Modeling Performance Outcomes in MLB: A Comparison of Top-Loaded and Balanced Rosters

Michael Drake

Abstract

MLB teams employ different roster construction strategies, with some relying heavily on a small number of star players while others emphasize depth and balance across the roster. This study explores whether these contrasting approaches—top-loaded versus balanced rosters—are associated with differences in team performance. Using publicly available MLB team-level data from multiple seasons, the analysis integrates offensive and pitching performance metrics to characterize team outcomes under each strategy.

The analysis begins with exploratory data analysis to compare overall performance patterns across the two roster types. In addition to evaluating average performance levels, this study calculates team variance in key performance metrics to assess whether balanced rosters produce more consistent results over the course of a season. Two-sample t-tests are then used to examine differences in both mean performance and variability between top-loaded and balanced teams.

The results suggest that teams with more balanced pitching rotations are more likely to make the postseason than teams with a more top-loaded rotation. On the other hand, the results suggest that teams with a more top-loaded lineup is more likely to make the postseason when compared to teams that have a more balanced lineup.

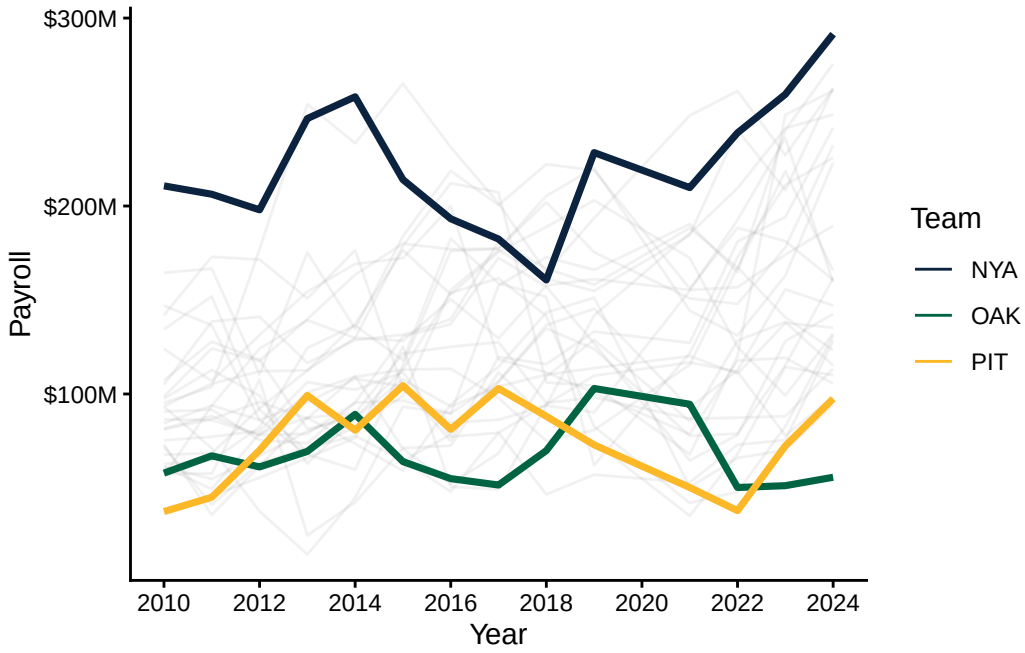
Overall, the results highlight the importance of roster depth and stability in MLB team construction. Limitations include simplifying statistical assumptions and the exclusion of additional contextual factors such as defensive performance and baserunning, which offer directions for future research.

