot_data.cumsum()
cumulative_data['yearid'] = pivot_data['yearid']
countries = grouped_data['birthCountry'].unique()
# preview the cleaned data
cumulative_data
```

[2]:

birthCountry	yearid	CAN	Cuba	D.R.	Germany	Netherlands	P.R.	Panama	\
0	1936	0	0	0	0	0	0	0	
1	1937	0	0	0	0	0	0	0	
2	1938	0	0	0	0	0	0	0	
3	1939	0	0	0	0	0	0	0	
4	1942	0	0	0	0	0	0	0	
..	…	…	…	…	…	…	…	…	
70	2012	1	4	1	1	1	3	1	
71	2013	1	4	1	1	1	3	1	
72	2014	1	4	1	1	1	3	1	
73	2015	1	4	2	1	1	3	1	
74	2016	1	4	2	1	1	3	1	

birthCountry	USA	United Kingdom	Venezuela
0	5	0	0
1	13	0	0
2	15	1	0
3	25	1	0
4	26	1	0
..	…	…	…
70	281	3	1
71	284	3	1
72	290	3	1
73	293	3	1
74	295	3	1

[75 rows x 11 columns]

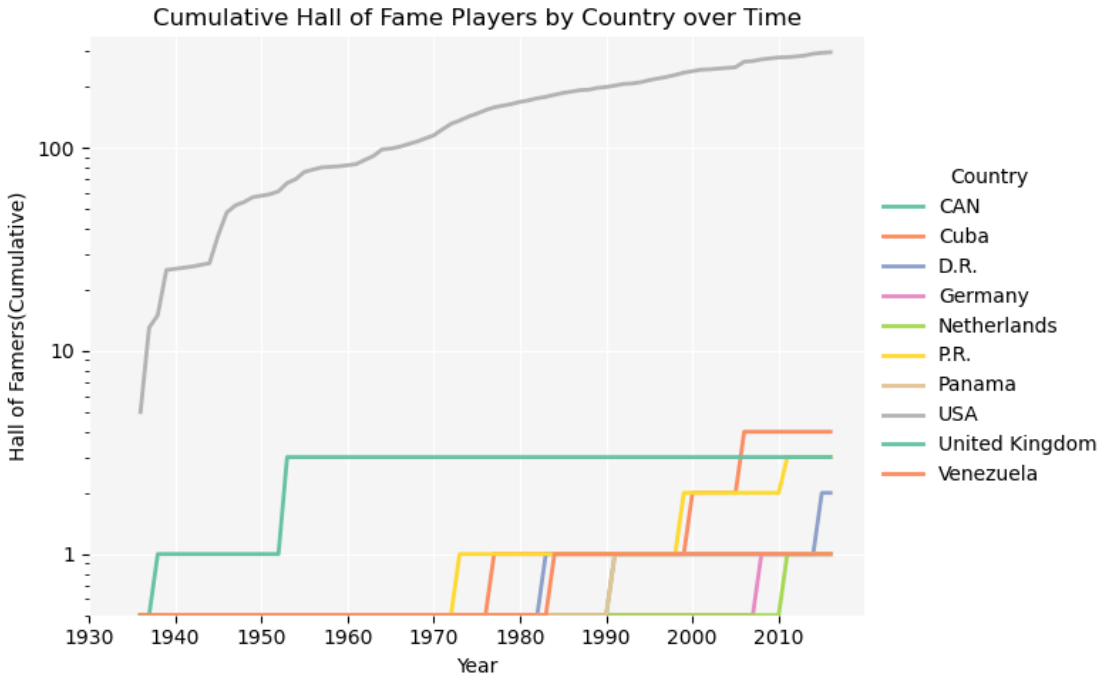
1.1.3 Make the Plots

Make a plot of players in the hall of fame over time, grouped by each country.

Logarithmic Scale We find the number of USA Hall of Famers dominates across all countries, so it is clear to make the plots in logarithmic scale. However, log0 makes no sense, so we replace 0 values with 0.5 to prevent this issues in log scale.

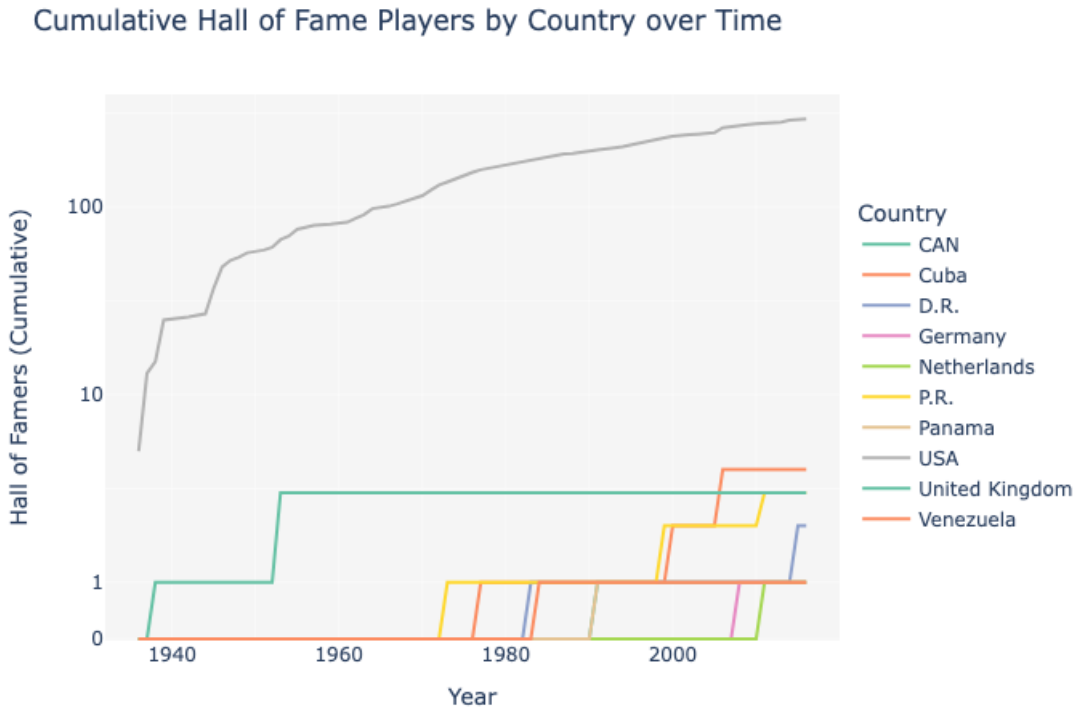
Create the plot using Matplotlib Adjust the x, y-axis limits and tick-marks for better visualization

[3]:
```
offset_data = cumulative_data.replace(0, 0.5)

