Class Code

```
library(Lahman)
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.2     v tibble    3.3.0
v lubridate 1.9.4     v tidyr     1.3.1
v purrr     1.1.0
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

```
library(knitr)
library(broom)
```

Create top 10 Yankee career batting average

```
Yankees <-
  Batting |>
  filter(teamID == 'NYA') |>
  summarize(hits = sum(H),at_bats=sum(AB),.by='playerID') |>
  mutate(BA = hits/at_bats) |>
  filter(at_bats>=2500) |>
  arrange(-BA) |>
  head(10)
```

```
Yankees |>
  left_join(select(People, playerID, nameLast, nameFirst, finalGame), by = "playerID" ) |>
  left_join(select(HallOfFame, playerID, inducted, yearID)|>
              filter(inducted!="N")
            , by = "playerID") |>
  select(-playerID) |>
  relocate(nameLast,nameFirst) |>
  arrange(inducted,finalGame) |>
  kable()
```

nameLast	nameFirst	hits	at_bats	BA	finalGame	inducted	yearID
Ruth	Babe	2518	7216	0.3489468	1935-05-30	Y	1936
Combs	Earle	1866	5746	0.3247477	1935-09-29	Y	1970
Gehrig	Lou	2721	8001	0.3400825	1939-04-30	Y	1939
Dickey	Bill	1969	6300	0.3125397	1946-09-08	Y	1954
DiMaggio	Joe	2214	6821	0.3245858	1951-09-30	Y	1955
Jeter	Derek	3465	11195	0.3095132	2014-09-28	Y	2020
Meusel	Bob	1565	5032	0.3110095	1930-09-26	NA	NA
Chapman	Ben	1079	3539	0.3048884	1946-05-12	NA	NA
Mattingly	Don	2153	7003	0.3074397	1995-10-01	NA	NA
Cano	Robinson	1649	5336	0.3090330	2022-07-27	NA	NA

```
Teams |>
  filter(yearID >1999) |>
  select(teamID,W,G,WSWin) |>
  # mutate(WSWin = ifelse(WSWin=='Y',1,0)) |>
  group_by(teamID) |>
  summarize(W=sum(W),G=sum(G),WSWins= sum(WSWin=="Y")) |>
  mutate(Wpct=W/G) |>
  arrange(-Wpct)
```

```
# A tibble: 33 x 5
   teamID     W     G WSWins  Wpct
   <fct>  <int> <int>  <int> <dbl>
 1 NYA     2286  3946      2 0.579
 2 LAN     2239  3948      2 0.567
 3 SLN     2179  3945      2 0.552
 4 ATL     2152  3945      1 0.546
 5 BOS     2145  3947      4 0.543
 6 ANA      425   810      1 0.525
```

```
 7 SFN     2052  3946      3 0.520
 8 CLE     2046  3945      0 0.519
 9 OAK     2030  3946      0 0.514
10 HOU     2029  3947      2 0.514
# i 23 more rows
```

```
names(Teams)
```

```
 [1] "yearID"         "lgID"           "teamID"         "franchID"
 [5] "divID"          "Rank"           "G"              "Ghome"
 [9] "W"              "L"              "DivWin"         "WCWin"
[13] "LgWin"          "WSWin"          "R"              "AB"
[17] "H"              "X2B"            "X3B"            "HR"
[21] "BB"             "SO"             "SB"             "CS"
[25] "HBP"            "SF"             "RA"             "ER"
[29] "ERA"            "CG"             "SHO"            "SV"
[33] "IPouts"         "HA"             "HRA"            "BBA"
[37] "SOA"            "E"              "DP"             "FP"
[41] "name"           "park"           "attendance"     "BPF"
[45] "PPF"            "teamIDBR"       "teamIDlahman45" "teamIDretro"
```

Lesson 2

AY26-2 Notes

Need to change group_keys portion because if there are different numbers of rows for players or teams, the split won't match up appropriately.

```
# Step 1 - Write a function.
mostWins <- function(data){
  data |>
    arrange(-W) |>
    select(teamID, W, WSWin) |>
    head(1)
}

# Step 2 - Pull the years you will split on.
year <- Teams |>
  filter(yearID >= 2000) |>
  group_by(yearID) |>
  group_keys() |>
  pull(yearID)
```

```r
# Steps 3 and 4 - Split and apply.
winLeaders <- Teams |>
  filter(yearID >= 2000) |>
  split(year) |>
  map_df(mostWins, .id = "yearID")

winLeaders |>
  kable(caption = "Major League win leaders by season and whether they won the World Series."
```

Table 2: Major League win leaders by season and whether they won the World Series.

yearID	teamID	W	WSWin
2000	SFN	97	N
2001	SEA	116	N
2002	NYA	103	N
2003	ATL	101	N
2004	SLN	105	N
2005	SLN	100	N
2006	NYN	97	N
2007	BOS	96	Y
2008	LAA	100	N
2009	NYA	103	Y
2010	PHI	97	N
2011	PHI	102	N
2012	WAS	98	N
2013	BOS	97	Y
2014	LAA	98	N
2015	SLN	100	N
2016	CHN	103	Y
2017	LAN	104	N
2018	BOS	108	Y
2019	HOU	107	N
2020	LAN	43	Y
2021	SFN	107	N
2022	LAN	111	N
2023	ATL	104	N
2024	LAN	98	Y

Pulling in the various data frames

```
library(baseballr)

Batting |>
  filter(between(yearID, 2010,2025)) |>
  left_join(People |> select(playerID,nameLast,nameFirst,bats)) |>
  summarize(H=sum(H),AB=sum(AB),.by=c('yearID','bats')) |>
  mutate(AVG = H/AB) |>
  pivot_wider(id_cols='yearID',names_from = 'bats',values_from ='AVG')
```

```
Joining with `by = join_by(playerID)`
```

```
# A tibble: 15 x 4
   yearID     R     L     B
    <int> <dbl> <dbl> <dbl>
 1   2010 0.257 0.259 0.256
 2   2012 0.255 0.253 0.255
 3   2013 0.253 0.255 0.254
 4   2015 0.253 0.259 0.250
 5   2011 0.253 0.257 0.259
 6   2014 0.251 0.252 0.248
 7   2016 0.256 0.255 0.255
 8   2017 0.254 0.257 0.253
 9   2019 0.251 0.251 0.260
10   2021 0.246 0.242 0.239
11   2023 0.248 0.249 0.248
12   2024 0.243 0.244 0.243
13   2022 0.247 0.239 0.231
14   2020 0.251 0.238 0.234
15   2018 0.248 0.249 0.245
```

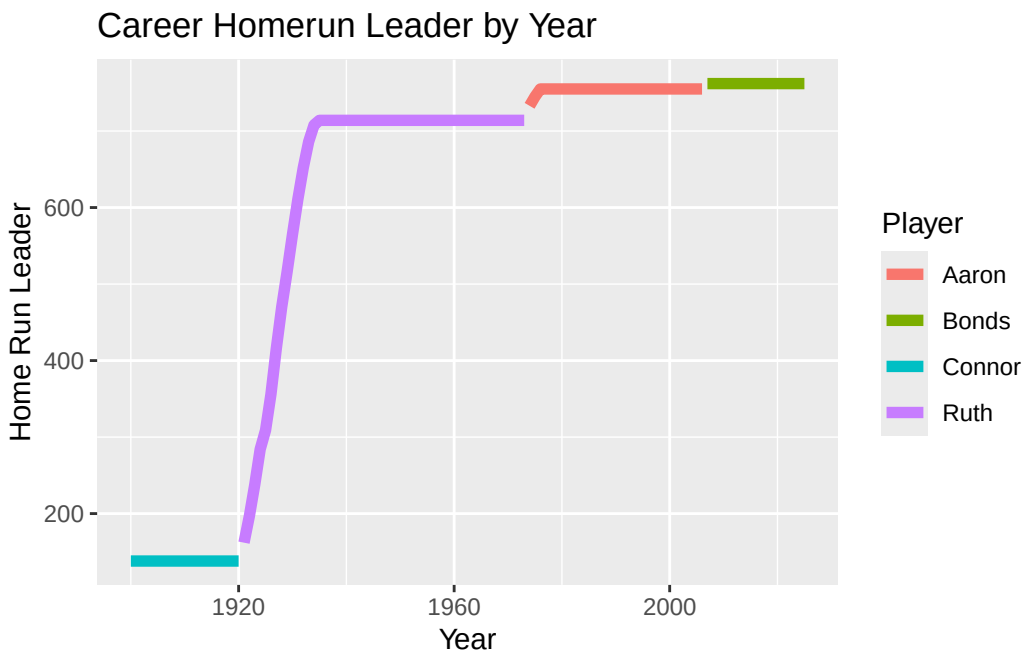
Lesson 4 In-Class

