Analyzing the Evolution of Team Play Styles in the NHL

Problem solved and project summary:

My project addresses the problem of quantifying how the play styles of specific NHL teams, as well as the whole league, have changed over time. I aggregated select team statistics from every NHL game from 2008 to 2024 to quantify the success of teams in four areas - their offensive output, defensive success, physical presence (grit), and how well they transition from offense to defense. I calculated these four measures for every team in every game, and found the four average scores for every team in each season. My program operates with a user input/output in the terminal, where the user can receive information on five different questions:

1) The four scores of a single team in a single season - not indicative of much and are simply used for a quick lookup to ensure reliability
2) The four scores of a single team in every season - indicates if a team improved or declined over the 17-year span in each play style
3) The four scores of every team in a single season - indicates how each team performed relative to each other in each play style
4) The average four scores of every team in every season - indicates if the league as a whole improved or declined over the 17-year span in each play style
5) The top 4 teams for each score in each season - indicates if there has been consistency or turnover in which teams are most successful across each measure

The four scores are calculated in complex, but logical, statistical aggregation using standardization. I calculated the mean and average standard deviation of every stat in Python (see below), and then used that to find the z-scores of every stat in the dataset. Once, standardized, I calculated the four play style scores as the sum of all related stats, with positive stats (i.e. goals) being added, and negative stats (i.e. goals against) being subtracted. I standardized values to account for the fact that the range of each stat differs (for example, an average of 30 shots per game, but only 3 goals), and I calculated distance away from the mean in either direction. This allowed me to quantify if each team over or underperformed compared to the average team in the average season, and also ensured that play style scores with more negative stats would not automatically be negative while play style scores with more positive stats would not automatically be positive. I then weighted the stats in each aggregated score by how important they are (for example, goals scored is more important in measuring offensive success than how many shots a team takes). These are the score calculations:

1) **Offensive score** = 0.45(goals scored) + 0.1(percentage of shots that stayed in the offensive zone) - 0.05(percentage of shots that exited the offensive zone) + 0.1(expected goals) + 0.1(dangerous shots)

2) **Defensive score** = 0.15(takeaways) - 0.1(shots allowed) - 0.1(giveaways in the defensive zone) + 0.05(opponent missed shots) - 0.4(opponent goals) - 0.1(opponent expected goals) - 0.1(opponent dangerous shots)
3) **Grit score** = 0.45(hits) + 0.3(blocks) - 0.05(percentage of total game penalties taken by team) + 0.15(opponent giveaways) + 0.5(opponent giveaways in the defensive zone)
4) **Transition score** = 0.1(percentage of total faceoffs won) - 0.2(percentage of opponent shots that stayed in the offensive zone) + 0.2(percentage of opponent shots that exited the offensive zone) - 0.4(giveaways) - 0.1(opponent takeaways)

Dataset and cleaning:

I used a dataset of metrics and advanced analytics from every NHL game from 2008 to 2024. Each observation is one team and includes a range of game situations (penalties, etc) with about 100 statistical measures. For a link to the csv of the original, uncleaned dataset, click here. A link is also in the readme file on GitHub, but it is not uploaded as an actual file because it is too large. The dataset is from MoneyPuck.com.

However, I did extensive cleaning in Python using DataFrames to filter for only regular season games, full-game metrics (rather than splitting it up by situation), and only the selected metrics for my projects. This vastly shrunk the size of the dataset, so the cleaned dataset was able to be uploaded on GitHub. I also changed team abbreviations to team names in the dataset, and accounted for teams changing their abbreviation as well as relocation, and I created six new metrics as a combination of stats. The cleaned data can be found here and the notebook can be found here, and both are uploaded on GitHub. The jupyter notebook also contains the calculations of the mean and standard deviation of each included stat used to calculate the z-scores.

Explanation of code:

My Rust code is broken up into three modules. playstyles.rs defines and builds all scores, main.rs outputs different formats of scores based on the user prompts, and tests.rs tests that sample scores are built properly.

playstyles.rs:

The first public function in playstyles.rs, calculate_z_score, calculates the weighted z-score of every value in the dataset. It takes each individual value along with that stat's benchmark (mean), weight, and standard deviation as floats and returns a single float value. The z-score is calculated as the distance of the value from the mean divided by the standard deviation. I multiplied z-scores by their stat's weights in this function for simplicity, so that score calculations don't have to incorporate the weights individually.