plt.figure(figsize=(8, 5))
colors = sns.color_palette('Set2', n_colors=len(offset_data.columns[1:]))
for i, country in enumerate(offset_data.columns[1:]):
    plt.plot(offset_data['yearid'], offset_data[country], color=colors[i],␣
 ↪label=country, linewidth=2)

plt.title('Cumulative Hall of Fame Players by Country over Time')
plt.xlabel('Year')
plt.ylabel('Hall of Famers(Cumulative)')
plt.legend(title='Country', loc='center left', bbox_to_anchor=(1, 0.5),␣
 ↪frameon=False)
plt.yscale('log')
plt.yticks([0, 1, 10, 100], [0, 1, 10, 100])
plt.ylim([0.5, 350])
plt.xticks(range(1930, 2020, 10))
plt.xlim(1930, 2020)
plt.grid(True, axis='y', linestyle='-', linewidth=1, color='white')
plt.grid(True, axis='x', linestyle='-', linewidth=1, color='white')
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)
plt.gca().spines['left'].set_visible(False)
plt.gca().spines['bottom'].set_visible(False)
plt.gca().set_facecolor('#F5F5F5')
plt.tight_layout()
plt.show()
```



Create the plot using Plotly for better visualization Use Plotly to get an even nicer graph.

[4]:
```
colors = px.colors.qualitative.Set2
fig = px.line(offset_data, x='yearid', y=offset_data.columns[1:],␣
 ↪color_discrete_sequence=colors)
fig.update_yaxes(type="log", range=[-0.31, 2.6], tickvals=[0.5, 1, 10**0.5, 10,␣
 ↪10**1.5, 100, 10**2.5], ticktext=['0', '1', '', '10', '', '100', ''])
fig.update_xaxes(range=[1932, 2020], tickmode='array',␣
 ↪tickvals=list(range(1940, 2020, 10)), ticktext=[str(year) if year % 20 == 0␣
 ↪and year != 2020 else '' for year in range(1940, 2021, 10)], showgrid=True,␣
 ↪gridwidth=0.5, gridcolor='white')
fig.update_layout(title='Cumulative Hall of Fame Players by Country over Time',␣
 ↪xaxis_title='Year', yaxis_title='Hall of Famers (Cumulative)')
fig.update_layout(plot_bgcolor='#F5F5F5')
fig.update_layout(xaxis=dict(showgrid=True, gridwidth=0.5, gridcolor='white'),␣
 ↪yaxis=dict(showgrid=True, gridwidth=0.5, gridcolor='white'))
fig.update_layout(legend=dict(title='Country', orientation='v',␣
 ↪yanchor='middle', y=0.5, xanchor='left', x=1.02), showlegend=True,␣
 ↪legend_title_text='Country', legend_title_font=dict(size=14),␣
 ↪legend_font=dict(size=12), hovermode='x', margin=dict(t=80, r=80, b=80,␣
 ↪l=80),)
fig.update_layout(height=500, width=650)
fig.show()
```



1.2 Q2. Observations and Questions

After making this graph and giving it back to Paul, what questions and/or observations would you bring up? Are there any interesting research questions or findings you would explore further?

1.2.1 Observations:

1. The United States dominates the Hall of Fame, which is not surprising given that baseball is a major sport in the USA.
2. The first non-US player in the Hall of Fame was from United Kingdom, but United Kingdom didn't have any more inductees after 1953.
3. Other countries besides the USA and United Kingdom didn't start to have players entering the Hall of Fame until 1973, which could indicate a period of internationalization of the sport.
4. The maximum number of non-US players inducted into the Hall of Fame in a year is two. This could suggest that the number of internationally recognized talent in baseball is limited, or that there may be biases in the selection process.

1.2.2 Questions:

1. Why did the induction of non-US players into the Hall of Fame start in 1938 and then not again until 1973? (Henry Chadwick, 1938, sports journalist)
2. What changes or events in global baseball might have contributed to the induction of non-US players starting from 1973?

3. Why did United Kingdom stop having players entering the Hall of Fame after 1953? (minor sports in UK)
4. Could there be any specific reasons why Cuba has the most non-US players in the Hall of Fame as of 2016? (popular, international influence)
5. What are the criteria for induction into the Hall of Fame, and do these criteria favor US players in any way? (statistical achievement and contributions to the sport)