```
hr_df <-
  Batting |>
  mutate(HRcareer = cumsum(HR),.by='playerID')
```

```
hr_leader <-
  function(data, year){
    data |>
      select(yearID,playerID,HRcareer) |>
      filter(yearID<=year) |>
      slice_max(HRcareer) |>
      # arrange(-HRcareer) |>
      # head(1) |>
      mutate(year=year) |>
      relocate(year)
  }


home_runs<-
map_df(.x=1900:2025, .f = ~hr_leader(data=hr_df,year=.x)) |>
  left_join(People |> select(playerID,nameLast,nameFirst),by='playerID')
```

GGplot

```
home_runs |>
  ggplot(aes(x=year,y=HRcareer,color=nameLast))+
  geom_line(linewidth=2)+
  labs(x='Year',
       y='Home Run Leader',
       color='Player',
       title='Career Homerun Leader by Year')
```



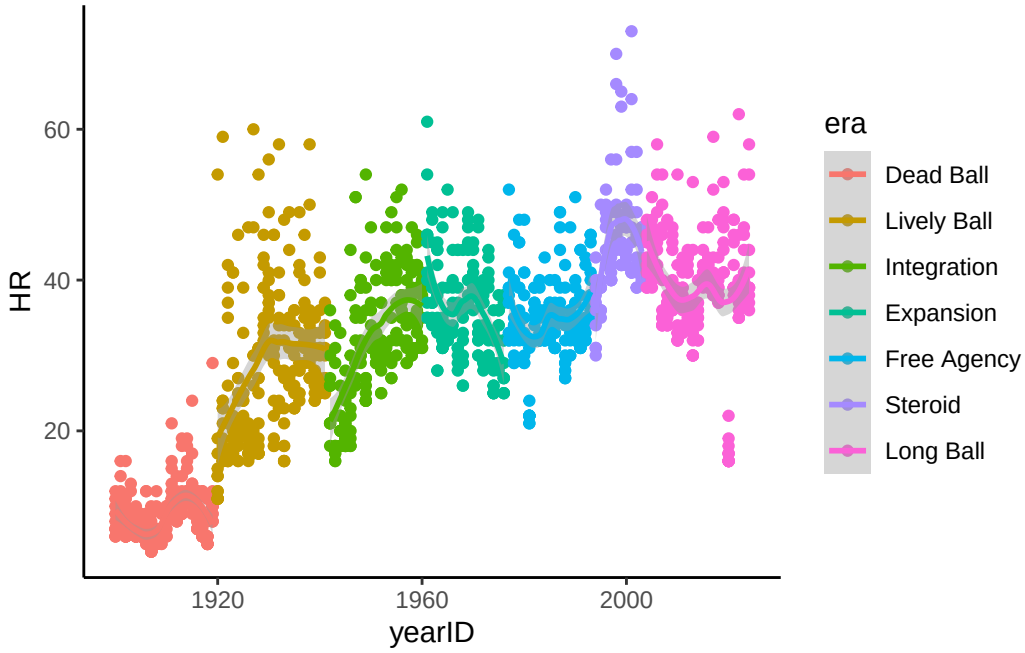
Lesson 5

```
batting_df <-
  Batting |>
  slice_max(HR,n=10,by=yearID) |>
  arrange(yearID) |>
  mutate(era = cut(yearID,
                   breaks = c(1899, 1919, 1941, 1960, 1976,
                              1993, 2003, 2050),
                   labels = c("Dead Ball", "Lively Ball", "Integration",
                              "Expansion", "Free Agency", "Steroid",
                              "Long Ball")
  )
  )
```

```
batting_df |>
  # filter(yearID>1899) |>
  filter(!is.na(era)) |>
  ggplot(aes(x=yearID, y=HR, color=era)) +
  geom_point()+
```

```
  geom_smooth()+
  theme_classic()
```

```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

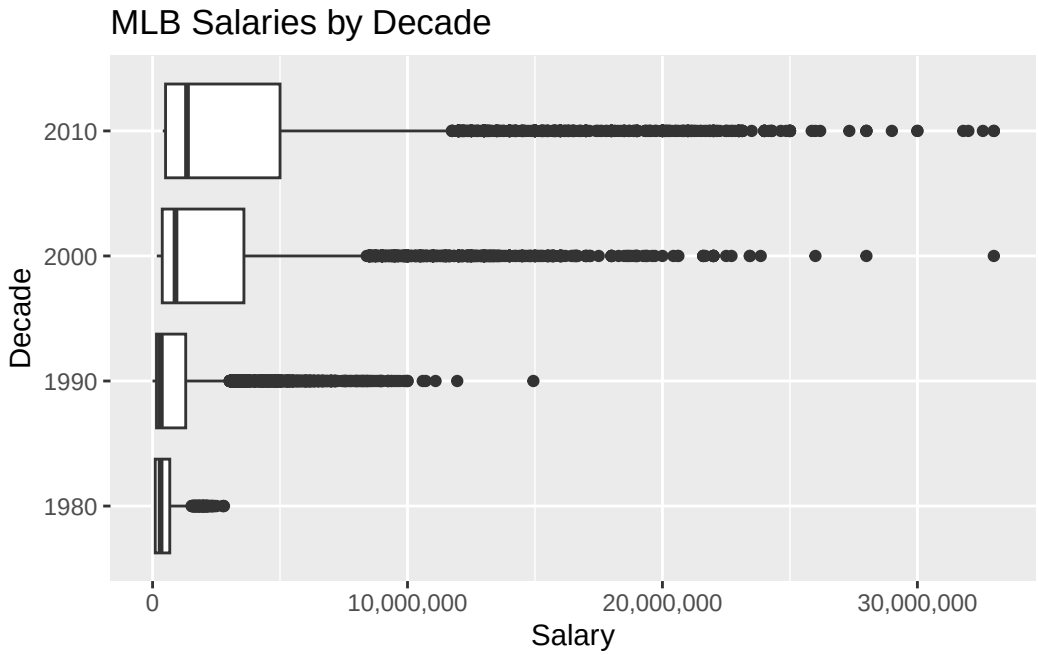



```
Salaries %>%
  mutate(decade = floor(yearID / 10) * 10) %>%
  head()
```

```
  yearID teamID lgID  playerID  salary decade
1   2004    SFN   NL aardsda01  300000   2000
2   2007    CHA   AL aardsda01  387500   2000
3   2008    BOS   AL aardsda01  403250   2000
4   2009    SEA   AL aardsda01  419000   2000
5   2010    SEA   AL aardsda01 2750000   2010
6   2011    SEA   AL aardsda01 4500000   2010
```

```
Salaries %>%
  mutate(decade = floor(yearID / 10) * 10) %>%
  # ggplot(aes(x =decade, y = salary)) +
  ggplot(aes(x = as.factor(decade), y = salary)) +
```

```
  geom_boxplot() +
  scale_y_continuous(labels = scales::comma) +
  labs(title = "MLB Salaries by Decade",
       y = "Salary",
       x = "Decade") +
  coord_flip()
```

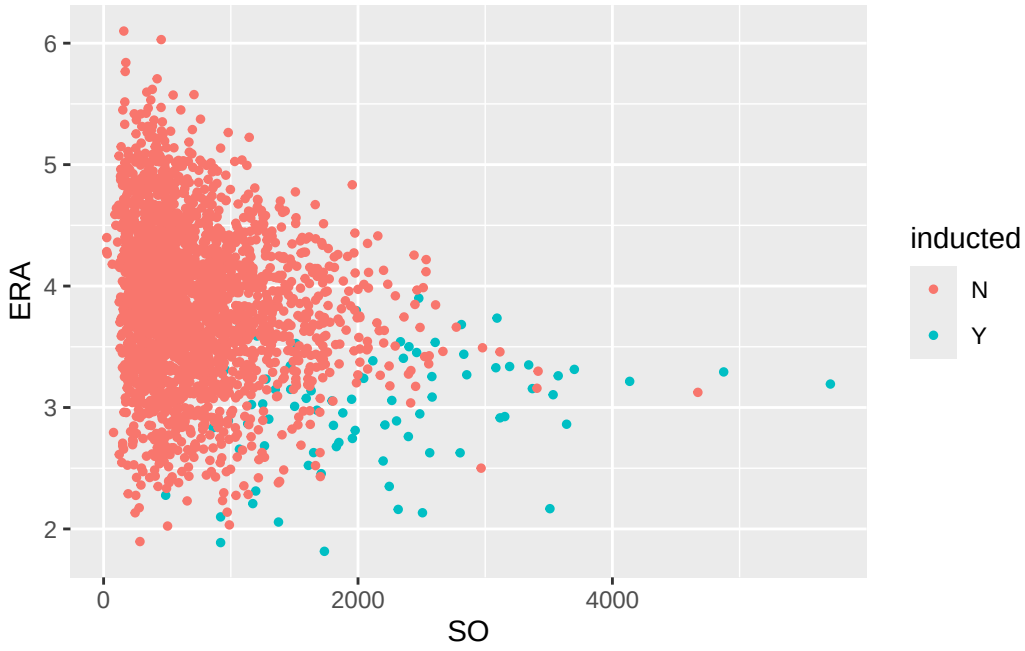



```
# Get career wins, earned run average, and strikeouts.
careers <- Pitching %>%
  group_by(playerID) %>%
  summarize(W = sum(W),
            ER = sum(ER),
            IPouts = sum(IPouts),
            SO = sum(SO)) %>%
  mutate(ERA = 9*ER/(IPouts/3)) %>%
  filter(IPouts/3 > 500)

# Add whether they were inducted in the Hall of Fame.
careers <- HallOfFame %>%
  filter(inducted == "Y",
         category == "Player") %>%
  select(playerID, inducted) %>%
```

```
  right_join(careers) %>%
  mutate(inducted = replace_na(inducted,"N"))
```

```
Joining with `by = join_by(playerID)`
```

```
careers %>%
  ggplot(aes(x = SO, y = ERA, color = inducted)) +
  geom_point(size = 1)
```



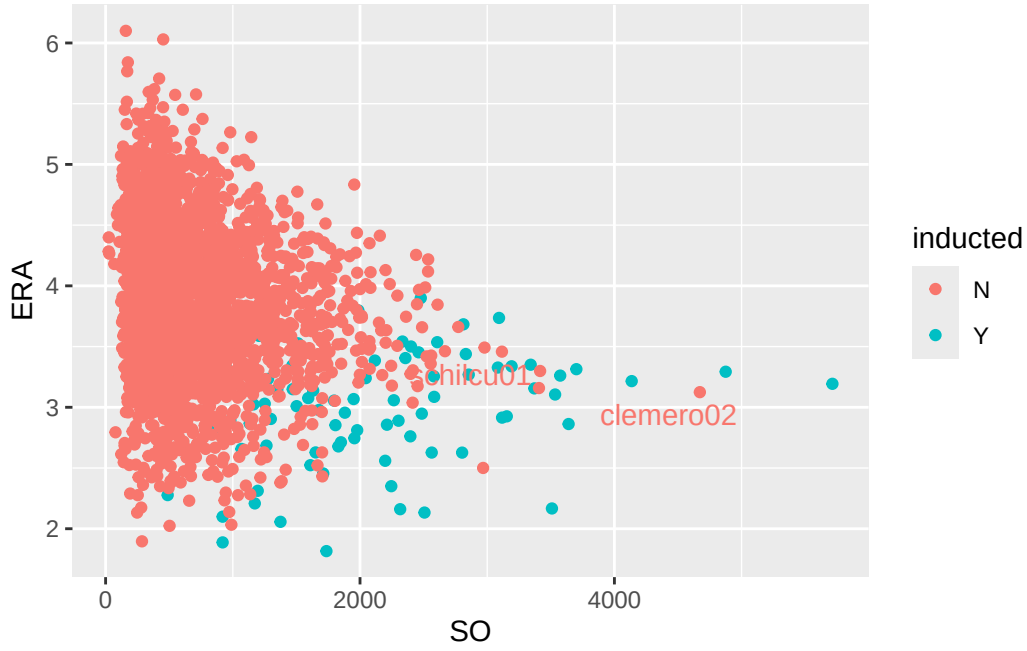

```
clemschil <- careers %>%
  filter(playerID %in% c("clemero02", "schilcu01"))
```

```
library(ggrepel)
```

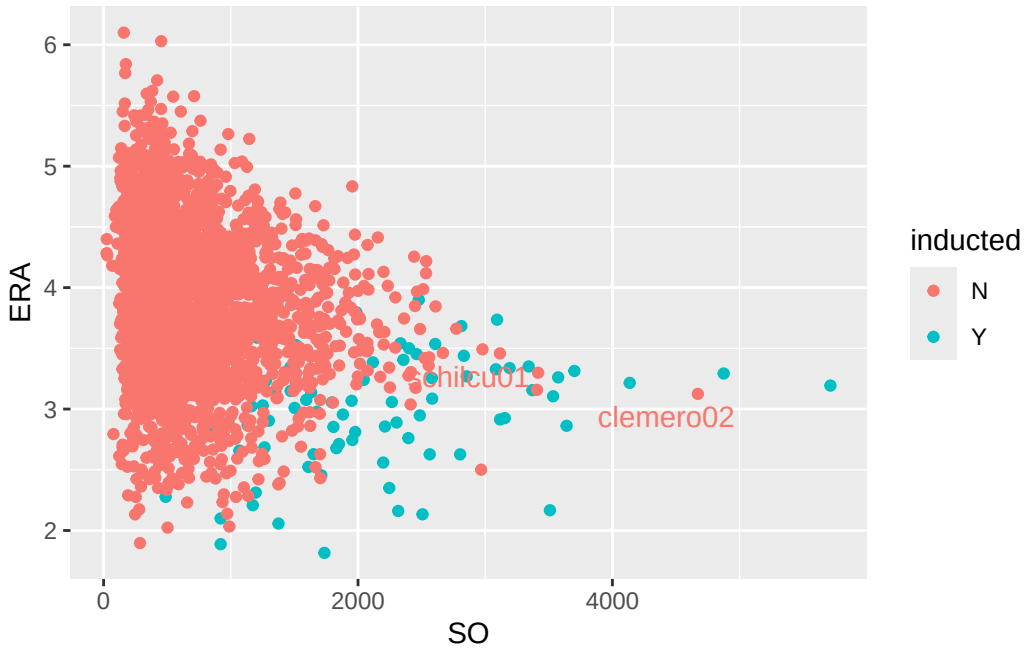
```
Warning: package 'ggrepel' was built under R version 4.5.2
```

```
clemschil <- careers %>%
  filter(playerID %in% c("clemero02", "schilcu01"))

careers %>%
```

```
  ggplot(aes(x = SO, y = ERA, color = inducted)) +
  geom_point() +
  geom_text_repel(data = clemschil,
                  aes(x = SO, y = ERA, label = playerID),
                  show.legend = FALSE)
```




```
library(ggrepel)
clemschil <- careers %>%
  filter(playerID %in% c("clemero02", "schilcu01"))

careers %>%
  ggplot(aes(x = SO, y = ERA, color = inducted)) +
  geom_point() +
  geom_text_repel(data = clemschil,
                  aes(x = SO, y = ERA, label = playerID),
                  show.legend = FALSE)
```



Lesson 8

AY26-2 Notes

Went through the way we get to k from the pythagoreon model on the board which was good.

Also did the partial derivative with respect to Runs to find the incremental number of wins per net runs scored.

Ran out of time a little at the end to get to the incremental runs model.