Introduction

Major League Baseball (MLB) has become highly driven by statistics, as teams use them to evaluate players when compiling a roster. Most teams within MLB have limited budgets, such that they need to determine the best strategy for acquiring players via free agency or the trade market. The success of roster construction strategies are becoming only more imperative, as the discrepancy between the highest and lowest payroll teams has increased over the past 15 years, as shown in the figure below that highlights the highest and lowest payroll teams since 2010.

```
#Load Salaries Data from Baseball Reference
salary_paths = list.files("~/0 - Data Science Masters/Courses/Data 607/Project",
                          pattern = "salaries_\\d{4}.csv",
                          full.names = TRUE)

salaries = salary_paths |>
  set_names(basename) |>
  map(read_csv) |>
  list_rbind(names_to = "year") |>
  mutate(year = parse_number(year)) |>
  select("Tm", "year", "Est. Payroll") |>
  mutate(across("Est. Payroll", \(x) parse_number(x))) |>
  mutate(Tm = if_else(Tm == "Cleveland Guardians", "Cleveland Indians", Tm)) |>
  rename("payroll" = "Est. Payroll")

team_abr_plot = read_csv("TEAMABR.csv", col_names = c("abbreviation",
                                                      "blank", "city",
                                                      "name",
                                                      "start_year",
                                                      "end_year")) |>
  filter(end_year == 2021) |>
  mutate(name = if_else(name == "Devil Rays", "Rays", name)) |>
  select(abbreviation, name)

salaries = salaries |>
  regex_left_join(team_abr_plot, by = c("Tm" = "name")) |>
  select(abbreviation, year, payroll)

#Plot Team Salaries over Time
salaries = salaries |>
  mutate(abbreviation = factor(abbreviation, levels = unique(abbreviation)))

salaries |>
```

```
  rename(Team = abbreviation) |>
  ggplot(aes(x = year, y = payroll, color = Team, alpha = Team)) +
  geom_line(aes(alpha = Team %in% c("NYA", "PIT", "OAK"),
                size = Team %in% c("NYA", "PIT", "OAK"))) +
  theme_classic() +
  labs(x = "Year", y = "Payroll") +
  scale_color_manual(values = c(NYA = "#0C2340", PIT = "#FDB827", OAK = "#006341")) +
  scale_alpha_manual(values = c("TRUE" = 1, "FALSE" = 0.1)) +
  scale_size_manual(values = c("TRUE" = 1.25, "FALSE" = 0.5)) +
  guides(alpha = "none") +
  guides(size = "none") +
  scale_x_continuous(breaks = c(2010,2012,2014,2016,2018,2020,2022,2024)) +
  scale_y_continuous(labels = label_dollar(scale = 1/1000000, suffix = "M"))
```



There are multiple strategies that teams can utilize, but this report explores two popular strategies that are opposite in nature: Top-Loaded vs. Balanced Rosters. The idea of the Top-Loaded strategy is that teams will acquire a few highly talented players at a high price, and then fill the remaining roster with cheaper options. The Balanced Roster strategy, on the other hand, looks to build a roster with players of similar caliber at similar costs. For the analysis, two key statistics were looked at: Earned Run Average (ERA) for pitching and On-Base Plus Slugging Percentage (OPS) for hitting. Two separate analyses were conducted, one for pitching and one for hitting, because these aspects of baseball are very different from one another, so it's difficult to find a common statistic that can compare pitchers and hitters.

Data was collected from the Lahman Baseball Database and Retrosheet and was used to calculate the variance in ERA (`era_variance`) and variance in OPS (`ops_variance`) for every team from 2010-2024. To evaluate which strategy is more effective, a t-test comparing the sample means for both `era_variance` and `ops_variance` was conducted between teams that made the postseason and teams that did not. The choice for this grouping is based on the idea that teams are ultimately building a roster to succeed and get them into the postseason for an opportunity to win the world series.

Data Sets and Acquistion

Data for this project was acquired from two different sources: the Lahman Baseball Database and Retrosheet. The Lahman Baseball Database had the necessary data for the analysis available in CSV files directly on their website. The CSV files that were utilized were `pitching.csv`, `seriespost.csv`, and `teams.csv`. The code used to load in these CSV files into R is below.

```
#Load Pitching Data from Lahman Baseball Database
pitching_data = read_csv("pitching.csv")

#Load Postseason Data from Lahman Baseball Database
postseason_data = read_csv("seriespost.csv")

#Load Team Data from Lahman Baseball Database
teams_data = read_csv("teams.csv")
```

After the data from these CSV files were loaded in, the data within these CSV files were filtered and transformed to meet certain requirements of the analysis.

For the `pitching_data` dataframe, a new variable called `perc_gs` was created based on the quotient of the number of Games Started (GS) and the number of Games (G) the player appeared in. This new variable, along with `yearID` and `IPouts`, were used to filter the data to the years 2010-2024 (excluding 2020) and to players that recorded at least 180 outs, with at least 50% of appearances being from games started. The 2020 season was excluded due to the pandemic shortened season that also had situations and rules atypical to a normal season. The player filters were used to eliminate players that made only a few appearances throughout the season that may have had extreme statistics that could skew the data.