1.2.3 Potential Research Questions:

1. Market Expansion and Globalization: How has the internationalization of baseball impacted the sport's market and economic value? What strategies were used to expand the sport outside of the US and Britain, and how successful were they?
2. Talent Management and Development: How does the talent management and development process differ across countries? What factors contribute to the dominant position of US players in the Hall of Fame?
3. Economic Impact of Hall of Fame Induction: How does the induction into the Hall of Fame impact a player's market value, endorsement opportunities, and overall career earnings? Does this differ between US and non-US players?
4. Organizational Behavior and Bias: Is there evidence of bias in the Hall of Fame selection process? If so, how does it impact international players and what can be done to ensure a more equitable process?
5. Business Strategy of Baseball Teams: How do baseball teams strategize their player acquisitions and training considering the nationality of the players? Is there a correlation between the international diversity of a team and its success or profitability?

1.3 Q3. Centralize Data

The data in the "Baseball Databank" currently exist as multiple csv files. Create a plan to centralize these data. What data management tools would you leverage (or build) in order to house and easily manage all of these growing data?

1.3.1 1. Data Cleaning and Preparation

Start by inspecting the CSV files to understand the data's structure, quality, and any inconsistencies that might exist. Use Python libraries like pandas for this step, which can handle CSV files and provide tools for cleaning and wrangling data.

1.3.2 2. Database Design

Next, design a relational database schema that can accommodate the data in the CSV files. Consider using a database management system (DBMS) like MySQL or PostgreSQL. These systems are robust, open-source, and support SQL, a language that's useful for querying and manipulating data.

1.3.3 3. Data Import

Once the database schema is ready, import the cleaned data from the CSV files into the database. Use SQL commands or Python libraries like SQLAlchemy (for MySQL) or psycopg2 (for PostgreSQL) to automate this process.

1.3.4 4. Data Management and Querying Tools

To easily manage and query the data, consider using a tool like MySQL Workbench for MySQL. These tools provide a graphical interface to the database, making it easier to handle data management tasks.

1.3.5 5. Data Version Control

While MySQL itself does not have built-in data versioning features, we can use a version control system like Git to track changes to your SQL schema files and stored procedures.

For tracking changes to the data in your tables, we can create "audit tables" that log changes to your data. Each time a record in a table is inserted, updated, or deleted, create a new record in the corresponding audit table that records the change.

1.3.6 6. ETL Tools

If the data in the CSV files is updated regularly, consider setting up an ETL (Extract, Transform, Load) process to automate the data update process. Tools like Apache Airflow can help with this.

1.3.7 7. Data Warehouse

If the data is large and needs to be accessed by multiple users for analysis, consider setting up a data warehouse. Tools like Amazon Redshift or Google BigQuery can handle large amounts of data and provide fast query performance.

1.3.8 8. Data Visualization Tools

To help users understand the data, consider setting up data visualization tools like Tableau, Power BI, or open-source alternatives like Apache Superset.

1.3.9 Here is an example using Python, MySQL, and MySQL Workbench to centralize these data.

[5]:
```
import os
import pandas as pd
import mysql.connector
from sqlalchemy import create_engine, text