The four scores are calculated in their own functions, but they have a pretty similar structure. Firstly, the public compute_offensive_score function takes a slice of a HashMap holding the game data (with keys as strings of statistic names and their corresponding values as strings), a team name and season as string slices, benchmarks as a list of tuples containing the mean, weight, and standard deviation of each stat as floats and a boolean value that is true if the stat is positive. It returns a single float value representing the computed offensive score. The function filters out the data for the specified team and season by iterating through each HashMap and collecting the rows with the matching team name and season. It then initiates an empty vector of z-scores to hold the computed weighted z-scores and initiates a counter for the number of games iterated through as a float. The function then iterates through every collected game and every benchmark, matching each benchmark index to its corresponding statistic to pull the necessary values for the calculation. It parses the statistic value and calculates its weighted z-score by calling the calculate_z_score function, adding the score if it is positive and subtracting it if negative. The total sum of weighted z-scores is then added together to compute the offensive score. This process is the same for the next three public functions, compute_defense_score, compute_grit_score, and compute_transition_score, with the only difference being the statistics that the functions pull in the matching process.

The last public function, compute_all_scores calls all of the score calculators and computes the score for a given team in a single season. It takes a slice of HashMaps as the game data, and a team and season as strings, and returns the four computed scores as floats. This function is where the mean, weight, standard deviation, and boolean value for positive or negative are defined for each statistic. These tuples are divided by their play style in a separate benchmark vector to be passed into the score calculators, as defined above. For example, the goals statistic belongs to the offense_benchmarks vector, while the opponent goals statistic belongs to the defense_benchmarks vector. The function returns all four scores.

main.rs:

The first function in the main.rs module reads in the csv of the dataset. It takes a path to the uploaded csv in the project folder and returns a vector of HashMaps with string keys and values. It initiates a csv reader that expects headers and loads the data from the provided file path and raises an error if the file can't be opened. It then retrieves, clones, and stores headers (column names) and initiates a mutable vector to store the HashMaps of rows. The function iterates over every row, unwraps it, and initializes an empty vector for the row. Then for every field in the data the function retrieves its corresponding header and checks if it is a team or opponent. If so, it inserts that field into the HashMap automatically, converting both the header and value to strings. Otherwise, it parses the field as a float and converts it into a string, or keeps the value as a string if it already is. Then, the header and value are inserted into the HashMap for that row. Each completed HashMap (row) is inserted into the vector of rows, and it returns the vector if there are no errors. In the main function, read_csv is called with the file path.

The prompt_for_input function interacts with the user, prompting them for which question they want to be answered and to specify the team or season if necessary. It takes a user input as a

string slice and returns a String. The function displays the prompt in the terminal, reads a line of input and stores it, then removes any whitespace and converts it into a string. The input is then returned for further computation. In the main function, a loop is initiated to continually present the five questions a user can request, followed by a seventh option to break the loop. The prompt_for_input function is first called to allow the user to request a numbered question. The loop matches the number provided by the user as a string to the related computational function. These six functions are defined above the main function in the code but are described in the paragraphs below. The main function prompts the user to provide both a team name and season for question 1, just a team name for question 2, and just a season for question 3. If option 6 is chosen, the loop breaks, and there is also a base case that returns an error message and prompts the user to choose again if anything other than the numbers 1 to 6 are inputted.

Results of code:

Question 1 is pretty straightforward. It takes the data, a team name, and a season and computes all four scores for that team and season. It calls the compute_all_scores function and returns the four scores in individual print statements. Below is a sample output for the Canadiens in 2018, and the Oilers in 2023. They don't indicate much on their own, but in comparison to the average team in the average season over the 17-year span:

- The 2018 Canadiens had a slightly high offense score, while the Oilers had a very low offense score
- The Canadiens had a slightly low defense score, while the Oilers had a slightly high defense score
- The Canadiens had a very high grit score, while the Oilers had a pretty high grit score
- The Canadiens had a pretty low transition score, while the Oilers had a slightly low transition score

```
For the Canadiens in 2018:
Offense Score: 0.15
Defense Score: -0.12
Grit Score: 0.61
Transition Score: -0.41
```

```
For the Oilers in 2023:
Offense Score: 0.54
Defense Score: 0.08
Grit Score: 0.21
Transition Score: -0.16
```

Question 2 takes the data and a team name and computes the four scores of that team in every season. It iterates over every row in the data and extracts every unique season in the code. It then iterates over each season, passing the specified team name and each season into the compute_all_scores function. Below is a sample output for the Penguins over each season. It indicates that relative to the average team in the average season, they have consistently had high offense, defense grit, and transition scores, with a few down years for each one. For reference, the Penguins won Stanley Cups (championships) in 2009, 2016, and 2017. We can draw two conclusions from the championship years:

- In 2009, the Penguins had positive scores across the board, meaning their success could be a result of being a well-rounded team

- In 2016 and 2017, the Penguins were close to average in their defense, grit, and transition scores, but might have been successful because they were able to make up for it with high offensive output

```
For the Penguins:
Scores in 2008: 0.02 Offense, 0.01 Defense, -0.09 Grit, 0.04 Transition
Scores in 2009: 0.15 Offense, 0.02 Defense, 0.11 Grit, 0.17 Transition
Scores in 2010: -0.03 Offense, 0.19 Defense, 0.24 Grit, 0.29 Transition
Scores in 2011: 0.27 Offense, 0.12 Defense, 0.15 Grit, 0.33 Transition
Scores in 2012: 0.06 Offense, 0.12 Defense, -0.01 Grit, 0.29 Transition
Scores in 2013: -0.03 Offense, 0.15 Defense, 0.00 Grit, 0.11 Transition
Scores in 2014: -0.00 Offense, 0.09 Defense, 0.30 Grit, 0.10 Transition
Scores in 2015: 0.09 Offense, 0.15 Defense, -0.01 Grit, 0.14 Transition
Scores in 2016: 0.27 Offense, -0.04 Defense, 0.07 Grit, 0.03 Transition
Scores in 2017: 0.26 Offense, -0.03 Defense, 0.04 Grit, -0.04 Transition
Scores in 2018: 0.25 Offense, -0.02 Defense, 0.33 Grit, -0.08 Transition
Scores in 2019: 0.13 Offense, 0.06 Defense, 0.19 Grit, -0.06 Transition
Scores in 2020: 0.14 Offense, 0.10 Defense, -0.15 Grit, 0.13 Transition
Scores in 2021: 0.30 Offense, 0.05 Defense, -0.01 Grit, 0.08 Transition
Scores in 2022: 0.37 Offense, -0.13 Defense, 0.37 Grit, -0.13 Transition
Scores in 2023: 0.31 Offense, 0.04 Defense, 0.05 Grit, 0.01 Transition
Scores in 2024: 0.22 Offense, -0.42 Defense, 0.41 Grit, -0.49 Transition
```

Question 3 essentially does the same thing as question 2 and has the same code structure, but it takes a specific year, rather than a team, and computes the scores for every team in that year. Below is a sample output for the most recent season, 2021. As it is very dense, here are a couple of insights:

- The whole league in 2021 seemed to be pretty close to the average performance of teams across the 17-year period, as there was good a balance between positive and negative scores
- The 2021 Stanley Cup champion was the Lightning, which had a relatively high offense score, but especially excelled in the transition game while almost every other team had a negative or near-zero transition score
- Most of the teams that made the playoffs in 2021 had very high offensive score, suggesting that a high-powered offense may have led to the most success, on average

```
In 2021:
Scores for Rangers: -0.01 Offense, 0.14 Defense, 0.00 Grit, -0.03 Transition
Scores for Sharks: -0.18 Offense, -0.04 Defense, -0.05 Grit, -0.06 Transition
Scores for Penguins: 0.30 Offense, 0.05 Defense, -0.01 Grit, 0.08 Transition
Scores for Oilers: 0.38 Offense, -0.11 Defense, 0.11 Grit, -0.16 Transition
Scores for Jets: 0.16 Offense, -0.26 Defense, 0.08 Grit, -0.15 Transition
Scores for Flyers: -0.10 Offense, -0.37 Defense, -0.04 Grit, 0.09 Transition
Scores for Stars: 0.06 Offense, -0.09 Defense, 0.02 Grit, -0.01 Transition
Scores for Devils: 0.01 Offense, -0.20 Defense, -0.20 Grit, -0.04 Transition
Scores for Lightning: 0.25 Offense, 0.03 Defense, -0.05 Grit, 0.35 Transition
Scores for Wild: 0.33 Offense, -0.07 Defense, -0.08 Grit, 0.19 Transition
Scores for Avalanche: 0.42 Offense, -0.00 Defense, -0.01 Grit, 0.16 Transition
Scores for Ducks: -0.13 Offense, -0.15 Defense, -0.13 Grit, -0.02 Transition
Scores for Golden Knights: 0.29 Offense, 0.01 Defense, -0.10 Grit, -0.03 Transition
Scores for Maple Leafs: 0.49 Offense, -0.01 Defense, 0.05 Grit, -0.07 Transition
Scores for Islanders: -0.11 Offense, -0.10 Defense, 0.09 Grit, -0.03 Transition
Scores for Canadiens: -0.14 Offense, -0.54 Defense, 0.23 Grit, -0.40 Transition
Scores for Kraken: -0.18 Offense, -0.12 Defense, -0.12 Grit, 0.09 Transition
Scores for Blues: 0.24 Offense, -0.01 Defense, -0.30 Grit, 0.02 Transition
Scores for Panthers: 0.70 Offense, -0.03 Defense, 0.29 Grit, -0.28 Transition
Scores for Sabres: -0.15 Offense, -0.24 Defense, -0.55 Grit, 0.13 Transition
Scores for Coyotes: -0.33 Offense, -0.38 Defense, -0.18 Grit, -0.19 Transition
Scores for Kings: 0.20 Offense, -0.06 Defense, -0.16 Grit, 0.29 Transition
Scores for Senators: -0.00 Offense, -0.23 Defense, 0.33 Grit, -0.16 Transition
Scores for Capitals: 0.15 Offense, 0.02 Defense, 0.07 Grit, 0.01 Transition
Scores for Blue Jackets: 0.03 Offense, -0.31 Defense, -0.24 Grit, 0.12 Transition
Scores for Blackhawks: -0.23 Offense, -0.22 Defense, 0.15 Grit, -0.10 Transition
Scores for Predators: 0.10 Offense, -0.11 Defense, 0.28 Grit, -0.07 Transition
Scores for Flames: 0.34 Offense, 0.13 Defense, 0.05 Grit, -0.04 Transition
Scores for Bruins: 0.25 Offense, 0.09 Defense, 0.11 Grit, 0.08 Transition
Scores for Red Wings: -0.06 Offense, -0.39 Defense, -0.35 Grit, 0.14 Transition
Scores for Canucks: 0.08 Offense, -0.04 Defense, 0.02 Grit, 0.14 Transition
Scores for Hurricanes: 0.39 Offense, 0.19 Defense, 0.03 Grit, -0.02 Transition
```