For the `postseason_data` dataframe, the data was filtered to the years 2010-2024, excluding 2020, for reasons mentioned above.

For the `teams_data` dataframe, the data was filtered to the years 2010-2024, excluding 2020, for reasons mentioned above. Additionally, the dataframe was filtered to only include columns of interest. Finally, a new variable called `win_percentage` was created based on the quotient of the number of Wins (W) and the total number of Games (G) the team played.

The code used to perform this filtering and transformations is below.

```
#Filter and Transform Pitching Dataframe
pitching_data = pitching_data |>
  mutate(perc_gs = GS/G) |>
  filter(yearID > 2009, yearID != 2020, IPouts >= 180, perc_gs >= 0.5) |>
  arrange(yearID, teamID)

#Filter Postseason Dataframe
postseason_data = postseason_data |> filter(yearID > 2009, yearID != 2020)

#Filter and Transform Teams Dataframe
teams_data = teams_data |>
  filter(yearID > 2009, yearID != 2020) |>
  select(yearID, teamID, G, W, L, ERA) |>
  mutate(win_percentage = W/G)
```

Most of the data from Retrosheet was gathered via webscraping techniques. The exception was `TEAMABR.csv`, which came from a CSV file that Retrosheet made available. The code below was used to load the `TEAMABR.csv` file into an R dataframe, followed by a variety of transformations and filters used to clean and organize the data for future use.

```
team_abr = read.csv("TEAMABR.csv", header = FALSE)
team_abr = team_abr |>
  rename("abbreviation" = V1, "league" = V2, "city" = V3,
         "nickname" = V4, "first_year" = V5, "last_year" = V6)
team_abr = team_abr |>
  filter(last_year >= 2010) |>
  mutate(nickname = if_else(nickname == "Colts", "Astros", nickname),
         last_year = if_else(last_year == 2021, 2024, last_year))
```

For the remaining data needed from Retrosheet, webscraping was used. Three separate functions were defined to aid in the webscraping, which are described further in the sections and code chunks below.

The first function called `baseball_urls` takes a dataframe with every year from 2010-2024 (excluding 2020) and every team abbreviation and creates a list of URLs from a common base URL. Data from Retrosheet is spread across multiple URLs, which have the same base URL with variations based on the year and the team. This function compiles these URLs into one list of URLs that is then used in both the `extract_baseball` and `extract_ops` functions, which are described below.

```
baseball_urls = function(df, team, yr, url_constant) {
  df |>
    mutate(yr_c = as.character({{ yr }}),
           full_url = str_c(url_constant, yr_c, "/W", {{ team }}, "0", yr_c, ".htm")) |>
    select(full_url)
}

all_years = list(2010:2024)

url_constant = "https://www.retrosheet.org/boxesetc/"

team_years = team_abr |>
  mutate(yearID = all_years) |>
  unnest_longer(yearID) |>
  filter(yearID >= first_year, yearID <= last_year, yearID != 2020) |>
  select(yearID, abbreviation)

all_urls = baseball_urls(team_years, abbreviation, yearID, url_constant)
all_urls = as.list(all_urls)

all_urls$full_url |> head(5)
```

```
[1] "https://www.retrosheet.org/boxesetc/2010/WFLO02010.htm"
[2] "https://www.retrosheet.org/boxesetc/2011/WFLO02011.htm"
[3] "https://www.retrosheet.org/boxesetc/2010/WHOU02010.htm"
[4] "https://www.retrosheet.org/boxesetc/2011/WHOU02011.htm"
[5] "https://www.retrosheet.org/boxesetc/2012/WHOU02012.htm"
```