directory = 'core'

# get a list of all files in the directory
files = os.listdir(directory)

# filter the list for only .csv files
csv_files = [f for f in files if f.endswith('.csv')]

# Create a SQLAlchemy engine
engine = create_engine('mysql+mysqlconnector://root:51585158@localhost:3306/',␣
 ↪echo=False)
```

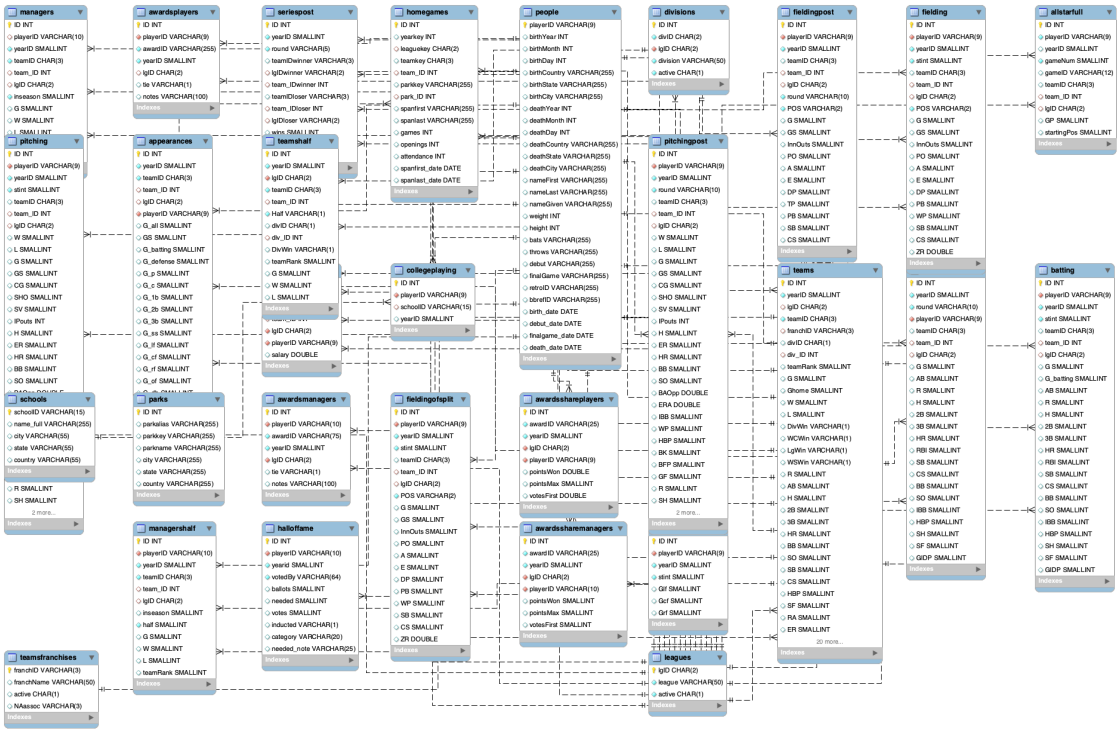
```
# Open and read the SQL file
with open('mysql.sql', 'r') as file:
    sql_script = file.read()