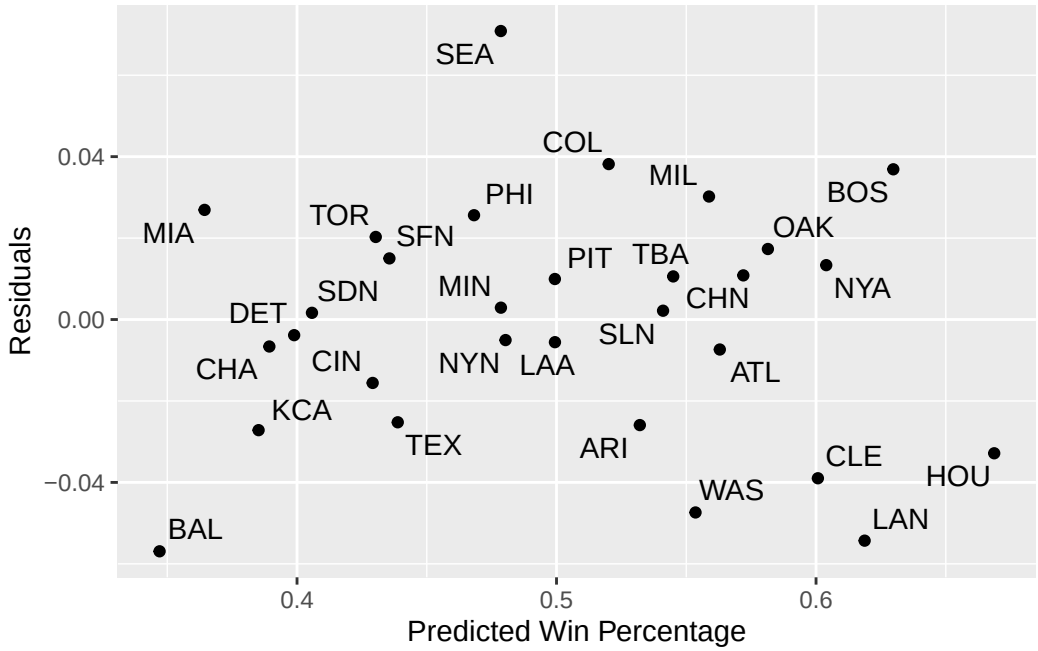
Lesson 9

AY26-2 Notes

Went well. Good to emphasize the retrosheet and statcast at the end.

```
library(Lahman)

teams2018 <- Teams |>
  filter(yearID == 2018) |>
  mutate(Wpct = W/(W+L),
         logWL = log(W/L),
```

```
         logRRA = log(R/RA))

k <- teams2018 |>
  lm(logWL ~ 0 + logRRA, data = _) |>
  tidy() |>
  pull(estimate)

teams2018 |>
  mutate(pred_Wpct = R^k/(R^k+RA^k),
         residual = Wpct - pred_Wpct) |>
  ggplot(aes(x=pred_Wpct,y=residual,label=teamID)) +
  geom_point()+
  geom_text_repel()+
  labs(x="Predicted Win Percentage",y='Residuals')
```




```
library(baseballr)

baseballr::retrosheet_data(years_to_acquire = 2018) |>
  pluck('events')
```

```
NULL
```

Lesson 10

AY26-2 Notes

Make sure the RunExpectancyMatrix.R is in the files for cadets to download. Maybe use the RunExpectancyMatrixFunctions.R file instead.

Making them re-run the data from last class was a little clunky.

Didn't have enough time to really dive into the run value by `hit_type` much. Need more on the motivation of each.

Should explain how those values aren't perfectly linear and aren't direct multiples of each other like slugging percentage.

Lesson 11

AY26-2 Notes

Talked about the RunExpectancyMatrixFunctions.R script and Problem Set 3, but still had some extra time at the end.

Add in some Regular expression from AI prompts for the end of class

```
library(baseballr)
retro2024 <-
retrosheet_data(years_to_acquire = '2024') |>
pluck('2024') |>
pluck('events')
# Add states and run values, output as retro_event_run_value
# Also gives the run expectancy matrix as erm_wide
source("./RunExpectancyMatrixFunctions.R")
get_run_expectancy_retro(retro2024)
```

```
`summarise()` has grouped output by 'bases'. You can override using the
`.groups` argument.
```