The second function called `extract_baseball` takes in a URL and uses `read_html` to read the information from the webpage into an HTML object. This HTML object is then used with a variety of regular expressions, including `str_match`, `str_replace_all`, and `str_extract` to search for specific data within the HTML object. Additionally, the `read_table` function is used to format the extracted data into a table, which is then mutated by using `across()` to modify multiple columns, as well as pulling information from the URL used in the function. The `extract_baseball` function is then called within the `map_dfr` function to apply `extract_baseball` on all URLs generated from the `baseball_urls` function previously discussed. Finally, the data was filtered to exclude pitchers (P) and any positions that played in less than or equal to 40 games, and a new variable for On-Base + Slugging Percentage (OPS) was calculated using On-Base Percentage (OBP) and Slugging Percentage (SLG). This data was saved to a new CSV file for later use so this function only needed to be called once.

```
extract_baseball = function(url) {

  new_data = read_html(url) |>
    html_elements("body") |>
    html_elements("pre:nth-of-type(4)") |>
    html_text2()

  Sys.sleep(1)

  new_data = new_data |> str_match(regex("Position.*(?=\\nAT PH)", dotall = TRUE))
  new_data = str_replace_all(new_data, "AT ", "")

  baseball_data = read_table(new_data)
  baseball_data = baseball_data |>
    mutate(across(G:SLG, \(x) as.numeric(x)),
           yearID = as.numeric(str_extract(url, "\\d{4}")),
           teamID = str_extract(url, "(?<=/W).{3}"))
}

hitting_data = all_urls$full_url |> map_dfr(extract_baseball)
hitting_data = hitting_data |>
  mutate(OPS = OBP + SLG)
new_hitting_data = hitting_data |>
  filter(Position != "P", G > 40)

write_csv(new_hitting_data, "hitting.csv")
```

The third function called `extract_ops` is very similar to the `extract_baseball` function, which also takes in a URL and uses `read_html` to read the information from the webpage into an HTML object. The major difference between these functions are the regular expressions used to find and filter the desired data. This data was saved to a new CSV file for later use so this function only needed to be called once.

```
extract_ops = function(url) {

  new_data = read_html(url) |>
    html_elements("body") |>
    html_elements("pre:nth-of-type(4)") |>
    html_text2()

  Sys.sleep(1)
```

```
  new_data = new_data |>
    str_match(regex("G.*(?=\\nHome)", dotall = TRUE)) |>
    str_replace("Total ", "")

  team_data = read_table(new_data)
  team_data = team_data |>
    mutate(across(G:SLG, \(x) as.numeric(x)),
           OPS = OBP + SLG,
           yearID = as.numeric(str_extract(url, "\\d{4}")),
           teamID = str_extract(url, "(?<=/W).{3}")) |>
    select(yearID, teamID, OPS)
}

team_hitting_data = all_urls$full_url |> map_dfr(extract_ops)

write_csv(team_hitting_data, "teamhitting.csv")
```

Now that all data necessary for the analysis has been extracted and saved to a variety of CSV files, all datasets were loaded in and merged into one dataframe for easier use and better organization. The code chunk below shows the code used to transform the data as necessary and merge each dataset with the appropriate keys. Within these steps, the Variance in OPS (`ops_variance`) and the Variance in ERA (`era_variance`) was calculated for every team within every season. These two statistics will be the independent variables for any analysis going forward.

```
#Load Hitting Data from extract_baseball Function
hitting_data = read_csv("hitting.csv")
hitting_data = hitting_data |>
  mutate(teamID = if_else(teamID == "ANA", "LAA", teamID))

#Calculate Variance of Hitter OPS Grouped by Year and Team
team_hitting_variance = hitting_data |>
  group_by(yearID, teamID) |>
  summarize(ops_variance = var(OPS))

#Load Team Hitting Data from extract_ops Function
team_hitting_data = read_csv("teamhitting.csv")
team_hitting_data = team_hitting_data |>
  mutate(teamID = if_else(teamID == "ANA", "LAA", teamID))
team_hitting_data = team_hitting_data |>
  left_join(team_hitting_variance, join_by(yearID, teamID))
```

```
#Calculate Variance of Pitching ERA Grouped by Year and Team
team_pitching_data = pitching_data |>
  group_by(yearID, teamID) |>
  summarize(num_pitchers = n(), era_variance = var(ERA))

#Merge Teams Dataframe and Team Pitching Dataframe
teams_data = teams_data |>
  left_join(team_pitching_data, join_by(yearID, teamID))

#Merge Teams Dataframe and Team Hitting Dataframe
teams_data = teams_data |>
  left_join(team_hitting_data, join_by(yearID, teamID))

#Calculate Total Number of Wins in Postseason Grouped by Year and Team
post_season_wins = postseason_data |>
  group_by(yearID, teamIDwinner) |>
  summarize(total_wins = sum(wins))

post_season_losses = postseason_data |>
  group_by(yearID, teamIDloser) |>
  summarize(total_wins = sum(losses))

all_post_season_wins = post_season_wins |>
  full_join(post_season_losses, join_by(yearID, teamIDwinner == teamIDloser)) |>
  mutate(total_wins.x = ifelse(is.na(total_wins.x), 0, total_wins.x),
         total_wins.y = ifelse(is.na(total_wins.y), 0, total_wins.y),
         postseason_wins = total_wins.x + total_wins.y) |>
  rename("teamID" = teamIDwinner) |>
  select(yearID, teamID, postseason_wins)

#Merge Updated Teams Dataframe with Updated Postseason Dataframe
teams_data = teams_data |>
  left_join(all_post_season_wins, join_by(yearID, teamID))

#Add Variable for if Team Made Postseason
teams_data = teams_data |>
  mutate(made_postseason = as.factor(if_else(is.na(postseason_wins), "No", "Yes")))

teams_data |>
  select(-G) |>
  head(5) |>
  kable(digits = 3) |>
```

```
  kable_styling(font_size = 5)
```