# separate the script into individual statements
statements = sql_script.split(';')

# execute each statement
for statement in statements:
    # remove leading/trailing white space
    statement = statement.strip()

    # skip empty statements (like trailing newlines)
    if statement:
        engine.execute(text(statement))

engine = create_engine('mysql+mysqlconnector://root:51585158@localhost:3306/
 ↪baseballdb', echo=False)

engine.execute(text('SET FOREIGN_KEY_CHECKS=0;'))

for csv_file in csv_files:
    # Load the CSV data into a pandas DataFrame
    df = pd.read_csv(f'{directory}/{csv_file}')

    # rename the column in the DataFrame
    df = df.rename(columns={'rank': 'teamRank', 'Rank': 'teamRank'})
    df.rename(columns=lambda x: x.replace('.', ''), inplace=True)

    # drop duplicates
    if csv_file == 'AwardsManagers.csv':
        df = df.drop_duplicates(subset=['playerID', 'awardID', 'yearID',␣
 ↪'lgID'])

    if csv_file == 'PitchingPost.csv':
        # handle inf values
        df.replace([np.inf, -np.inf], np.nan, inplace=True)
        # handle float values
        df['BAOpp'] = pd.to_numeric(df['BAOpp'], errors='coerce')
        # add columns for foreign keys
        df['team_ID'] = pd.to_numeric(df['teamID'], errors='coerce',␣
 ↪downcast='integer')

    # Write the data from the DataFrame to the MySQL table
    if csv_file != 'Master.csv':
```

```
        df.to_sql(name=csv_file.lower().replace('.csv', ''), con=engine,␣
↪if_exists='append', index=False)
    else:
        df.to_sql(name='people', con=engine, if_exists='append', index=False)

engine.execute(text('SET FOREIGN_KEY_CHECKS=1;'))
```

Here is the Enhanced Entity-Relationship diagram for our data generated by MySQL Workbench.

[6]:
```
# Specify the file path of the PNG image
image_path = 'EER.png'
# Display the image
Image(filename=image_path)
```

[6]:



1.4 Reference

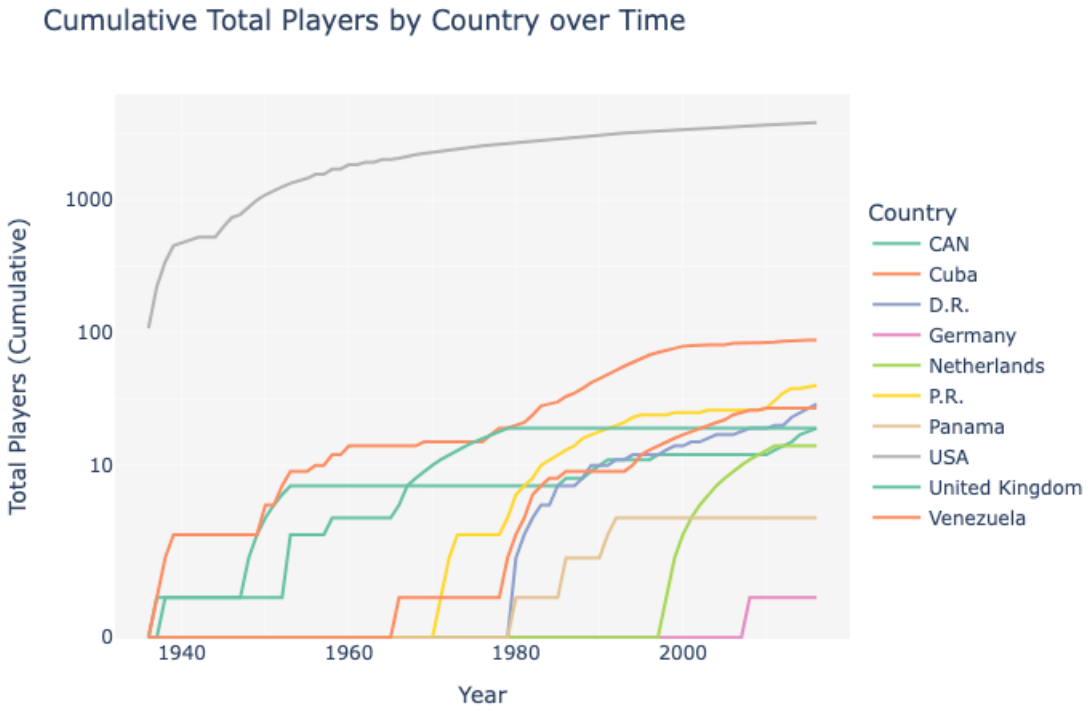
1. Society for American Baseball Research. “Henry Chadwick.” Society for American Baseball Research, https://sabr.org/bioproj/person/henry-chadwick/. Accessed 3 June 2023.

2. Lahman, Sean. “Sean Lahman’s Baseball Database.” SeanLahman.com, http://seanlahman.com/download-baseball-database/. Accessed 3 June 2023.

1.5 Appendix

I am playing around to see if cumulative sum of Hall of Fame players is related to cumulative total number of players by each country over time?

[7]:
```
# Plot Cumulative Total Players by Country over Time