Question 4 can best be used to track the evolution of the four playstyles over the 17-year span, as it computes the average scores across every team in each season, relative to the average team in the average year. This function and the next both take the data alone, with no additional user inputs. The function extracts all unique seasons from the dataset and iterates over each one, initializing counters for the total of each of the four scores, as well as the number of teams. Inside this loop, it extracts all unique teams from the dataset and iterates over each one, computing the four scores for every team in every season. Skipping over a team-season pairing if the team does not exist in that season (for example, the Golden Knights were only established in 2017 and the Kraken in 2021), calculates the sum of all offense scores, defense scores, grit scores, and transition scores. The function then divides each total score for each season by the number of teams that existed in that season. Below is the output for question 4, and there are again a couple of noticeable trends:

- The increase or decrease in the average league scores tends to occur in waves, meaning each score has multiple years of positive scores and then multiple years of negative scores, rather than inconsistency year-to-year
- Years with high average offense scores generally have low average defense scores, which makes sense
- Grit scores don't seem to be correlated with either offense or defense, but transition scores seem to move with defense scores

- Compared to the late 2000s and early 2010s, NHL teams, on average, are more offensive-oriented and physical, but less defensive-oriented

```
Average Scores in 2008: -0.05 Offense, 0.05 Defense, -0.17 Grit, 0.06 Transition
Average Scores in 2009: -0.06 Offense, 0.05 Defense, -0.09 Grit, 0.07 Transition
Average Scores in 2010: -0.08 Offense, 0.09 Defense, -0.03 Grit, 0.07 Transition
Average Scores in 2011: -0.12 Offense, 0.10 Defense, -0.03 Grit, 0.10 Transition
Average Scores in 2012: -0.15 Offense, 0.10 Defense, 0.03 Grit, 0.11 Transition
Average Scores in 2013: -0.11 Offense, 0.08 Defense, 0.01 Grit, 0.09 Transition
Average Scores in 2014: -0.10 Offense, 0.08 Defense, 0.12 Grit, 0.02 Transition
Average Scores in 2015: -0.11 Offense, 0.05 Defense, 0.01 Grit, 0.07 Transition
Average Scores in 2016: -0.08 Offense, 0.04 Defense, -0.04 Grit, 0.02 Transition
Average Scores in 2017: 0.05 Offense, -0.03 Defense, 0.00 Grit, -0.06 Transition
Average Scores in 2018: 0.06 Offense, -0.04 Defense, 0.05 Grit, -0.16 Transition
Average Scores in 2019: 0.04 Offense, -0.05 Defense, -0.04 Grit, -0.10 Transition
Average Scores in 2020: -0.05 Offense, -0.03 Defense, -0.10 Grit, 0.07 Transition
Average Scores in 2021: 0.11 Offense, -0.11 Defense, -0.02 Grit, 0.00 Transition
Average Scores in 2022: 0.18 Offense, -0.12 Defense, 0.01 Grit, -0.04 Transition
Average Scores in 2023: 0.17 Offense, -0.07 Defense, 0.12 Grit, -0.01 Transition
Average Scores in 2024: 0.09 Offense, -0.20 Defense, 0.31 Grit, -0.41 Transition
```

.

