Analyzing Trends in the MLB, NBA, and NFL

Aditya Saripalli[1], James Souk[2], Joshua Petzer[3], Pranav Reddy[4]

Purdue University

CS 17600: Data Engineering In Python

Xiaojin Liu

December 7, 2024

Analyzing Trends in the MLB, NBA, and NFL

Our research team has analyzed trends across the three most popular professional sports leagues in America—Major League Baseball (MLB), the National Basketball Association (NBA), and the National Football League (NFL). In the MLB, we compared rookie performances against veteran performances and sought to model how, on average, a player's workload changes as they age. In the NBA, we compared the impact of international players against American players and dove into another rookie versus veteran performance analysis. In the NFL, we looked closely at quarterback performances: first, their main trends across the NFL's two conferences, and second, a comparison between Tom Brady and Peyton Manning—two of the most famous players to hold the position. We learned a lot about our favorite sports with this project and enjoyed putting our skills to the test. Enjoy our findings!

Major League Baseball

Rookies vs Veterans

Note: All code snippets in this section are from *rookies_vs_vets.py* by Souk and Petzer (2024), available at https://github.com/chrissouk/CS-176-Final-Project.

Data Collection

To compare rookie baseball players against veteran baseball players, you need data. Luckily, we found an excellent dataset on Kaggle, titled *Every Player In History* (Anhorn, 2021), containing stats for every hitter and pitcher in MLB history. Each row of data represented each season a player played, so for this comparison, we managed to compare veterans against their younger, rookie selves for a fuller analysis.

The second thing you need to compare rookie baseball players against veteran baseball players is a good definition of what exactly rookies and veterans are. To answer this, we found

two fantastic sources that gave us crystal clear definitions: a rookie is an MLB player who has had not more than 130 at-bats or 50 innings pitched (Major League Baseball, 2023), and a veteran is an MLB player who has at least 5 seasons under their belt (Baseball Almanac, 2014).

Now we knew what we were doing, we were ready to import the datasets:

Code Snippet 1

```
import kagglehub
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt

# debugging
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)
pd.set_option('display.max_colwidth', None)


### get datasets via kagglehub
path = kagglehub.dataset_download('finkleiseinhorn/mlb-every-player-in-mlb-history')
files = os.listdir(path)

pitchers_csv = [f for f in files if f.endswith('csv')][0]
pitchers_path = os.path.join(path, pitchers_csv)
pitchers_df = pd.read_csv(pitchers_path)

hitters_csv = [f for f in files if f.endswith('csv')][1]
hitters_path = os.path.join(path, hitters_csv)
hitters_df = pd.read_csv(hitters_path)
```

Filtering

Next, we filtered out bad data. In this case, that included NaN values and zero values for a player's number of games, at-bats, and innings pitched, as well as infinite ERA values.

Code Snippet 2

```
### filter datasets

# filter nan games played
pitchers_df = pitchers_df.dropna(subset=['G'])
hitters_df = hitters_df.dropna(subset=['G'])
# filter no games played
pitchers_df = pitchers_df[pitchers_df['G'] != 0]
hitters_df = hitters_df[hitters_df['G'] != 0]
# filter nan at bats and nan innings pitched
pitchers_df = pitchers_df.dropna(subset=['IP'])
hitters_df = hitters_df.dropna(subset=['PA'])
# filter no at bats and no innings pitched
pitchers_df = pitchers_df[pitchers_df['IP'] != 0]
hitters_df = hitters_df[hitters_df['PA'] != 0]
# remove infinite ERA values
pitchers_df.replace(np.inf, np.nan, inplace=True)
pitchers_df.dropna(subset=['ERA'], inplace=True)
```

Sorting

Now that we had only good data, we needed to sort it into rookie and veteran subsets according to our sources to make our comparisons. To do this, we augmented the data frames by adding columns to the end that calculated career total stats for each of the relevant statistics: games played, at-bats, and innings pitched. Using these new columns, we were able to easily sort out who the rookies were and who the veterans were.

Code Snippet 3

```
### sorting

## create extra columns to calculate relevant experience criteria
pitchers_df['Years Played'] = pitchers_df.groupby('name').cumcount() + 1
pitchers_df['Total IP'] = pitchers_df.groupby('name')['IP'].transform('sum')

hitters_df['Years Played'] = hitters_df.groupby('name').cumcount() + 1
hitters_df['Total AB'] = hitters_df.groupby('name')['AB'].transform('sum')

## rookie criteria via mlb.com/glossary/rules/rookie-eligibility
# no more than 50 innings pitched for pitchers
rookie_pitchers_df = pitchers_df[(pitchers_df['Total IP'] <= 50)].copy()
# no more than 130 at bats for hitters
rookie_hitters_df = hitters_df[(hitters_df['Total AB'] <= 130)].copy()

## vet criteria via baseball-almanac.com
# at least 5 seasons played
vet_pitchers_df = pitchers_df[pitchers_df['Years Played'] >= 5].copy()
vet_hitters_df = hitters_df[hitters_df['Years Played'] >= 5].copy()
```

Stat Selection & Normalization

To continue preparing our data for our comparison, we needed to specify exactly which columns of data we wanted to use to compare the different sets of players. For hitters, we chose basic stats that represented a player's performance well. But, many of these stats were running totals for the season, so we decided to normalize them by the PA (plate appearance) column, to account for rookies not getting as much playing time. For pitchers, the statistics we chose were already normalized for playing time, so we did not implement redundant normalization.

Code Snippet 4

```
# filter comparison stats to only those that create a fair comparison, accounting for differences in playing time
hitting_comparison_columns = [
    'H',   # Hits
    '2B',  # Doubles
    '3B',  # Triples
    'HR',  # Home Runs
    'RBI', # Runs Batted In
    'BB',  # Walks
    'SO',  # Strikeouts
    'OBP', # On-Base Percentage
    'SLG', # Slugging Percentage
    'OPS'  # On-Base Plus Slugging
]
pitching_comparison_columns = [
    'ERA',  # Earned Run Average
    'WHIP', # Walks + Hits per Inning Pitched
    'H9',   # Hits per 9 innings
    'HR9',  # Home Runs per 9 innings
    'BB9',  # Walks per 9 innings
    'SO9',  # Strikeouts per 9 innings
    'SO/W', # Strikeout-to-Walk Ratio
]

# normalize hitting stats per plate appearance to account for differences in playing time
def normalize_batter_stats(df):
    for col in hitting_comparison_columns:
        if col not in ['OBP', 'SLG', 'OPS']:
            df[col + '_per_PA'] = df[col] / df['PA']
    return df

rookie_hitters_df = normalize_batter_stats(rookie_hitters_df)
vet_hitters_df = normalize_batter_stats(vet_hitters_df)
```

Compare & Visualize

Now that our data was ready to go, we created boxplots to compare average performance between the two groups.

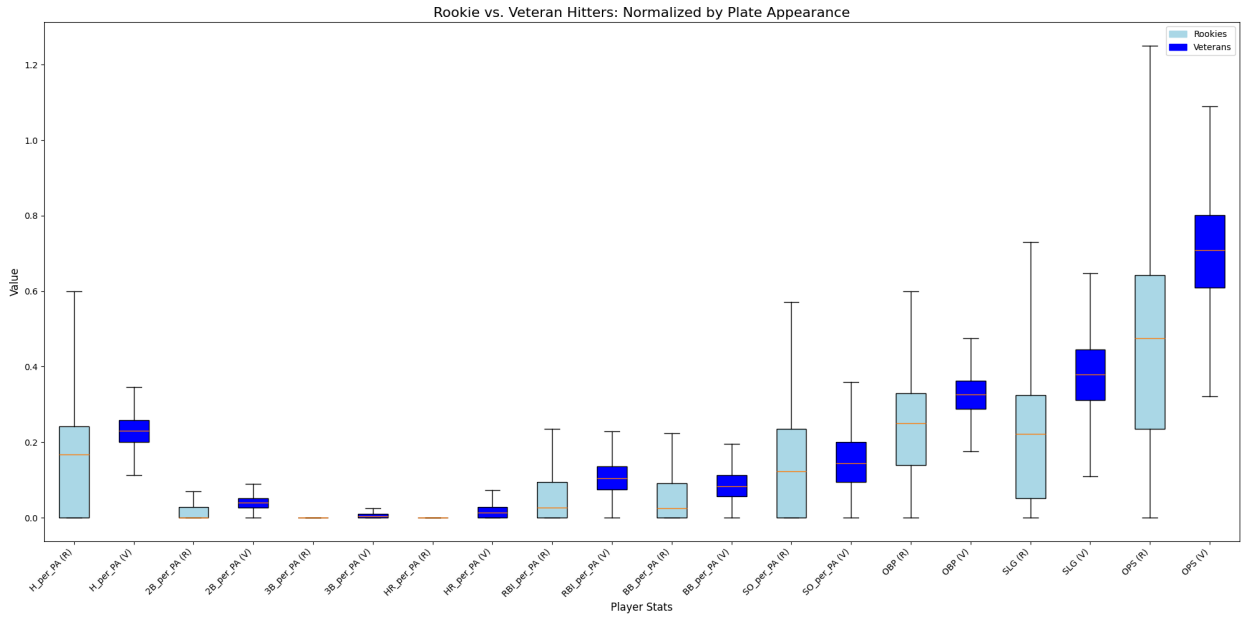
Code Snippet 5

```
      def save_boxplot_comparison(df_rookies, df_vets, columns, title, filename,
colors):
          fig, ax = plt.subplots(figsize=(20, 10))

          plot_data = []
          labels = []
          for col in columns:
              rookie_col = pd.to_numeric(df_rookies[col], errors='coerce').dropna()
              vet_col = pd.to_numeric(df_vets[col], errors='coerce').dropna()

              plot_data.append(rookie_col)
              plot_data.append(vet_col)

              labels.append(f'{col} (R)')
              labels.append(f'{col} (V)')

          bp = ax.boxplot(plot_data,
                          patch_artist=True,
                          labels=labels,
                          showfliers=False)

          colors = colors * len(columns)
          for patch, color in zip(bp['boxes'], colors):
              patch.set_facecolor(color)

          legend_elements = [
              plt.Rectangle((0,0), 1, 1, color=colors[0], label='Rookies'),
              plt.Rectangle((0,0), 1, 1, color=colors[1], label='Veterans')
          ]
          ax.legend(handles=legend_elements, loc='upper right')

          ax.set_title(title, fontsize=16)
          ax.set_xlabel('Player Stats', fontsize=12)
          ax.set_ylabel('Value', fontsize=12)
```

```
        plt.setp(ax.get_xticklabels(), rotation=45, ha='right')

        plt.tight_layout()
        plt.savefig(filename)
        plt.close()


    save_boxplot_comparison(
        rookie_hitters_df,
        vet_hitters_df,
        [col + '_per_PA' if col not in ['OBP', 'SLG', 'OPS'] else col for col in
hitting_comparison_columns],
        'Rookie vs. Veteran Hitters: Normalized by Plate Appearance',
        'rookie_vs_veteran_hitters.png',
        ['lightblue', 'blue']
    )

    save_boxplot_comparison(
        rookie_pitchers_df,
        vet_pitchers_df,
        pitching_comparison_columns,
        'Rookie vs. Veteran Pitchers',
        'rookie_vs_veteran_pitchers.png',
        ['pink', 'red']
    )
```

Visualization 1

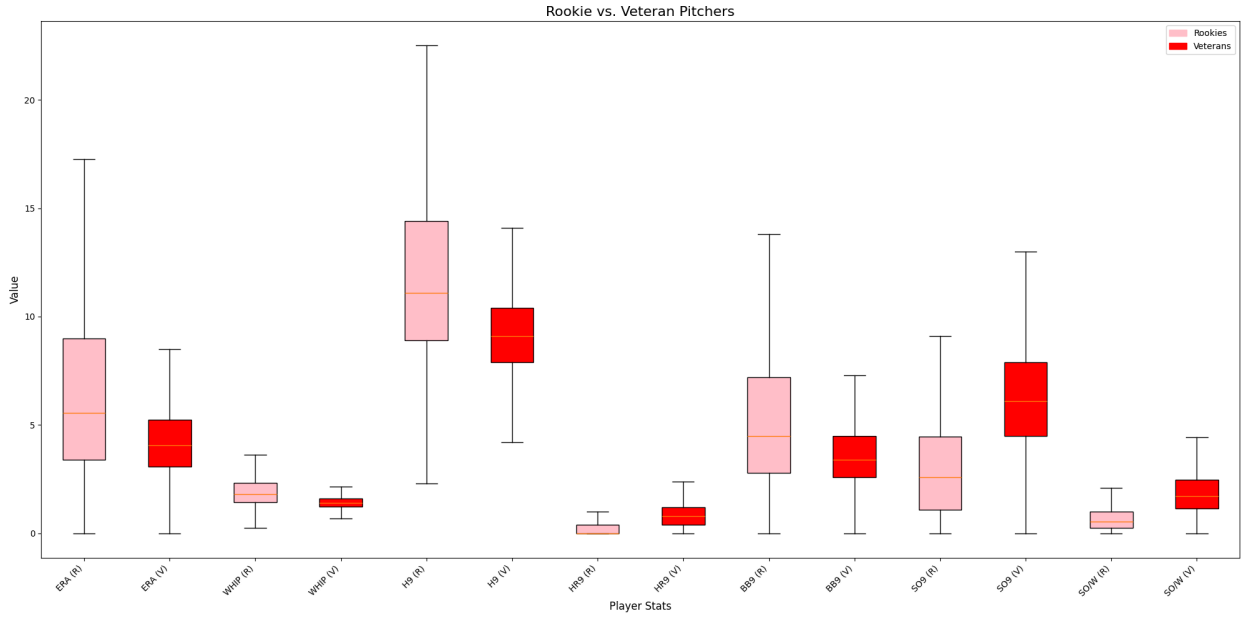
Rookie vs. Veteran Hitters: Normalized by Plate Appearance



Note: Higher averages across all statistics correspond to better performances.

Visualization 2

Rookie vs. Veteran Pitchers



Note: A lower average for all pitching stats except HR9, SO9, and SO/W indicates better performances, on average, and vice versa for the excepted stats.

Findings

What we found was unsurprising and remarkably consistent across every stat: veterans are more consistent and better on average than their rookie counterparts. You can see this plainly in the graphs: darker veteran distributions are, on average, tighter-grouped, and consistently better across every single stat, except only one—pitcher's HR9 (home runs allowed per 9 innings).

Average Workload per Player Age

Note: All code snippets in this section are from *workload_per_age.py* by Souk and Petzer (2024), available at https://github.com/chrissouk/CS-176-Final-Project.

Gathering Data

To see how average workload evolves as an MLB player ages, we used the same hitting and pitching datasets as before since they contained both of our stats in question: games played (also referred to as workload), and player age.

Code Snippet 6

```
import kagglehub
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt


### get datasets via kagglehub
path = kagglehub.dataset_download('finkleiseinhorn/mlb-every-player-in-mlb-history')
files = os.listdir(path)
```

```
pitchers_csv = [f for f in files if f.endswith('csv')][0]
pitchers_path = os.path.join(path, pitchers_csv)
pitchers_df = pd.read_csv(pitchers_path)

hitters_csv = [f for f in files if f.endswith('csv')][1]
hitters_path = os.path.join(path, hitters_csv)
hitters_df = pd.read_csv(hitters_path)
```

Filtering

Since we're concerned with the same data as the last dataset, we filtered it the same way.

Code Snippet 7

```
### filter datasets

# filter nan games played
pitchers_df = pitchers_df.dropna(subset=['G'])
hitters_df = hitters_df.dropna(subset=['G'])
# filter no games played
pitchers_df = pitchers_df[pitchers_df['G'] != 0]
hitters_df = hitters_df[hitters_df['G'] != 0]
# filter nan at bats and nan innings pitched
pitchers_df = pitchers_df.dropna(subset=['IP'])
hitters_df = hitters_df.dropna(subset=['PA'])
# filter no at bats and no innings pitched
pitchers_df = pitchers_df[pitchers_df['IP'] != 0]
hitters_df = hitters_df[hitters_df['PA'] != 0]
# remove infinite ERA values
pitchers_df.replace(np.inf, np.nan, inplace=True)
pitchers_df.dropna(subset=['ERA'], inplace=True)
```

Sorting

This time, however, instead of creating our rookie and veteran subsets of data, we merged the hitting and pitching datasets together to get a data frame of all MLB players, since both hitters and pitchers have the relevant stats.

Code Snippet 8

```
### prepare data for comparisons
## merge for workload/age comparison
all_players_df = pd.merge(pitchers_df, hitters_df, how='outer')
## find the average workload for all ages
avg_workload_per_age = all_players_df.groupby('Age')['G'].mean().reset_index()
```

Compare & Visualize

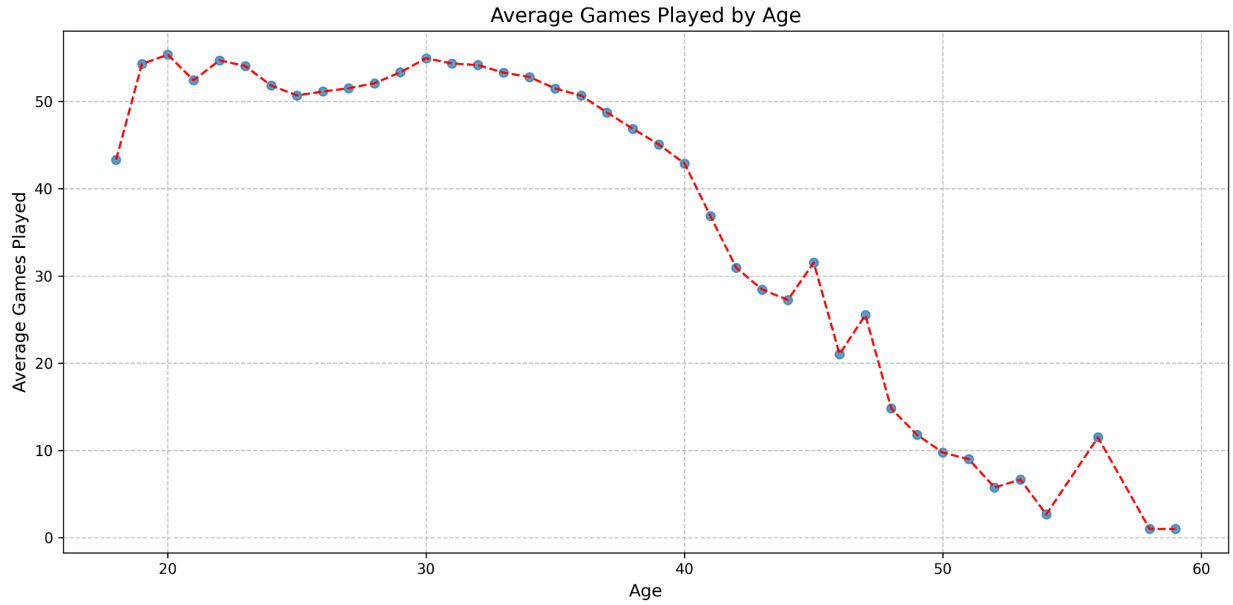
Now, all that's left to do is make our plot and see what we can learn from our data.

Code Snippet 9

```
### compare & visualize

plt.figure(figsize=(12, 6))
plt.scatter(avg_workload_per_age['Age'], avg_workload_per_age['G'], alpha=0.7)
plt.plot(avg_workload_per_age['Age'], avg_workload_per_age['G'], color='red',
linestyle='--')
plt.title('Average Games Played by Age', fontsize=14)
plt.xlabel('Age', fontsize=12)
plt.ylabel('Average Games Played', fontsize=12)
plt.grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()

plt.savefig('workload_per_age.png', dpi=300)
plt.close()
```

Visualization 3

Average Games Played by Age



Findings

Again, what we found was unsurprising: rookies don't play many games when they start, but quickly reach their prime playing years. They gradually play fewer and fewer games after reaching around thirty years old, and once they reach around the age of forty, their playing time sharply declines.

National Basketball Association

Note: All code snippets in this section are from *Trends_Sports_NBA.ipynb* by Reddy (2024), available at https://github.com/chrissouk/CS-176-Final-Project.

Filtering

Filtering our data within the NBA dataset (Cirtautas, 2023) allowed us to look at a specific season while also comparing it to the other seasons instead of completely disregarding them. For our first NBA visualization, we filtered the data so that it only returns international

players in the 2022-2023 NBA season, showing the effect international players have in the NBA to demonstrate the global influence that the sport has. We also took similar steps to filter out all international players to find just American players so that we could compare players from two categories: International vs USA.

Similar filtering steps were also used to figure out players drafted in 2018 vs 2022 in aid to find trends amongst Rookies vs Veterans.

Code Snippet 10

```
import pandas as pd

df = pd.read_csv('all_seasons.csv')

# Filters for international players in the 2022 NBA season
international_players_2022 = df[(df['country'] != 'USA') & (df['season'] == '2022-23')]

# Shows the filtered df
print(international_players_2022)
```

```
import pandas as pd

df = pd.read_csv('all_seasons.csv')

# Filters for only American born players in the year 2022
american_players_2022 = df[(df['country'] == 'USA') & (df['season'] == '2022-23')]

# Shows the filtered dataframe
print(american_players_2022)
```

```
import pandas as pd

# Load the dataset
df = pd.read_csv('all_seasons.csv')

# Shows the drafted players in the 2022 NBA season
df = df[df['draft_year'] != 'Undrafted']

# Converts the draft year to a numeric
df['draft_year'] = df['draft_year'].astype(int)

# Filters the dataset to only find players who were drafted in 2022
players_drafted_2022 = df[df['draft_year'] == 2022]

# Shows the filtered df
print("Players Drafted in 2022:")
print(players_drafted_2022)
```

```
import pandas as pd

# Load the dataset
df = pd.read_csv('all_seasons.csv')

#Shows the drafted players in the 2018 NBA season
df = df[df['draft_year'] != 'Undrafted']

# Converts the draft year to a numeric
df['draft_year'] = df['draft_year'].astype(int)

# Filter for players who were drafted in the year 2018
players_drafted_201 = df[df['draft_year'] == 2018]

# Shows the filtered DataFrame
print("Players Drafted in 2018:")
print(players_drafted_2018)
```

Sorting

Filtering for all American and International players in the 2022-2023 NBA season resulted in a vast amount of data. We then sorted the data to show the top 5 American and International players that had the most points per game, as this is arguably one of the most important statistics in basketball as a winning team is defined by the number of points they can score. This consolidated the data and made it more effective to understand which players (International or American) had greater scoring averages across the 2022-2023 NBA season. Sorting the data like this also made it easier to understand what visualization technique would be best to compare the top scoring averages from either an International or American player.

Code Snippet 11

```
import pandas as pd

# Load the dataset
df = pd.read_csv('all_seasons.csv')

# Filters data so that only items in the 2022-2023 NBA season are
season_2022 = df[df['season'] == '2022-23']

#Filters the dataset so that the top 5 international scorers are shown
international_players_2022 = season_2022[season_2022['country'] != 'USA']
top_5_international_scorers = international_players_2022.sort_values(by='pts', ascending=False).head(5)

#Filters the dataset so that the top 5 American scorers are shown
american_players_2022 = season_2022[season_2022['country'] == 'USA']
top_5_american_scorers = american_players_2022.sort_values(by='pts', ascending=False).head(5)

# Display the results of the top 5 International and American Scorers including their name, points, team, and country are shown
print("Top 5 International Scorers (2022 NBA Season):")
print(top_5_international_scorers[['player_name', 'pts', 'team_abbreviation', 'country']])

print("Top 5 American Scorers (2022 NBA Season):")
print(top_5_american_scorers[['player_name', 'pts', 'team_abbreviation', 'country']])
```

Pivoting

Pivoting data is valuable because it allows for better organization, analysis, and interpretation of complex datasets. It ensures that data is easy to read and helps people analytically visualize numbers. Using the pivot function showed the differences between filtering the data normally versus using the function that displayed the data in a way that was very easy to understand.

Code Snippet 12

```
import pandas as pd


# Filters data for the 2022-2023 NBA season
season_2022 = df[df['season'] == '2022-23']

# Filters for the top 5 International scorers
international_players_2022 = season_2022[season_2022['country'] != 'USA']
top_5_international_scorers = international_players_2022.sort_values(by='pts', ascending=False).head(5)

# Filters for the top 5 American scorers
american_players_2022 = season_2022[season_2022['country'] == 'USA']
top_5_american_scorers = american_players_2022.sort_values(by='pts', ascending=False).head(5)

# Combine the top players into one df
combined_top_scorers = pd.concat([top_5_international_scorers, top_5_american_scorers])

#Pivot the data to have the columns represented as countries and the player statistics as values
pivoted_data = combined_top_scorers.pivot(index='player_name', columns='country', values='pts')

print("Pivoted Data: Points by Country")
print(pivoted_data)
```

Figure 1

Pivoted Data: Points by Country

Pivoted Data: Points by Country

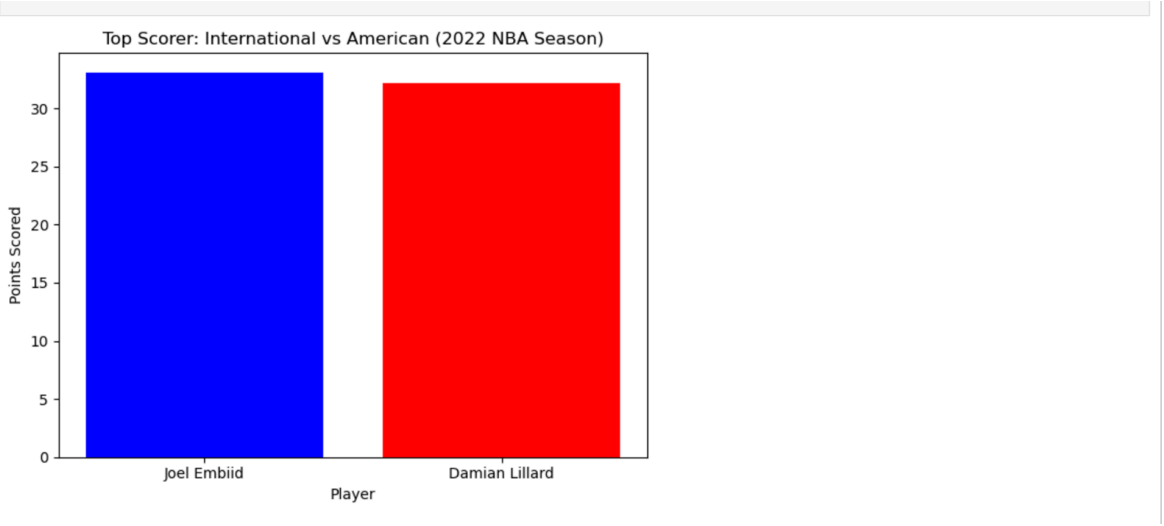
country	Australia	Cameroon	Canada	Greece	Slovenia	USA
player_name						
Damian Lillard	NaN	NaN	NaN	NaN	NaN	32.2
Giannis Antetokounmpo	NaN	NaN	NaN	31.1	NaN	NaN
Jayson Tatum	NaN	NaN	NaN	NaN	NaN	30.1
Joel Embiid	NaN	33.1	NaN	NaN	NaN	NaN
Kevin Durant	NaN	NaN	NaN	NaN	NaN	29.1
Kyrie Irving	27.1	NaN	NaN	NaN	NaN	NaN
LeBron James	NaN	NaN	NaN	NaN	NaN	28.9
Luka Doncic	NaN	NaN	NaN	NaN	32.4	NaN
Shai Gilgeous-Alexander	NaN	NaN	31.4	NaN	NaN	NaN
Stephen Curry	NaN	NaN	NaN	NaN	NaN	29.4

International vs. American Players

The dataset was filtered in two different ways to account for two different visualizations. The first way the data set was filtered was to find the top American and International scorers in the NBA and then create a visualization based on that. To do that we first filtered the data set for only international NBA players in the 2022-2023 NBA season - this is the most recent season in the NBA dataset. The same thing was then done for American-born players in the 2022-2023 NBA season. After understanding the distinction between the American and International players we accounted for the top 5 international and American scorers. This revealed a list of the top 5 players from both International and American which was later able to help formulate a graph. Using the filtered data from the code, we were able to find that Joel Embiid was the international top scorer, and Damian Lillard was the American top scorer. We then made a bar chart so that the difference in top scorers could be visualized. An international scorer scoring more points than the top American scorer in the NBA during this season shows the global influence of the sport of Basketball and why the NBA is the premier destination for all basketball players regardless of country of origin.

Visualization 4

Top Scorer: International vs American (2022 NBA Season)



Code Snippet 13

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('all_seasons.csv')

# Filter for the 2022 NBA season
season_2022 = df[df['season'] == '2022-23']

# Finds the top international scorer (Joel Embiid)
top_international_scorer = season_2022[season_2022['country'] != 'USA'].sort_values(by='pts', ascending=False).iloc[0]

# Finds the top American Scorer (Damian Lillard)
top_american_scorer = season_2022[season_2022['country'] == 'USA'].sort_values(by='pts', ascending=False).iloc[0]

# Data for the chart
players = [top_international_scorer['player_name'], top_american_scorer['player_name']]
points = [top_international_scorer['pts'], top_american_scorer['pts']]

# Bar Chart
plt.bar(players, points, color=['blue', 'red'])
plt.title('Top Scorer: International vs American (2022 NBA Season)')
plt.xlabel('Player')
plt.ylabel('Points Scored')
plt.tight_layout()
plt.show()
```

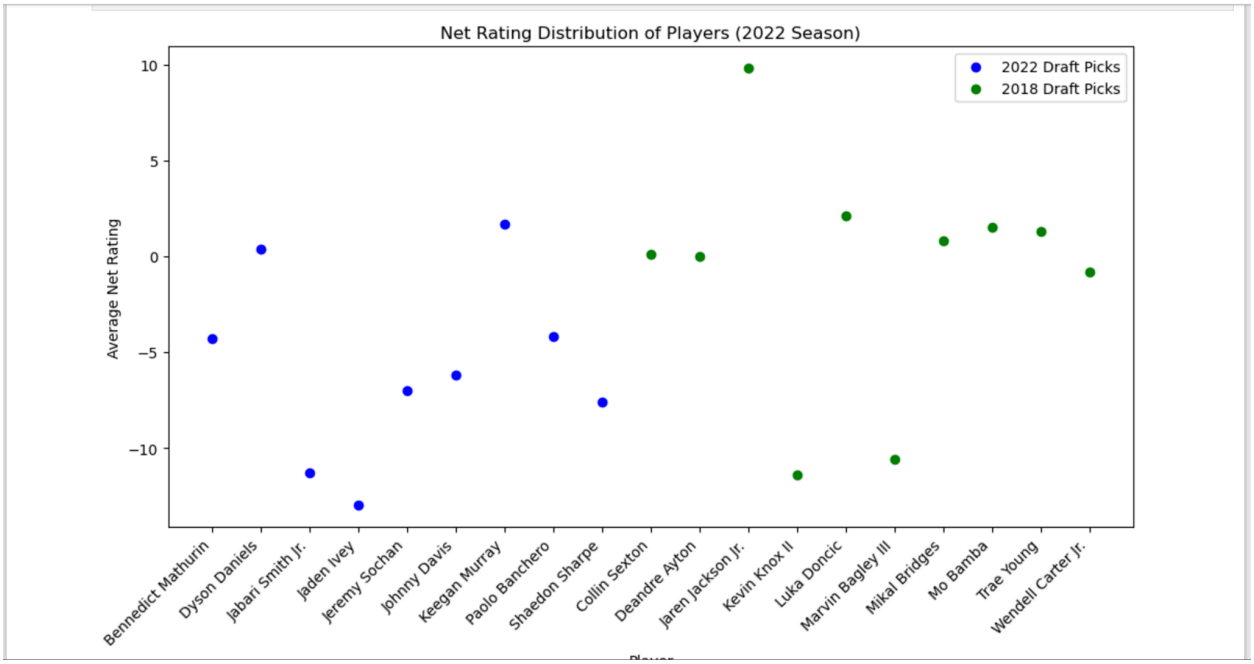
Rookies vs. Veterans

The second part of filtering data noted the distinction between veterans (Players drafted in 2018) and Rookies (Players drafted in 2022). Similar to filtering for American and

International players, similar methods were used to filter out players drafted in 2018 and 2022. Using external research we were able to deduce the top 10 players from the 2018 draft class (Veterans) and 2022 draft class (Rookies). After understanding who the top 10 players were for each of the drafts we decided to compare their net rating - the point differential per 100 possessions for their team while a specific player is on the court. From the scatter plot, we can deduce that the veterans (green) have a bigger impact on the game compared to rookies (blue). Rookies are distributed lower on the scatterplot which implies a negative net rating while Veterans are concentrated towards the top which indicates a positive net rating.

Visualization 5

Net Rating Distribution of Players (2022 Season)



Code Snippet 14

```
import pandas as pd
import matplotlib.pyplot as plt

# Split the players into two groups for different colors
#Group 1 of players represent the top 10 draft picks in the year 2022, these people are categorized as rookies
group_1 = [
    'Paolo Banchero', 'Chet Holmgren', 'Jabari Smith Jr.', 'Keegan Murray',
    'Jaden Ivey', 'Bennedict Mathurin', 'Shaedon Sharpe', 'Dyson Daniels',
    'Jeremy Sochan', 'Johnny Davis'
]

#Group 2 of players represents the top 10 draft picks in the year 2018, these people are categorized as veterans
group_2 = [
    'Deandre Ayton', 'Marvin Bagley III', 'Luka Doncic', 'Jaren Jackson Jr.',
    'Trae Young', 'Mo Bamba', 'Wendell Carter Jr.', 'Collin Sexton',
    'Kevin Knox II', 'Mikal Bridges'
]

# Filters the dataset to find the names in the 2022 NBA season set of data
group_1_stats = df[(df['player_name'].isin(group_1)) & (df['season'] == '2022-23')]
group_2_stats = df[(df['player_name'].isin(group_2)) & (df['season'] == '2022-23')]

# Takes the player name and also their mean net rating
group_1_net_ratings = group_1_stats.groupby('player_name')['net_rating'].mean().reset_index()
group_2_net_ratings = group_2_stats.groupby('player_name')['net_rating'].mean().reset_index()
```

```
# Create a scatter plot for the net ratings of players from both groups in the 2022 season
plt.figure(figsize=(12, 6))

# Scatter plot for group 1 (players from 2022 draft)
plt.scatter(group_1_net_ratings['player_name'], group_1_net_ratings['net_rating'], color='blue', label='2022 Draft Picks')

# Scatter plot for group 2 (players from 2018 draft)
plt.scatter(group_2_net_ratings['player_name'], group_2_net_ratings['net_rating'], color='green', label='2018 Draft Picks')

# Creates the defining characteristics of the scatterplot
plt.title('Net Rating Distribution of Players (2022 Season)')
plt.xlabel('Player')
plt.ylabel('Average Net Rating')
plt.xticks(rotation=45, ha='right')
plt.legend()

# Display the plot
plt.show()
```

Findings

Through making visualizations and comprehensively understanding the dataset better we found that the NBA has more global influence than we initially had thought. Most of the top scoring performances in the league are from international players which bucks against the trend of people's conclusion that basketball is an American-dominated sport. It was also interesting

how almost all of the rookies' net average ratings were negative; we would have initially thought the majority to hover around 0 and be a little above. The scatterplot was able to demonstrate the vast difference in net rating for some players who had just entered the league to others who had spent a couple of years in it.

National Football League

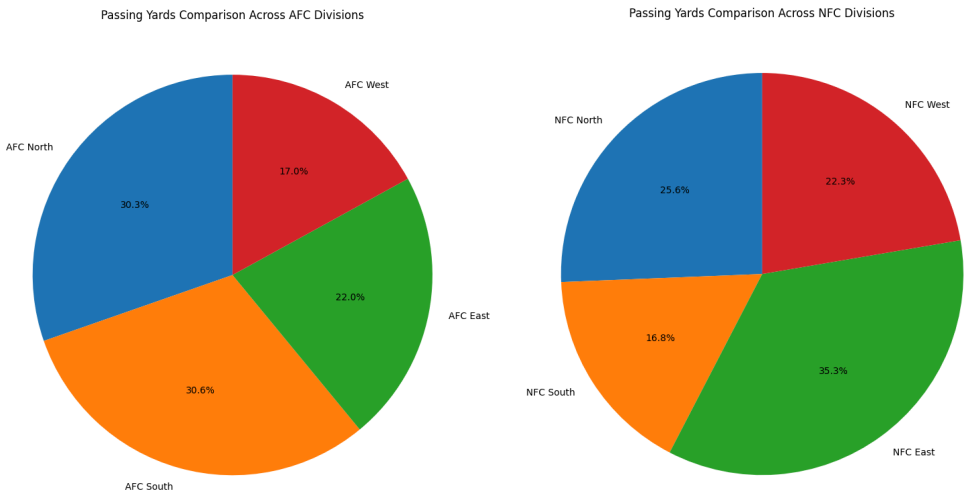
Inter-Conference Passing Comparison

Note: All code snippets in this section are from *PieChart.ipynb* by Saripalli (2024), available at https://github.com/chrissouk/CS-176-Final-Project.

The following pie charts in Visualization 6 compare the QBs from each division in the NFC and AFC based on data from our NFL dataset (Jadhav, 2024). In the NFC, the pie chart reveals a more balanced distribution of passing yards among the divisions. None of the divisions significantly outweigh the others in terms of their contribution to the total passing yards. In contrast, the AFC pie chart often shows one or two divisions contributing substantially more to the total passing yards than the others.

Visualization 6

Passing Yards Across NFL Conferences and their Divisions



Code Snippet 15

```
import pandas as pd
import matplotlib.pyplot as plt  # Importing matplotlib for plotting

data = pd.read_csv("Datasets/passing_cleaned.csv")

# Define the AFC and NFC divisions
afc_divisions = {
    "AFC North": ["PIT", "BAL", "CIN", "CLE"],
    "AFC South": ["TEN", "HOU", "IND", "JAX"],
    "AFC East": ["NE", "MIA", "BUF", "NYJ"],
    "AFC West": ["DEN", "KC", "LV", "LAC", "SD", "OAK"]
}

nfc_divisions = {
    "NFC North": ["GB", "CHI", "DET", "MIN"],
    "NFC South": ["TB", "CAR", "NO", "ATL"],
    "NFC East": ["DAL", "PHI", "NYG", "WAS"],
    "NFC West": ["SEA", "SF", "ARI", "STL", "LA"]
}

# Sum passing yards for each division
def calculate_division_yards(data, divisions):
    division_yards = {}
    for division, teams in divisions.items():
        division_yards[division] = data[data["Tm"].isin(teams)]["Yds"].sum()
    return division_yards

afc_division_yards = calculate_division_yards(data, afc_divisions)
nfc_division_yards = calculate_division_yards(data, nfc_divisions)

# Plot AFC pie chart
plt.figure(figsize=(8, 8))
plt.pie(
    afc_division_yards.values(),
    labels=afc_division_yards.keys(),
    autopct="%1.1f%%",
    startangle=90
)

plt.title("Passing Yards Comparison Across AFC Divisions")
plt.tight_layout()
plt.show()

# Plot NFC pie chart
plt.figure(figsize=(8, 8))
plt.pie(
    nfc_division_yards.values(),
    labels=nfc_division_yards.keys(),
    autopct="%1.1f%%",
    startangle=90
)
plt.title("Passing Yards Comparison Across NFC Divisions")
plt.tight_layout()
plt.show()
```

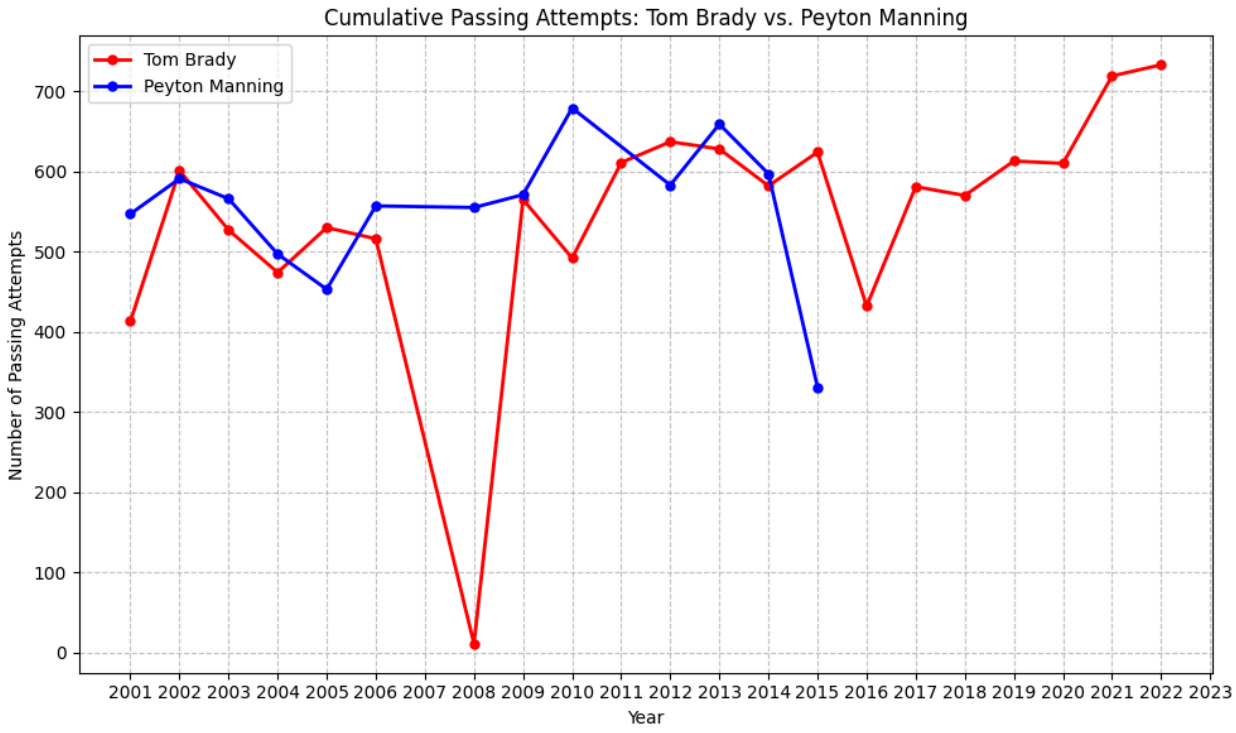
Tom Brady vs. Peyton Manning

Note: All code snippets in this section are from *Linechart.ipynb* by Saripalli (2024), available at https://github.com/chrissouk/CS-176-Final-Project.

The line chart analyzes the passing attempt trends for Tom Brady and Peyton Manning over the different seasons throughout their careers, after filtering the data to include just the two quarterbacks. Peyton Manning generally has a higher passing attempt count than Tom Brady.

Visualization 7

Passing Attempts over Different Seasons: Tom Brady vs. Peyton Manning



Code Snippet 16

```
import pandas as pd
import matplotlib.pyplot as plt

# Import data
df = pd.read_csv("Datasets/passing_cleaned.csv")

# Filter to just Tom Brady and Peyton Manning
tom_df = df[(df["Player"] == "Tom Brady")]
peyton_df = df[(df["Player"] == "Peyton Manning")]

print(df.head())

# Create the chart
fig, ax = plt.subplots(figsize=(10, 6))

ax.plot(tom_df["Year"], tom_df["Att"], label="Tom Brady", marker="o", markersize=5, linestyle="-", linewidth=2, color="red")
ax.plot(peyton_df["Year"], peyton_df["Att"], label="Peyton Manning", marker="o", markersize=5, linestyle="-", linewidth=2, color="blue")

ax.set_xticks(range(min(df["Year"]), max(df["Year"]) + 1, 1))

plt.title("Cumulative Passing Attempts: Tom Brady vs. Peyton Manning")
plt.xlabel("Year")
plt.ylabel("Number of Passing Attempts")
plt.legend(loc="upper left")
plt.grid(True, linestyle="--", alpha=0.7)
plt.tight_layout()

plt.show()
```

Conclusions

We described above how we came to understand the key statistics indicative of player performance and league evolution within the MLB, the NBA, and the NFL while analyzing trends in their dynamics.

In the case of the MLB, we were able to compare games played by rookies and veterans and concluded that veterans outperformed rookies in every category—except only HR9, as previously mentioned. We also showed the effects of aging on a player's performance. Our research shows that throughout a player's professional career, they'll face a lighter workload in their first couple of years, a long sustained peak amount of workload during midlife, and a sharp decline in their late thirties and early forties.

We researched this effect in the context of the NBA as well and proved our hypothesis of there being a growing number of basketball fans around the globe. Visualization 4 highlights scoring averages by Joel Embiid and other international players, who outperformed top American scorers like Damian Lillard during that season. That showcased the NBA's current growth into a global league, attracting talent from around the world. We gained a clearer picture of the games played by veterans and rookies by tabulating the net ratings of new and experienced players. Older players had greater net ratings, reflecting the significance of factors such as skill learning and gaining experience.

In the National Football League, we looked at the performance of quarterbacks from the two conferences. In the NFC, passing yards were more equally distributed between divisions while in the AFC, few divisions seemed to hold the advantage for most of the passing yards. Additionally, we analyzed the player profiles of Tom Brady and Peyton Manning. Although

Manning had more passing attempts cumulatively, Brady remained one of the most dependable players throughout his years of playing due to his consistently good performance.

In general, these outcomes offer insights into how sports performance is influenced by factors such as experience, country of origin, and career perspective. This project assisted us in grasping the understanding of the disparities and similarities that exist in these leagues and deepened our respect for the players and the games.

References

Anhorn, B. (2021). *MLB - Every Player in History*. Kaggle.com. https://www.kaggle.com/datasets/finkleiseinhorn/mlb-every-player-in-mlb-history/data?select=allpitchers.csv

Baseball Almanac. (2014). *Veteran Player Baseball Dictionary | Baseball Almanac*. Baseball-Almanac.com. https://www.baseball-almanac.com/dictionary-term.php?term=veteran+player

Cirtautas, J. (2023, October 13). *NBA Players*. Kaggle.com. https://www.kaggle.com/datasets/justinas/nba-players-data

Jadhav, R. (2024, April). *NFL Passing Statistics (2001-2023)*. Kaggle.com. https://www.kaggle.com/datasets/rishabjadhav/nfl-passing-statistics-2001-2023

Major League Baseball. (2023). *Rookie Eligibility | Glossary*. MLB.com. https://www.mlb.com/glossary/rules/rookie-eligibility

Reddy, P. (2024). *Trends_Sports_NBA.ipynb* [Source code]. GitHub. https://github.com/chrissouk/CS-176-Final-Project

Saripalli, A. (2024). *Linechart.ipynb* [Source code]. GitHub. https://github.com/chrissouk/CS-176-Final-Project

Saripalli, A. (2024). *PieChart.ipynb* [Source code]. GitHub. https://github.com/chrissouk/CS-176-Final-Project

Souk, J., & Petzer, J. (2024). *rookies_vs_vets.py* [Source code]. GitHub. https://github.com/chrissouk/CS-176-Final-Project

Souk, J., & Petzer, J. (2024). *workload_per_age.py* [Source code]. GitHub. https://github.com/chrissouk/CS-176-Final-Project