# group the data by country and year and count the number of players
all_grouped_data = merged_data.groupby(['birthCountry', 'yearid']).size().
 ↪reset_index(name='count')
# filter out the countries who have HOF
all_grouped_data = all_grouped_data[all_grouped_data['birthCountry'].
 ↪isin(countries)]
# pivot the data to have each country as a column and fill missing values with␣
 ↪zeros
all_pivot_data = all_grouped_data.pivot_table(index='yearid',␣
 ↪columns='birthCountry', values='count', fill_value=0).reset_index()
# calculate cumulative sum for each country
all_cumulative_data = all_pivot_data.cumsum()
all_cumulative_data['yearid'] = all_pivot_data['yearid']
all_offset_data = all_cumulative_data.replace(0, 0.5)

colors = px.colors.qualitative.Set2
fig = px.line(all_offset_data, x='yearid', y=all_offset_data.columns[1:],␣
 ↪color_discrete_sequence=colors)
fig.update_yaxes(type="log", range=[-0.31, 3.8], tickvals=[0.5, 10, 10**1.5,␣
 ↪100, 10**2.5, 1000, 10**3.5], ticktext=['0', '10', '', '100', '', '1000',␣
 ↪''])
fig.update_xaxes(range=[1932, 2020], tickmode='array',␣
 ↪tickvals=list(range(1940, 2020, 10)), ticktext=[str(year) if year % 20 == 0␣
 ↪and year != 2020 else '' for year in range(1940, 2021, 10)], showgrid=True,␣
 ↪gridwidth=0.5, gridcolor='white')
fig.update_layout(title='Cumulative Total Players by Country over Time',␣
 ↪xaxis_title='Year', yaxis_title='Total Players (Cumulative)')
fig.update_layout(plot_bgcolor='#F5F5F5')
fig.update_layout(xaxis=dict(showgrid=True, gridwidth=0.5, gridcolor='white'),␣
 ↪yaxis=dict(showgrid=True, gridwidth=0.5, gridcolor='white'))
fig.update_layout(legend=dict(title='Country', orientation='v',␣
 ↪yanchor='middle', y=0.5, xanchor='left', x=1.02), showlegend=True,␣
 ↪legend_title_text='Country', legend_title_font=dict(size=14),␣
 ↪legend_font=dict(size=12), hovermode='x', margin=dict(t=80, r=80, b=80,␣
 ↪l=80),)
fig.update_layout(height=500, width=650)
fig.show()
```



[8]:
```
# Calculate the cumulative sum of Hall of Fame players by country over time
cum_grouped_data = grouped_data
cum_grouped_data['Cumulative_HOF_Players'] = grouped_data.
 ↪groupby('birthCountry')['count'].cumsum()

# Calculate the cumulative sum of total players by country over time
cum_all_grouped_data = all_grouped_data
cum_all_grouped_data['Cumulative_Total_Players'] = cum_all_grouped_data.
 ↪groupby('birthCountry')['count'].cumsum()

# Merge the two DataFrames based on country and time
cum_merged_data = pd.merge(cum_grouped_data, cum_all_grouped_data,␣
 ↪on=['birthCountry', 'yearid'])

# Calculate the correlation coefficient between cumulative Hall of Fame players␣
 ↪and cumulative total players
correlation_coefficient = cum_merged_data['Cumulative_HOF_Players'].
 ↪corr(cum_merged_data['Cumulative_Total_Players'])

# Print the correlation coefficient
print("Correlation Coefficient:", correlation_coefficient)
```

```
Correlation Coefficient: 0.9798267050489666
```

[ ]: