Comparing MLB Run Value and Awards

Andrew Eross, Owen Wassel, Nick McConnell

Introduction

Baseball has always been called "America's Pastime." However, beginning in the end of the 20th century and the early 2000's, the evolution of baseball analytics has greatly evolved the game. In the past, baseball people had just looked at offensive stats such as batting average, home runs, RBI, and strikeouts. But with the ability to track more and more data related to baseball, new stats have been developed to analyze and evaluate teams and players.

Many attribute the "Moneyball A's" as being a source renaissance for baseball analytics, where the early 2000's Oakland Athetics focused their roster building and talent acquisition by evaluating players on less popular statistics such as OBP (on base percentage), an effective way to acquire impact players who were overlooked by other teams. However, this age has now past, and new numbers drive baseball decision making.

MLB introduced Statcast in 2016, which, from the MLB website, is described as the "state-of-the-art tracking technology that allows for the collection and analysis of a massive amount of baseball data" through means such as cameras, radars, and other tracking devices. This data is accessible on Baseball Savant (an MLB licensed website). Baseball Savant takes these observations and calculates statistics related to how well players move and how balls are hit or thrown by players based on their spin, direction, and velocity. Some well-known Baseball Savant stats are hard-hit rates (how often a player hits a ball well) and xBA (the expected batting average of a player given how well they hit the ball).

The statistic we would like to analyze is Baseball Savant's run value, described on their website: "Every pitch is assigned a run value based on its outcome (ball, strike, home run, etc.). The sum of all of a player's contributions across a season, or multiple seasons, measures his overall batting or pitching run value. A positive value represents runs created for hitters, and runs prevented for pitchers." We are just looking at offensive run value, so how many runs hitters supposedly create at the plate (not including base running). This run value calculation is all "theoretical" meaning it is not derived by summing all the runs that score as a result of a hitter's plate appearance, rather how many *should* score based on how well they hit the ball, neutralizing factors such as opponent defense and teammate base running.

The calculation for run value is also not public. We have an idea of what data should be included in its derivation (such as on base percentage, home runs, etc), but Baseball Savant hides its exact calculation (or else others would copy its formula). Our report also includes research on estimating the weights of counting stats on run value, so seeing what stats might influence its calculation the most.

Run value is also calculated in each part of a hitter's zone. Depending on where the ball was pitched to the hitter, the associated run value may be attributed to the hitter's zone, shadow, chase, or waste run value. The image below shows where each of these zones are relative to the strike zone, as we will create visuals to compare hitters in different parts of the zone.

ENTER IMAGE

Research Questions

The reason why we want to use run value is to estimate the impact of a player's run value on their likelihood of winning an award. We would like to answer the following questions:

- 1.) Are the best offensive players (in terms of run value) more likely to finish atop the MVP (most valuable player) voting?
- 2.) Are other statistics (like WAR) a better predictor of who will place higher for MLB awards?
- 3.) How does this change when we analyze an award like Silver Slugger, an award that disregards defense and base running?

ENTER MORE

Data Provenance

As mentioned above, our *primary data set* is the run value data that comes from Baseball Savant, an official MLB website. However, we are also pulling the data from Baseball Reference, another online, reputable and accurate site to get the award voting data. This is our *secondary data set* and also includes many of the averages and counting stats that go with vote receivers. It is also worth noting that we are pulling this data from the 2019 season (just the regular season, not including playoffs). The reason for this was because this year was considered the "juiced ball year" meaning the physical baseball had extra bounce. This meant that the ball would bounce better off the bat and carry longer distances, helping hitters. Overall batting averages and runs scored were a relative high in 2019, meaning we have ample offensive data to draw from in a year known for its offense. Choosing any other year should reach similar conclusions (except for 2020 perhaps, where teams only played 60 games instead of 162).

Primary Data Set

- Source: Baseball Savant
- Description: display batter's run value data, calculated through Statcast measurables.
- Purpose: estimate a hitter's offensve imopact measured by calculating runs created.
- Cases: rows are hitters, colunmns have run value data overall and in each zone.

Secondary Data Set

- Source: Baseball Reference
- Description: displays award votes, who won each award, and basic statistics on these players.
- Purpose: give a basic description of a player's stats and where they finished for an award.
- Cases: rows are players who received votes for an award.
- Note: may include several tables from the same site depending on the award/conference of the player.

Data Wrangling

Because we are pulling the data from multiple data sources and combining into other data frames, several data wrangling steps must be taken. We downloaded all of the Baseball Savant run value and Baseball Reference voting data. Some important steps/ideas in the data wrangling process were as followed:

Headers

Extract the headers separately from the Baseball Reference cases. This ensures later we can better tidy the data by combining a separate header data frame to the rest of the data frame.

Tidying

In each table, make sure that column names/data are consistent. The way names appeared were different in both, so it was useful to separate names in a first and last name column. Also, rename columns to relevant, understandable names. It was also important to rename any values that contained accents, as loading the data frame would mess up those cell's formatting.

Filtering

It is important to filter out all of the pitchers who received award votes since they do not carry meaningful offensive data. We are just comparing hitters with their voting results.

Merging

An inner join was necessary on First_Name, Last_Name to combine the data frames to include run value data with voting data. Since we only wanted data on those receiving votes, only run vale data would be added to the names in the voting data frame.

Reproducibility

This process should be reproducible since it had to be repeated for data frame in both conferences. Avoiding "hard-coding" ensures that we are able to quickly change our code to produce other data frames.

FAIR Principles

Our data set must follow the FAIR principles so to ensure our data is trustworthy and valid to use.

Findable

Both data sets come from very well known baseball data pages. A quick google search of run value and 2019 award voting will take a user to these online data sets.

Accessible

Downloading the data is made very easy via the tables' settings/options on the website. This allows us to easily upload the data to our repo for our use.

Interoperable

For the most part, column names were very easy to understand what data came with it, and if not, we renamed those columns. The .csv files we downloaded from the sites were very easy to read into an R file, and understandable once doing so.

Reusable

Detailing our process should make the wrangling and EDA very reproducible for other researchers. There is adequate information on the structures of the data sets both in our documentation and from the website so other can use them for their own research.

CARE Principles

In most cases, analyzing professional sports data follows CARE principles because it is objective analysis on public data meant to study the highlights of the sport. Our research does so, uplifting the positives and accomplishments of baseball measurables. We understand we do not own any of these public data sets, and we provide proper reference information

Exploratory Data Analysis

To first tackle how well run value is related to MVP voting, we will plot a player's run value with their MVP vote points. Some context on how voting works- since MVP is conference specific, there is an award for both leagues. Therefore, we will be running this twice, one for each conference. Also, the y-axis is referenced as "Vote Points" because the way voting works is that each voter (30 in total) gets to vote for a 1st place finisher, 2nd place finisher, third place finisher, all the way down to a 10th place finisher. A player who is picked in first place gets 14 points, second place gets 9, 8 for third, and all the way down to 1 for 10th.

The number of vote points is summed for each player, and the player with the most points is the winner.

The **?@fig-nl-mvp-rv**, plots the player's run value vs vote points for the NL (National League). We see a pretty strong, positive correlation between vote points and run value. Our predictions were correct, in that players with higher run values are more likely to get more votes for the award. This of course makes sense, as the league's most valuable players should be great offensive players.

The points represent each player who received MVP votes. Originally, the player's name was listed above the point, but this was changed to improve readability.

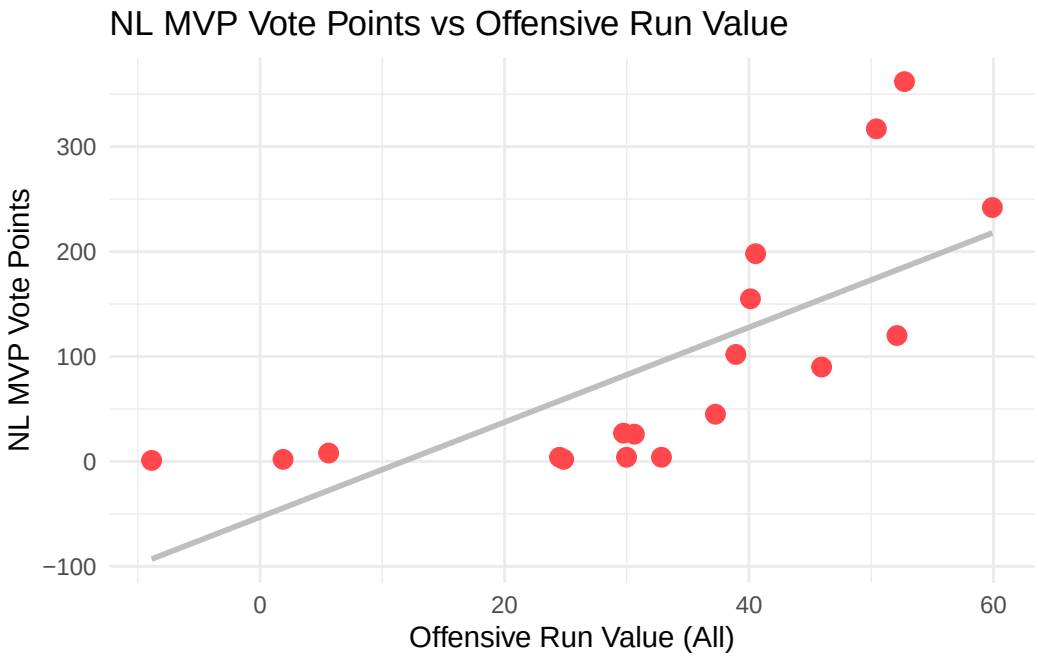


Figure 1: Run Value vs NL MVP Vote Points

The line of best fit is also graphed, with a calculated R squared value of **0.515**.

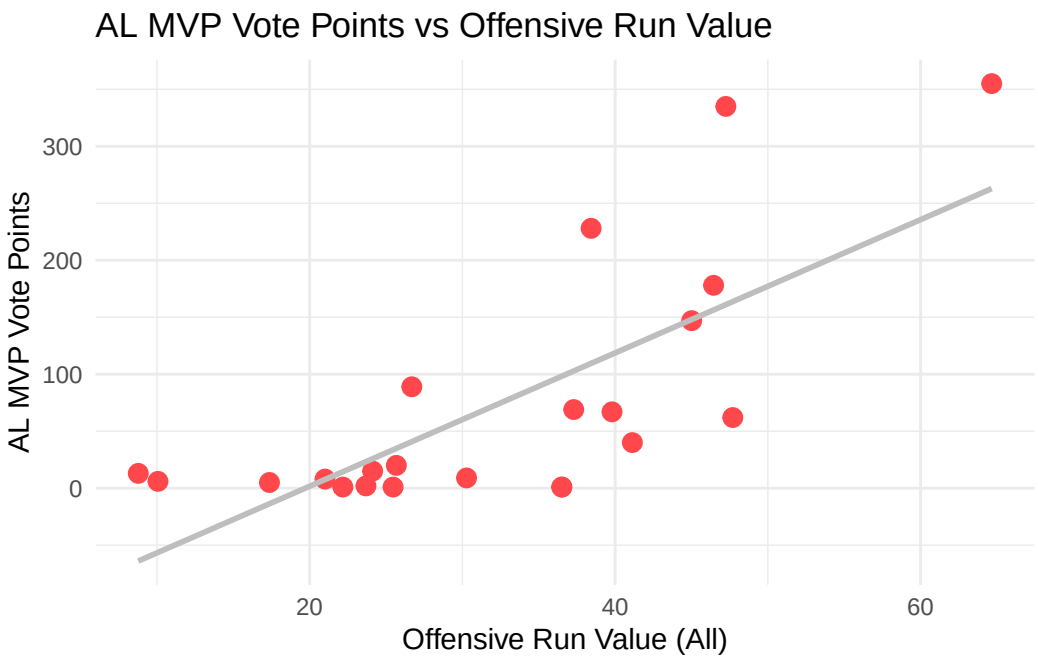


Figure 2: Run Value vs AL MVP Vote Points

We repeated this process for the AL MVP vote getters, with the similar figure shown. This produced an R squared value of **0.544**.

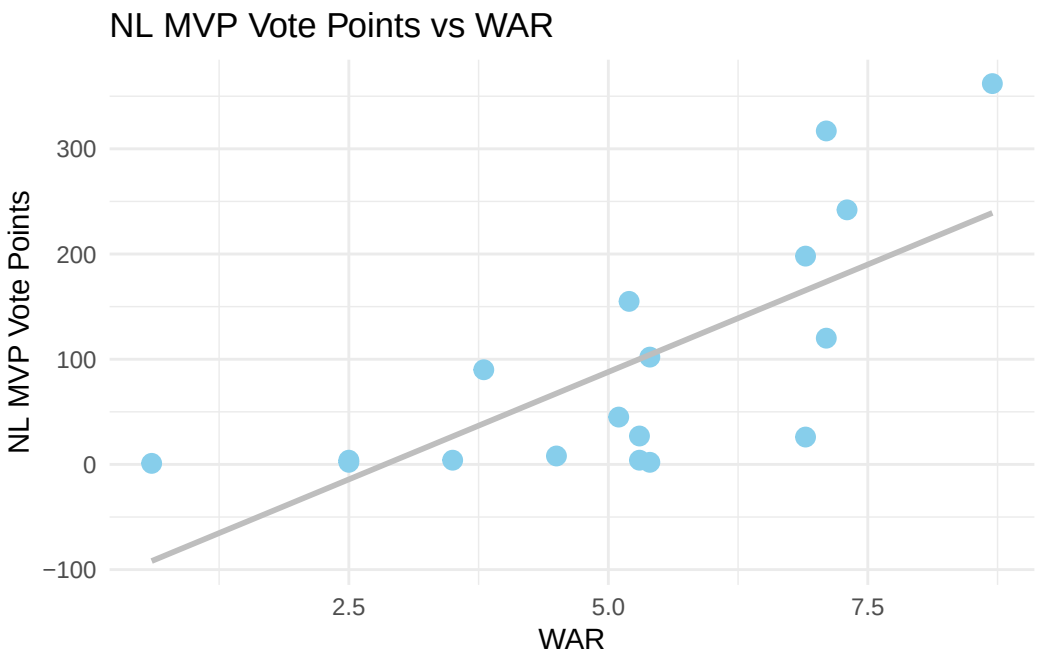


Figure 3: WAR vs NL MVP Vote Points

The R-squared value for the NL MVP model using WAR is 0.52.

The R-squared value for the AL MVP model using run value is 0.544.

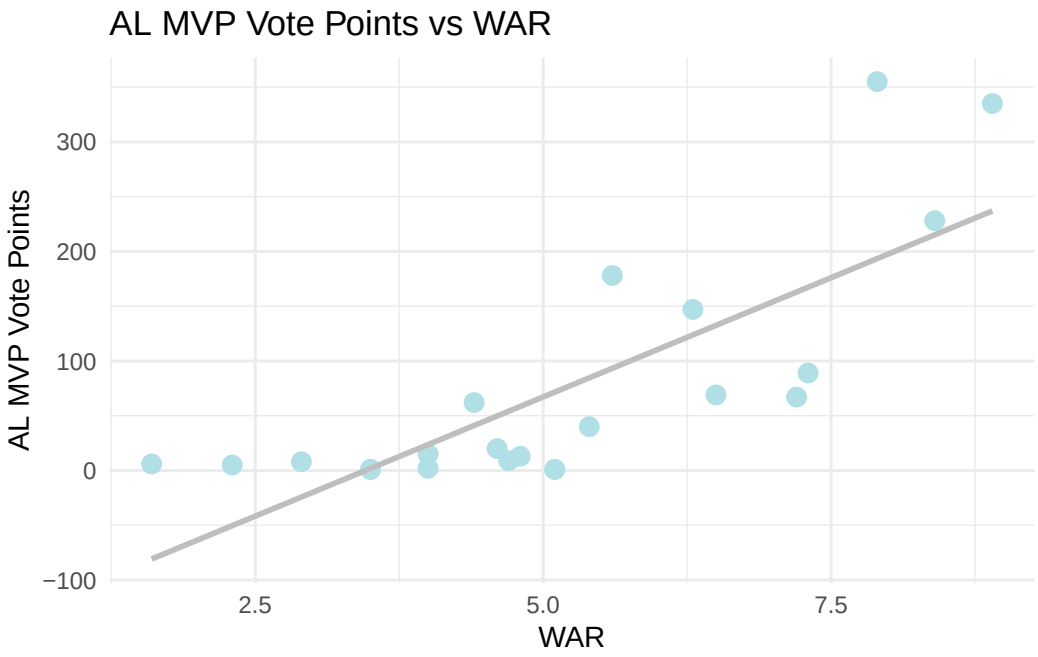


Figure 4: WAR vs AL MVP Vote Points

The R-squared value for the AL MVP model using WAR is 0.619.

Given each silver slugger from the AL and NL, we want to see what each player's run value was. Here is a table that displays that information:

```
# A tibble: 17 x 9
   Year  League Player       Position runs_all runs_heart runs_shadow runs_chase
   <chr> <chr>  <chr>        <chr>       <dbl>      <dbl>       <dbl>      <dbl>
 1 2019  NL     Realmuto, J~ C            5.61    -14.8        -3.75        15.6
 2 2019  NL     Freeman, Fr~ 1B          45.9      10.3         0.277       20.9
 3 2019  NL     Albies, Ozz~ 2B          25.6       9.05       -5.93        12.9
 4 2019  NL     Rendon, Ant~ 3B          59.9      22.2        -1.96        23.9
 5 2019  NL     Story, Trev~ SS          30.6      13.2       -19.2         20.5
 6 2019  NL     Bellinger, ~ OF          52.7      12.6         2.04        23.5
 7 2019  NL     Yelich, Chr~ OF          50.4      23.8       -10.8         21.9
 8 2019  NL     Acuña Jr., ~ OF          40.1       6.43      -10.9         28.5
 9 2019  AL     Garver, Mit~ C           25.6      11.7        -7.32        13.5
10 2019  AL     Santana, Ca~ 1B          30.3      -7.73       -8.42        31.7
11 2019  AL     LeMahieu, DJ 2B          46.4      19.5        -2.99        21.6
12 2019  AL     Bregman, Al~ 3B          47.3      -8.32       -1.58        40.2
13 2019  AL     Bogaerts, X~ SS          45.0      13.4         0.561       18.2
```

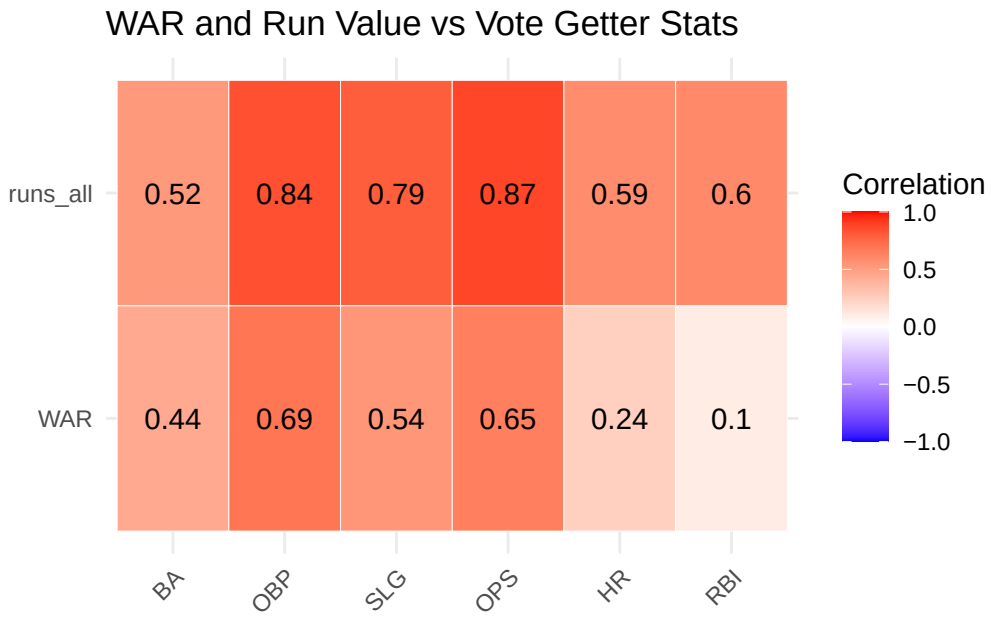
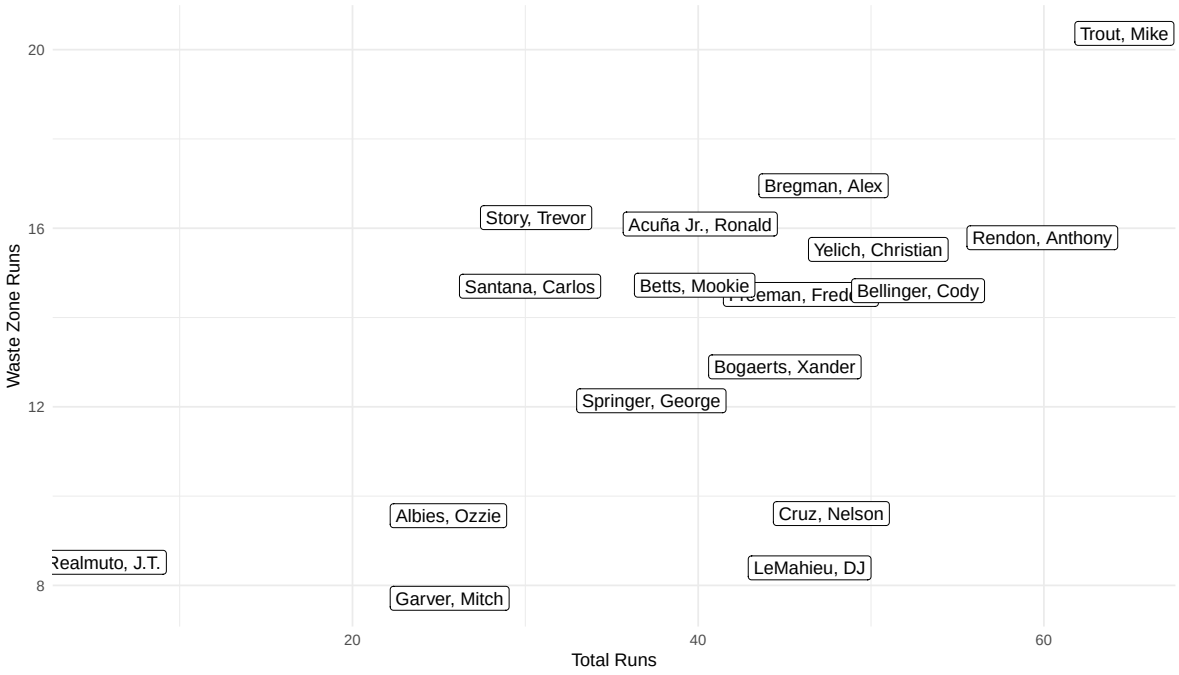


Figure 5: WAR and RV Correlations

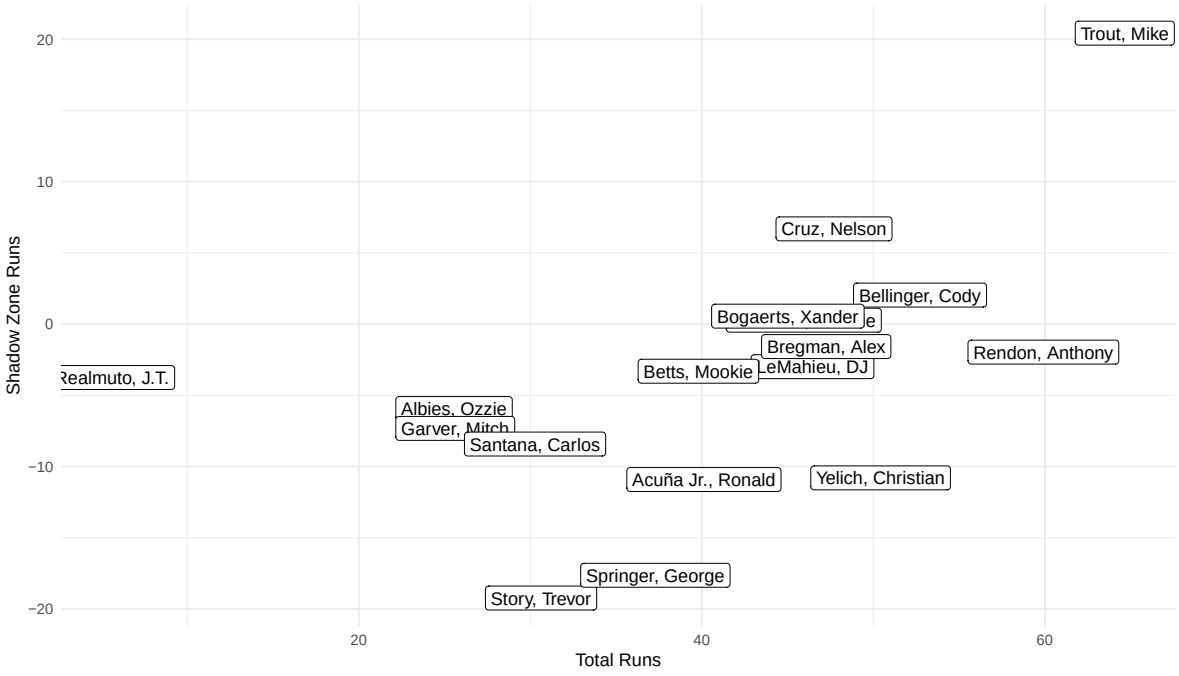
```
14 2019   AL     Betts, Mook~ OF          39.8      -0.333      -3.31        28.7
15 2019   AL     Trout, Mike  OF          64.7      -4.97       20.4         28.8
16 2019   AL     Springer, G~ OF          37.3      15.8       -17.6         26.9
17 2019   AL     Cruz, Nelson DH          47.7      18.3         6.69        13.1
# i 1 more variable: runs_waste <dbl>
```

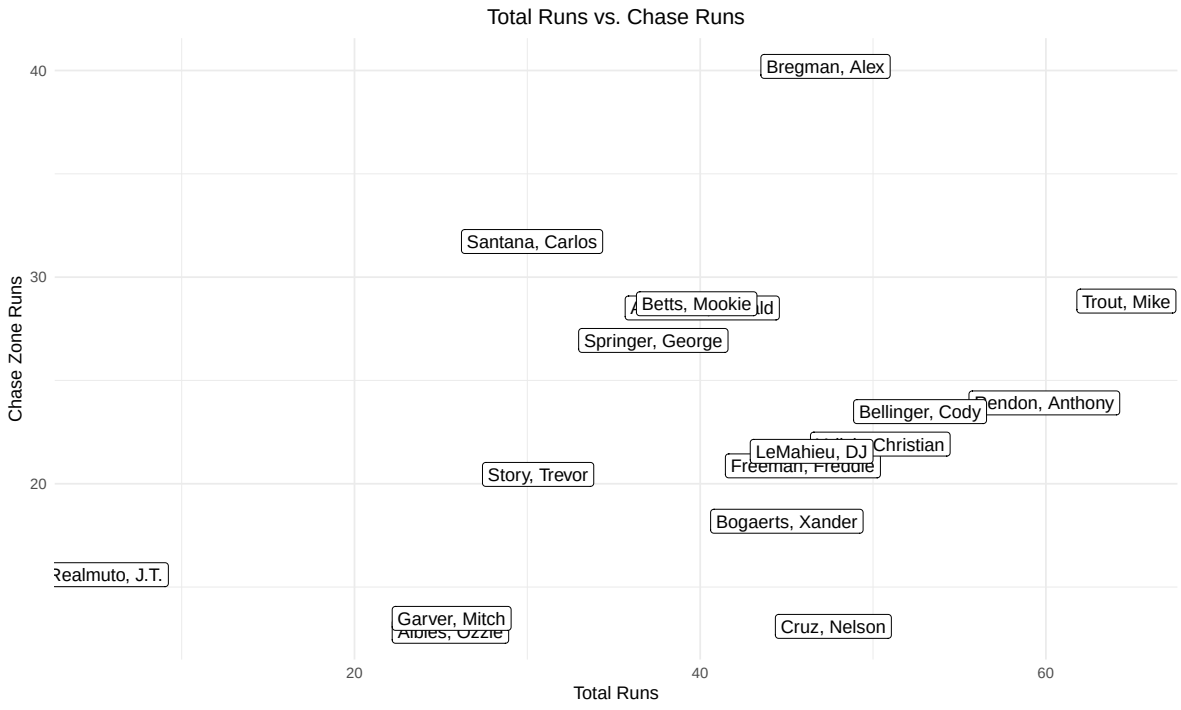
Now that we have a general idea of how many runs each silver slugger has, let's investigate each zone to see where the best hitters get most of their runs. Below are 3 graphs depicting this. Notice that Mike Trout has the most runs total (thus he is farthest right on the graph) but is lower when comparing total runs to heart runs. Meaning he gets most of his runs elsewhere.

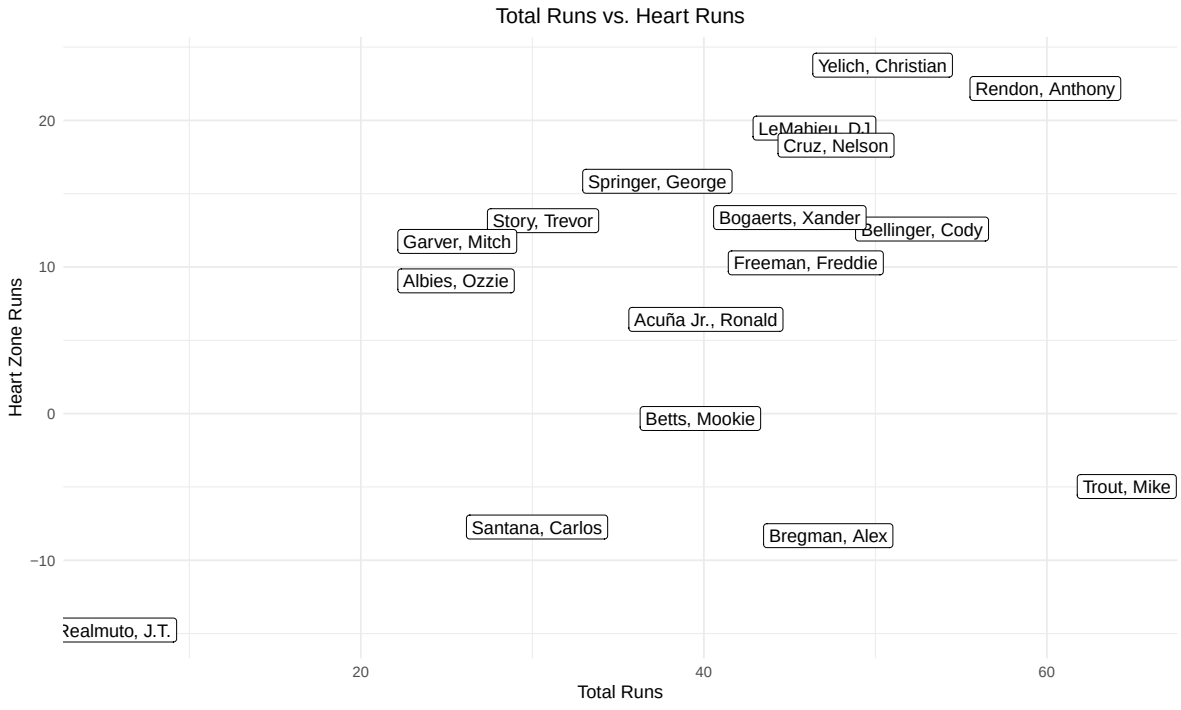
Total Runs vs. Waste Runs



Total Runs vs. Shadow Runs




Total Runs vs. Chase Runs
Chase Zone Runs
Total Runs
40
30
20
20
40
60
Bregman, Alex
Santana, Carlos
Betts, Mookie
Trout, Mike
Springer, George
Bellinger, Cody
Christian
LeMahieu, DJ
Story, Trevor
Bogaerts, Xander
Realmuto, J.T.
Garver, Mitch
Cruz, Nelson


Total Runs vs. Heart Runs
Heart Zone Runs
Total Runs
20
10
0
−10
20
40
60
Yelich, Christian
Rendon, Anthony
LeMahieu, DJ
Cruz, Nelson
Springer, George
Story, Trevor
Bogaerts, Xander
Bellinger, Cody
Garver, Mitch
Freeman, Freddie
Albies, Ozzie
Acuña Jr., Ronald
Betts, Mookie
Trout, Mike
Santana, Carlos
Bregman, Alex
Realmuto, J.T.

Code Appendix

```
# ----DATA WRANGLING FOR NL MVP RUN VALUE DATA FRAME -------

# load packages
library(tidyverse)
library(ggplot2)

# Data Wrangling ---- Primary Data set
# read run value in from downloaded baseball savant
# table, use read csv
RVData <- read.table(
  file = "2019-batters.csv",
  sep =",",
  #include headers
  header = TRUE) %>%
  #separate first and last name into two columns
  separate(last_name..first_name, into = c("Last_Name", "First_Name"), sep = ", ")

# Data Wrangling ----- Secondary Data set

# Step 1: get headers from table, which is second row
NL_MVP_Header <- read.table(
  file = "2019NLMVP.csv",
  header = FALSE,
  sep = ",",
  skip = 1,
  nrows = 1
)

# Create large data frame through many piping steps
# firt get rest of cases, non-headers
NL_MVP <- read.table(
  file = "2019NLMVP.csv",
  header = FALSE,
  sep = ",",
  skip = 2,
  # use previous headers data frame to bind as headers
  col.names = as.character(unlist(NL_MVP_Header))
) %>%
  # select only offensive stats (remove pitching stats)
```

```
  select(1:19) %>%
  # filter out pitchers (any vote getters with less than 100 abs)
  filter(AB > 100) %>%
  # separate names into first and last
  separate(Name, into = c("First_Name", "Last_Name"), sep = " ") %>%
  # tidy ill-formatted names, fix accent errors
  mutate(Last_Name = ifelse(Last_Name == "SuÃ¡rez", "Suárez", Last_Name)) %>%
  mutate(Last_Name = ifelse(Last_Name == "AcuÃ±a", "Acuña Jr.", Last_Name)) %>%
  # rename column that makes more contextual sense
  rename('1st.Place.Votes' = X1st.Place) %>%
  # join on first name and last name with run value data frame
  inner_join(RVData, by = c("First_Name", "Last_Name")) %>%
  # drop unnecessary columns
  select(-player_id, -year, -pitches, -team_id)

#------PLOTTING RUN VALUE VS VOTE POINTS FOR NL MVP -------

#create plot for run value vs NL MVP vote points
# plotting from NL MVP data frame
# x axis is run value (all zones)
# y axis is vote points
ggplot(NL_MVP, aes(x = runs_all, y = Vote.Pts)) +
  # graph dots as points, light red for run value, size 3
  # note: not including names for readability
  geom_point(color = "#FF474C", size = 3) +
  # create line of best fit/regression line, light gray
  geom_smooth(method = "lm", se = FALSE, color = "gray") +
  # x, y, and graph labels
  labs(
    title = "NL MVP Vote Points vs Offensive Run Value",
    x = "Offensive Run Value (All)",
    y = "NL MVP Vote Points"
  ) +
  # minimize background to plain
  theme_minimal()
```

```
# create model to calculate r squared value for vote points vs run value from nl mvp
model_nlmvp_rv <- lm(Vote.Pts ~ runs_all, data = NL_MVP)

# calculate r squared value from model
rsq_nlmvp_rv<- summary(model_nlmvp_rv)$r.squared
```

```r
# I researched how to do this since it was not taught in course


# ----- PLOT WAR VS VOTE POINTS FOR NL MVP HITTERS --------

# similar approach to above, just now graphing for war vs vote points
# graph war vs vote points from NL MVP data set
ggplot(NL_MVP, aes(x = WAR, y = Vote.Pts)) +
  #color the points, will use a light blue for WAR
  geom_point(color = "#87CEEB", size = 3) +
  #create a regression line/line of best fit, color a light grat
  geom_smooth(method = "lm", se = FALSE, color = "gray") +
  # graph and axes labels
  labs(
    title = "NL MVP Vote Points vs WAR",
    x = "WAR",
    y = "NL MVP Vote Points"
  ) +
  #minimize background to white
  theme_minimal()
```

```r
#create model to get r squared value between vote and war
model_nlmvp_war <- lm(Vote.Pts ~ WAR, data = NL_MVP)
# calculate r squared value with model
rsq_nlmvp_war<- summary(model_nlmvp_war)$r.squared



# - CREATE PLOT FOR RUN VALUE VS VOTE POINTS FOR AL MVP ------

# create same plot of run value data vs vote points for the AL hitters
# data comes from AL MVP data frame, x axis is run value, y axis is vote points
ggplot(AL_MVP, aes(x = runs_all, y = Vote.Pts)) +
  # same idea with light red points for run value
  geom_point(color = "#FF474C", size = 3) +
  #create the line of best fit with a light gray line
  geom_smooth(method = "lm", se = FALSE, color = "gray") +
  #create graph, x, y labels
  labs(
    title = "AL MVP Vote Points vs Offensive Run Value",
    x = "Offensive Run Value (All)",
    y = "AL MVP Vote Points"
```

```
  ) +
  #minimize background
  theme_minimal()
```

```
#create model for r squared value vs AL run value vs vote points
model_almvp_rv <- lm(Vote.Pts ~ runs_all, data = AL_MVP)
#calculate r squared value using model
rsq_almvp_rv<- summary(model_almvp_rv)$r.squared


# ---- CREATE PLOT FOR WAR VS VOTE POINTS AL MVP --------

#same plot as above just change the x axis to war
# create plot from al mvp data frame, war is x axis and vote points are y-axis
ggplot(AL_MVP, aes(x = WAR, y = Vote.Pts)) +
  #create points with a light blue
  geom_point(color = "#B0E0E6", size = 3) +
  #line of best fit in a light grat
  geom_smooth(method = "lm", se = FALSE, color = "gray") +
  # graph and axes labels
  labs(
    title = "AL MVP Vote Points vs WAR",
    x = "WAR",
    y = "AL MVP Vote Points"
  ) +
  #minimize background
  theme_minimal()
```

```
# create model for vote points and war to calculate r squared value
model_almvp_war <- lm(Vote.Pts ~ WAR, data = AL_MVP)
# calculate r squared value with model
rsq_almvp_war<- summary(model_almvp_war)$r.squared



# ---- CREATE HEATMAP FOR CORRELATIONS ------

#this is for a heat map to show correlations between RV and War for offensive stats
# this is beyond our in class curriculum so I researched the process of making a heat map

#load package
library(reshape2)
```

```r
# combine hitters in both AL and NL into one data frame
MVP_Hitters <- rbind(AL_MVP, NL_MVP) %>%
  # can drop the rank column
  select(-Rank)

# rows are the derived stats (run value and WAR)
# columns are the counting stats that factor into these stats
rows <- c("WAR", "runs_all")
cols <- c("BA", "OBP", "SLG", "OPS", "HR", "RBI")

# create the correlation data frame with the hitters data set and given rows and columns
Correlation_DF <- MVP_Hitters[, c(rows, cols)]

# use the correlation data frame to compute the correlation matrix
Correlation_Matrix <- cor(Correlation_DF, use = "complete.obs")

# make a correlation subset using the matrix and given rows and columns
Correlation_Subset <- Correlation_Matrix[rows, cols]

# correlation long object by "melting" the subset
Correlation_Long <- melt(Correlation_Subset)

# plot the correlation long where x are the counting stats and y are the run value and war
# we are filling with the gradient of correlation
ggplot(Correlation_Long, aes(x = Var2, y = Var1, fill = value)) +
  # set tiles as white
  geom_tile(color = "white") +
  # create gradient, where low is -1 and high is 1, getting redder as correlation approaches
  #create colors from blue to white to red
  scale_fill_gradient2(low = "blue",
                       high = "red",
                       mid = "white",
                       midpoint = 0,
                       limit = c(-1, 1),
                       name = "Correlation") +
  # label with black text the actual correlation
  geom_text(aes(label = round(value, 2)), color = "black", size = 4) +
  # minimize background
  theme_minimal() +
  #axes and graph labels
  labs(x = "", y = "", title = "WAR and Run Value vs Vote Getter Stats") +
  # angle the x axis text for visual aesthetic
```

```
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