yearID	teamID	W	L	ERA	win_percentage	num_pitchers	era_variance	OPS	ops_variance	postseason_wins	made_postseason
2010	LAA	80	82	4.04	0.494	6	1.286	0.701	0.006	NA	No
2011	LAA	86	76	3.57	0.531	5	1.293	0.715	0.004	NA	No
2012	LAA	89	73	4.02	0.549	5	0.774	0.765	0.006	NA	No
2013	LAA	78	84	4.23	0.481	6	1.237	0.743	0.007	NA	No
2014	LAA	98	64	3.58	0.605	6	0.526	0.728	0.007	0	Yes

Exploratory Data Analysis

The table below provides a summary of some key data for the dataset, broken out by whether the team made the postseason or not. The data includes both the mean and variance for both `era_variance` and `ops_variance`. Based on the table below, both the mean and variance of `era_variance` is higher for teams that did not make the postseason when compared to teams that made the postseason. The opposite is true for `ops_variance`, where the mean and variance is higher for teams that made the postseason when compared to teams that did not make the postseason. Early analysis suggests that teams should prioritize balanced pitching staffs and top-heavy lineups. The significance of these potential differences are analyzed further in the data analysis section.

```
#Summarize Data Grouped by Whether Team Made Postseason
made_postseason_summary = teams_data |>
  group_by(made_postseason) |>
  summarize(n_teams = n(),
            avg_era_var = mean(era_variance),
            sd_era_var = sd(era_variance),
            avg_ops_var = mean(ops_variance),
            sd_ops_var = sd(ops_variance))

made_postseason_summary |> kable()
```

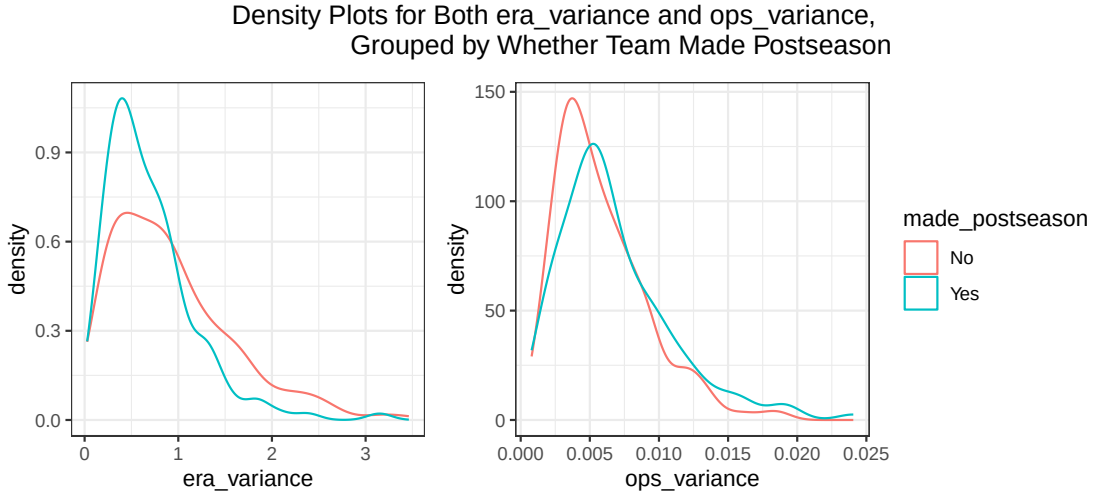
made_postseason	n_teams	avg_era_var	sd_era_var	avg_ops_var	sd_ops_var
No	278	0.9253179	0.6447771	0.0058640	0.0033303
Yes	142	0.6920167	0.4798731	0.0068025	0.0041285

To further explore the data, density plots were generated to explore the normality of the data, and boxplots were generated to explore differences in both medians and spread.

The first set of density plots show the distribution of `era_variance` and `ops_variance` for teams that made and did not make the postseason. Looking at just the normality of the

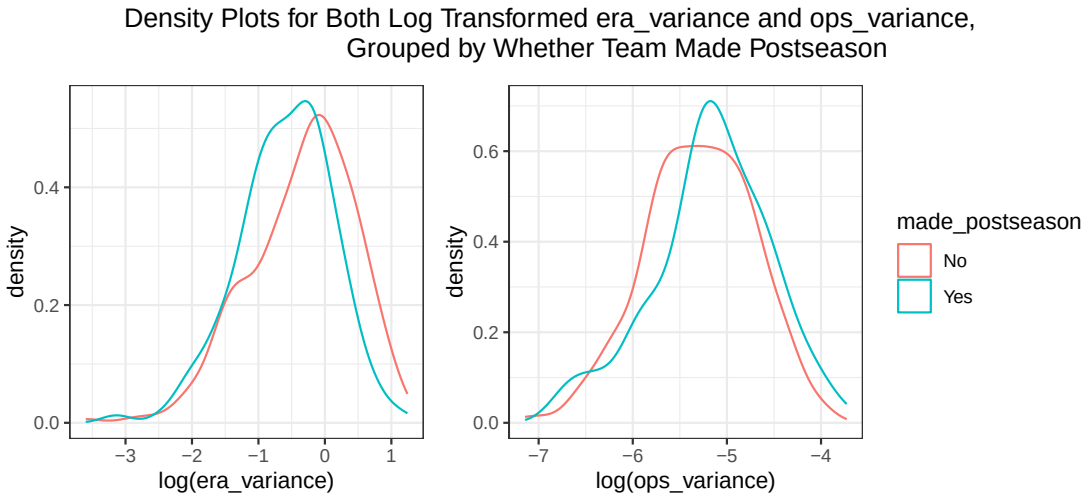
density plots, the distributions for both `era_variance` and `ops_variance` are right skewed, regardless on whether or not the team made the postseason.

```
#Create Density Plots
era_postseason_density = teams_data |>
  ggplot(aes(color = made_postseason, x = era_variance)) +
  geom_density() +
  theme_bw() +
  theme(aspect.ratio = 1, legend.position = "none")

ops_postseason_density = teams_data |>
  ggplot(aes(color = made_postseason, x = ops_variance)) +
  geom_density() +
  theme_bw()+
  theme(aspect.ratio = 1)

era_postseason_density + ops_postseason_density +
  plot_annotation("Density Plots for Both era_variance and ops_variance,
                  Grouped by Whether Team Made Postseason",
                  theme = theme(plot.title = element_text(hjust = 0.5)))
```



To better meet the normality requirement for a t-test, the data was transformed via a log transformation. The second set of density plots show the distribution of the log transformed `era_variance` and `ops_variance` data. Both density plots show that the log transformed data is nearly normal. As a result, the log transformed data will be used for the t-test comparing means.

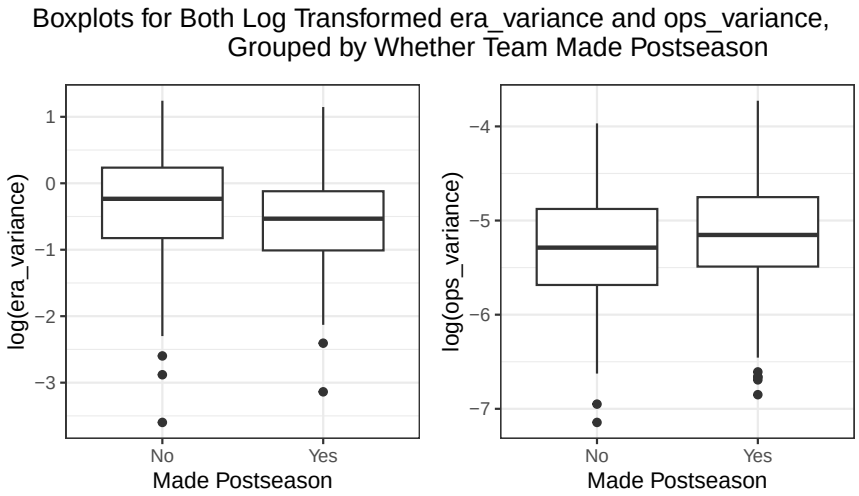
```r
#Create Density Plots
era_postseason_density = teams_data |>
  ggplot(aes(color = made_postseason, x = log(era_variance))) +
  geom_density() +
  theme_bw() +
  theme(aspect.ratio = 1, legend.position = "none")

ops_postseason_density = teams_data |>
  ggplot(aes(color = made_postseason, x = log(ops_variance))) +
  geom_density() +
  theme_bw()+
  theme(aspect.ratio = 1)

era_postseason_density + ops_postseason_density +
  plot_annotation("Density Plots for Both Log Transformed era_variance and ops_variance,
                  Grouped by Whether Team Made Postseason",
                  theme = theme(plot.title = element_text(hjust = 0.5)))
```



After transforming the data and checking the normality of the transformed data, I wanted to compare the medians and spreads for both groups. The set of boxplots below show the log transformed data for `era_variance` and `ops_variance`, grouped by whether or not the team made the postseason. Based on these boxplots, the overall spread for both `era_variance` and `ops_variance` are similar between teams that did and did not make the postseason. Comparing the medians, the median of `era_variance` for teams that did not make the postseason is higher than that of teams that did. On the other hand, the median of `ops_variance` for teams that did not make the postseason is lower than that of teams that did. This seems to match the earlier statement when looking at the summary data. Although there seems to be

a difference in means between teams that did and did not make the postseason, the following section will verify this by using a t-test.

```
#Create Boxplots
era_postseason_boxplot = teams_data |>
  ggplot(aes(x = made_postseason, y = log(era_variance))) +
  geom_boxplot() +
  theme_bw()+
  theme(aspect.ratio = 1) +
  labs(x = "Made Postseason")

ops_postseason_boxplot = teams_data |>
  ggplot(aes(x = made_postseason, y = log(ops_variance))) +
  geom_boxplot() +
  theme_bw()+
  theme(aspect.ratio = 1) +
  labs(x = "Made Postseason")

era_postseason_boxplot + ops_postseason_boxplot +
  plot_annotation("Boxplots for Both Log Transformed era_variance and ops_variance,
                  Grouped by Whether Team Made Postseason",
                  theme = theme(plot.title = element_text(hjust = 0.5)))
```



Statistical Model

In order to determine whether there is a significant difference between the means of `era_variance` and `ops_variance` for teams that did and did not make the postseason, a

t-test comparing the sample means was used. Before conducting the t-test, 2 conditions need to be met: (1) the data must be independent both within and between groups, and (2) the data must be normal. The second condition regarding normality was verified in the previous section after performing a log transformation on the data. For the first condition regarding independence, the data both within and between groups can be said to be independent because a player's and team's outcomes throughout the season are not directly determined by other players and teams.

The first t-test comparing the sample means of `log(era_variance)` between teams that made the postseason and teams that did not is shown below. The resulting t-statistic is 3.27, which gives a p-value of about 0.0012. As a result, the t-test shows that there is a significant difference between the sample means of `log(era_variance)` for teams that did and did not make the postseason when using = 0.05.

```
era_post_ttest = t.test(log(era_variance) ~ made_postseason, data = teams_data)
era_post_ttest
```

```
    Welch Two Sample t-test

data:  log(era_variance) by made_postseason
t = 3.2674, df = 314.34, p-value = 0.001205
alternative hypothesis: true difference in means between group No and group Yes is not equal
95 percent confidence interval:
 0.09917286 0.39938859
sample estimates:
 mean in group No mean in group Yes
       -0.3463349        -0.5956156
```

Looking at the 95% confidence interval and means of the two groups above, these values themselves don't mean much on their own since the data for the t-test was log transformed. To convert these into values that can be easily interpreted, the results of the t-test need to be transformed by exponentiating the values. These calculations are shown below for both the difference in means and the 95% confidence interval.

$$Difference\ in\ Means: e^{\mu_{no}-\mu_{yes}} = e^{-0.3463-(-0.5956)} = 1.2834$$

$$Confidence\ Interval: (e^{low\_interval}, e^{high\_interval}) = (e^{0.09917}, e^{0.39939}) = (1.1043, 1.4909)$$

For the difference in means, the value of 1.2834 signifies that the mean `era_variance` for teams that did not make the postseason is about 28.34% higher than that of teams that did not make the postseason. The confidence interval signifies that we are 95% confident that the true difference in means of `era_variance` between teams that did and did not make the postseason is somewhere between 10.43% and 49.09% higher for teams that did not make the postseason.