Finally, question 5 answers a more qualitative question, displaying the top 4 teams for each score in each season. This shows two things - whether teams who lead the league in one aspect of the game are likely to lead the league in another aspect, and how much turnover exists year-to-year between the teams that lead each play style. Passing the data, the function again extracts all unique seasons and iterates over each one. It initiates a vector to store each team and its four computed scores for each season, and then extracts the team names in the same way as before. After iterating over every team in every season and computing the scores just like in the previous function, it defines a closure top_closure that takes a score index and returns a vector of strings with the top 4 teams for that score. A vector is then created for each score index to hold the team name and the computer value of that score, and the index is matched to its appropriate score, with 0 being offense, 1 defense, 2 grit, and 3 transition. For each play style, the teams are then sorted and the first four teams, having the highest four scores, are collected in a new vector for each play style and printed. As the output for this question is very long, displaying 16 total teams for all 17 seasons, below is a sample of just the league leaders from the past five years, 2020 to 2024, with a few insights:

- If a team leads in one play style they seem slightly more likely to lead in another play style, but there isn't much of a pattern in the two play styles that such teams lead in this smaller sample
- There has been a decent amount of turnover in the top grit teams over the past five years, but the top offensive, defensive, and transition teams seem to have less turnover

```
In 2020:
Top offense teams: Avalanche, Panthers, Golden Knights, Maple Leafs
Top defense teams: Avalanche, Bruins, Golden Knights, Islanders
Top grit teams: Canadiens, Senators, Oilers, Jets
Top transition teams: Lightning, Avalanche, Kings, Flyers
In 2021:
Top offense teams: Panthers, Maple Leafs, Avalanche, Hurricanes
Top defense teams: Hurricanes, Rangers, Flames, Bruins
Top grit teams: Senators, Panthers, Predators, Canadiens
Top transition teams: Lightning, Kings, Wild, Avalanche
In 2022:
Top offense teams: Oilers, Panthers, Devils, Hurricanes
Top defense teams: Hurricanes, Bruins, Golden Knights, Kraken
Top grit teams: Senators, Penguins, Islanders, Flames
Top transition teams: Wild, Avalanche, Canucks, Kings
In 2023:
Top offense teams: Oilers, Avalanche, Maple Leafs, Panthers
Top defense teams: Kraken, Hurricanes, Panthers, Stars
Top grit teams: Panthers, Maple Leafs, Predators, Flyers
Top transition teams: Canucks, Kraken, Rangers, Flyers
In 2024:
Top offense teams: Hurricanes, Capitals, Jets, Rangers
Top defense teams: Jets, Maple Leafs, Kings, Wild
Top grit teams: Senators, Predators, Flames, Canucks
Top transition teams: Canucks, Senators, Kings, Lightning
```

Testing in tests.rs:

The test.rs module contains three tests. The first test ensures that the z-scores are correctly calculated, testing the calculate_z_score function. It creates a sample value with an arbitrary mean, weight, and standard deviation. For a value of 10, population mean of 5, weight of 0.2, and standard deviation of 2.5, the expected z_score is (10-5)/2.5 = 2 * 0.2 = 0.4, which equals the result of calling calculate_z_score function using assert_eq!

As it is difficult to write tests for the five functions written in main.rs, because it would require extensive mathematical computations by hand, I wrote functions for two of the score calculator functions instead - offense and grit. II assigned arbitrary values to the six offense-related metrics for team “Sample” in 2013. I had to round the score calculated by compute_offensive_score to compare it with mine, and the calculations were also long, but my assert_eq! statement passed. I did the same for the five grit-related metrics for team “Sample” in 2013, assigning arbitrary values and calculating the score, and the assert_eq! statement also passed. Below is a screenshot of the three tests passing.

```
running 3 tests
test tests::test_calculate_z_score ... ok
test tests::test_compute_grit_score ... ok
test tests::test_compute_offense_score ... ok

test result: ok. 3 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s
```