```
# ---- CREATE TABLE FOR SILVER SLUGGERS AND THEIR RUN VALUE ------
# Load required libraries
library(tidyverse)
library(rvest)
library(dplyr)
library(knitr)
# Scrape NL Silver Slugger Data from Baseball reference
SSNLRawList <- read_html(x = "https://www.baseball-reference.com/awards/silver_slugger_nl.sht
  html_elements(css = "table") %>%
  html_table()
SSNLData <- bind_rows(SSNLRawList)
# Extract 2019 year from 'Year & Common' column
SSNLData <- SSNLData %>%
  mutate(Year = str_extract(`Year & Common`, "^\\d{4}"))
SSNL2019 <- SSNLData %>%
  filter(Year == "2019")
# Scrape AL Silver Slugger Data from Baseball Reference
SSALRawList <- read_html(x = "https://www.baseball-reference.com/awards/silver_slugger_al.sht
  html_elements(css = "table") %>%
  html_table()
SSALData <- bind_rows(SSALRawList)
# Extract 2019 year from 'Year & Common' column
SSALData <- SSALData %>%
  mutate(Year = str_extract(`Year & Common`, "^\\d{4}"))
SSAL2019 <- SSALData %>%
  filter(Year == "2019")
# Label leagues
SSNL2019 <- SSNL2019 %>%
  mutate(League = "NL")
SSAL2019 <- SSAL2019 %>%
  mutate(League = "AL")
# Combine AL and NL data
SS2019 <- bind_rows(SSNL2019, SSAL2019)
# Reshape data from wide to long format for positions
SS2019_long <- SS2019 %>%
  pivot_longer(
    cols = c("P", "C", "1B", "2B", "3B", "SS", "OF...7", "OF...8", "OF...9", "OF...10", "DH"
    names_to = "Position",
    values_to = "Player"
  ) %>%
```

```
  filter(Player != "") %>%
  mutate(Position = as.character(Position),
         Position = ifelse(grepl("^OF", Position), "OF", Position)) %>%
  select(Year, League, Player, Position)
# Load 2019 batters data from baseball savant
batters2019 <- read.csv("C:\\Users\\owass\\OneDrive - The Pennsylvania State University\\Docu
# Rename player column for consistency and ease when merging data frames
colnames(batters2019)[colnames(batters2019) == "last_name..first_name"] <- "Player"
# Drop unnecessary columns
batters2019 <- batters2019 %>%
  select(-c(player_id, team_id, pa, pitches))
# Convert player names to characters to edit
SS2019_long$Player <- as.character(SS2019_long$Player)
# Manually edit names to match the data from baseball savant
SS2019_long$Player[grepl("Greinke", SS2019_long$Player)] <- "Greinke, Zack"
SS2019_long$Player[grepl("Realmuto", SS2019_long$Player)] <- "Realmuto, J.T."
SS2019_long$Player[grepl("Freeman", SS2019_long$Player)] <- "Freeman, Freddie"
SS2019_long$Player[grepl("Albies", SS2019_long$Player)] <- "Albies, Ozzie"
SS2019_long$Player[grepl("Rendon", SS2019_long$Player)] <- "Rendon, Anthony"
SS2019_long$Player[grepl("Story", SS2019_long$Player)] <- "Story, Trevor"
SS2019_long$Player[grepl("Bellinger", SS2019_long$Player)] <- "Bellinger, Cody"
SS2019_long$Player[grepl("Acuña", SS2019_long$Player)] <- "Acuña Jr., Ronald"
SS2019_long$Player[grepl("Yelich", SS2019_long$Player)] <- "Yelich, Christian"
SS2019_long$Player[grepl("Garver", SS2019_long$Player)] <- "Garver, Mitch"
SS2019_long$Player[grepl("Santana", SS2019_long$Player)] <- "Santana, Carlos"
SS2019_long$Player[grepl("LeMahieu", SS2019_long$Player)] <- "LeMahieu, DJ"
SS2019_long$Player[grepl("Bregman", SS2019_long$Player)] <- "Bregman, Alex"
SS2019_long$Player[grepl("Bogaerts", SS2019_long$Player)] <- "Bogaerts, Xander"
SS2019_long$Player[grepl("Trout", SS2019_long$Player)] <- "Trout, Mike"
SS2019_long$Player[grepl("Betts", SS2019_long$Player)] <- "Betts, Mookie"
SS2019_long$Player[grepl("Springer", SS2019_long$Player)] <- "Springer, George"
SS2019_long$Player[grepl("Cruz", SS2019_long$Player)] <- "Cruz, Nelson"
# Merge data frames to display each silver slugger and their run values
merged_data <- inner_join(SS2019_long, batters2019, by = "Player")
merged_data <- merged_data %>%
  select(-c(year))
# Create table to show relationships
kable(merged_data)

# ---- CREATE GRAPHS COMPARING TOTAL RUNS TO RUNS IN OTHER ZONES
# Load ggplot2 library for graph making
library(ggplot2)
```

```r
# adjust data name for simplicity
batters <- merged_data
# Graph displaying relationship between total runs and waste zone runs
ggplot(batters) +
  aes(x = runs_all, y = runs_waste, label = Player) +
  geom_label() +
  labs(
    x = "Total Runs",
    y = "Waste Zone Runs",
    title = "Total Runs vs. Waste Runs"
  ) +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```

```r
# Graph displaying relationship between total runs and shadow zone runs
ggplot(batters) +
  aes(x = runs_all, y = runs_shadow, label = Player) +
  geom_label() +
  labs(
    x = "Total Runs",
    y = "Shadow Zone Runs",
    title = "Total Runs vs. Shadow Runs"
  ) +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```

```r
# Graph displaying relationship between total runs and chase zone runs
ggplot(batters) +
  aes(x = runs_all, y = runs_chase, label = Player) +
  geom_label() +
  labs(
    x = "Total Runs",
    y = "Chase Zone Runs",
    title = "Total Runs vs. Chase Runs"
  ) +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```

```r
# Graph displaying relationship between total runs and heart zone runs
ggplot(batters) +
  aes(x = runs_all, y = runs_heart, label = Player) +
  geom_label() +
```

```
  labs(
    x = "Total Runs",
    y = "Heart Zone Runs",
    title = "Total Runs vs. Heart Runs"
  ) +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```