The second t-test comparing the sample means of `log(ops_variance)` between teams that made the postseason and teams that did not is shown below. The resulting t-statistic is -1.97, which gives a p-value of about 0.04936. As a result, the t-test shows that there is a significant difference between the sample means of `log(ops_variance)` for teams that did and did not make the postseason when using = 0.05.

```
ops_post_ttest = t.test(log(ops_variance) ~ made_postseason, data = teams_data)
ops_post_ttest
```

```
    Welch Two Sample t-test

data:  log(ops_variance) by made_postseason
t = -1.9746, df = 263.17, p-value = 0.04936
alternative hypothesis: true difference in means between group No and group Yes is not equal
95 percent confidence interval:
 -0.248602344 -0.000349351
sample estimates:
 mean in group No mean in group Yes
        -5.296130         -5.171654
```

Similar to the results of the t-test for `era_variance`, the results for the t-test for `ops_variance` needs to be transformed by exponentiating the values. These calculations are shown below for both the differences in means and the 95% confidence interval.

$$Difference\ in\ Means: e^{\mu_{no}-\mu_{yes}} = e^{-5.2961-(-5.1717)} = 0.8831$$

$$Confidence\ Interval: (e^{low\_interval}, e^{high\_interval}) = (e^{-0.2486}, e^{-0.0003}) = (0.7798, 0.9997)$$

For the difference in means, the value of 0.8831 signifies that the mean `ops_variance` for teams that did not make the postseason is about 11.69% lower than that of teams that did not make the postseason. The confidence interval signifies that we are 95% confident that the true difference in means of `ops_variance` between teams that did and did not make the postseason is somewhere between 22.02% and 0.03% lower for teams that did not make the postseason.

Conclusion

Based on the data exploration above, as well as the t-test comparing sample means between teams that made the postseason and did not, teams that did not make the postseason, on average, had about a 28.34% higher `era_variance` and 11.69% lower `ops_variance` than teams that made the postseason. These differences in means were significant at $= 0.05$. As a result, teams should attempt to build teams with a balanced pitching rotation, but a more top-loaded lineup for the best chance to make the postseason.

One limitation of the analysis is that the overall team ERA or OPS were not compared to the `era_variance` or `ops_variance`, respectively. This may be important for the analysis because ERA and OPS are typically a good predictor as to whether or not a team made the postseason or not. Therefore, for future analysis, it would be beneficial to see if `era_variance` is correlated with ERA and if `ops_variance` is correlated with OPS.

Another possible topic for further analysis is whether `era_variance` or `ops_variance` are different between teams that made or won the world series, compared to teams that did not. A team's ultimate goal is to win the world series, so making or winning the world series is a better outcome in a season than just making the postseason. However, one limitation with this is that grouping based on either of these factors would lead to a bigger disparity in sample sizes within the groups. For example, grouping by whether a team won the world series or not would have led to group sizes of 406 for "No" and 14 for "Yes." This could lead to some problems with regards to normality, which would thus affect the results of the t-tests.

A third topic for possible further analysis is taking other aspects of baseball into consideration, such as fielding and baserunning. Baseball has a variety of factors that can affect team success and cannot simply be summarized by hitting and pitching, making it possible that including other aspects would improve the results of the data analysis.

Citations

Baseball-Reference.com. (2025). *Major League Baseball statistics and historical data* [Data set]. Retrieved November 25, 2025 from https://www.baseball-reference.com

Lahman, S. (2025). *Lahman's Baseball Database.* Retrieved November 25, 2025, from https://www.seanlahman.com/baseball-archive/statistics/.(Datasets used: *Teams.csv, Batting.csv, Pitching.csv.*)

Retrosheet. (2025). *Retrosheet game logs and team split statistics* [Data set]. Retrieved November 25, 2025 from https://www.retrosheet.org