MLB Home Field Advantage

Matt Kalin

March 16, 2020

Abstract

In the Major League Baseball regular season, teams play three or four-game series against each other. The only game where either team traveled the day before is the opening game of the series, since the teams stay in town during the series. I was wondering if the number of rest days and/or the distance they had to travel would affect the rate at which the home team wins. I found that the only thing significant in predicting the winner of the opening game was the two teams' strength (I used their total season Pythagorean winning percentage as an estimate), and that travel and rest did not have a significant impact.

Step 1: Import Team Info

The first thing I did was get team information from ESPN; specifically, the team names and their url extension (so we can scrape their schedules later).

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(httr)
library(XML)
library(stringr)
teams.url = "https://www.espn.com/mlb/teams"
teams.html = rawToChar(GET(teams.url)$content)
h2.end.locs = gregexpr("/h2", teams.html)[[1]] # team headers end with an h2
mlb.teams = NA
espn.ids = NA
for (i in 1:length(h2.end.locs)) {
  html.substr = substr(teams.html, h2.end.locs[i] - 30, h2.end.locs[i] + 300)
  mlb.teams[i] = html.substr %>%
    str_extract(">.*?<") %>% # extract text between >...<
    str_replace(".{1}$", "") %>% # remove last character (<)
    str_replace("^.{1}", "") # remove first character (>)
  espn.ids[i] = html.substr %>%
    str_extract("/name/.*?/") %>% # extract text
    str_replace(".{1}$", "") %>% # remove last character (/)
    str_replace("^.{6}", "") # remove "/name/"
}
```

Step 2: Get team location info and merge with other info

I copied (and manually tidied) a table from wikipedia listing every team's location in longitude-latitude coordinates. I then converted the degrees-minutes-seconds format to degrees as a floating point value so the coordinates are compatible with the distHaversine function of the geosphere package to calculate the distance between two coordinate sets.

```
# source: "https://en.wikipedia.org/wiki/Major_League_Baseball"
library(readxl)
wiki.table = as.data.frame(read_excel("/Users/malexk999/Desktop/Cloud desktop/Miscellaneous/Coding/R/MLB/MLB Team
info.xlsx"))
master.table = data.frame("Team" = mlb.teams, "ESPN.ID" = espn.ids, "Latitude" = NA, "Longitude" = NA)
DegMinSecToCoordinates = function(deg.min.sec){
  return(deg.min.sec[1] + deg.min.sec[2] / 60 + deg.min.sec[3] / 3600)
}
StringToCoordinates = function(coord.string){
  numbers = (coord.string %>%
    str_extract_all("\\d*"))[[1]] %>%
    as.numeric() %>%
    na.omit()
  return(c(DegMinSecToCoordinates(numbers[1:3]), DegMinSecToCoordinates(numbers[4:6])))
} # first one is latitude
wiki.index = match(master.table$Team, wiki.table$Team)
for (i in 1:nrow(master.table)) {
  master.table[i, c("Latitude", "Longitude")] = StringToCoordinates(wiki.table[wiki.index[i], "Coordinates"])
}
```

Step 3: Import the standings from each year

I imported the standings from ESPN, which included runs scored and runs against. I used these to calculate each team's pythagorean winning percentage, which is an estimate of the team's true strength.

```
library(rvest)
```

```
## Loading required package: xml2
```

```
##
## Attaching package: 'rvest'
```

```
## The following object is masked from 'package:XML':
##
##     xml
```

```
minYear = 2010
maxYear = 2019
# analysis is for the decade 2010-2019
yearRange = minYear:maxYear
mlb.ratings = data.frame()
for (yr in yearRange) {
  standings.url = paste0("https://www.espn.com/mlb/standings/_/season/", yr)
  standings.data = html_table(html_nodes(read_html(standings.url), "table"))
  standings.teams = rbind(standings.data[[1]], standings.data[[3]])[,1] %>%
    as.character()
  standings.stats = rbind(standings.data[[2]], standings.data[[4]])
  # 1 and 3 are teams, 2 and 4 are rest of table
  team.ratings = data.frame("Team" = mlb.teams, "Season" = yr, "RS" = NA, "RA" = NA)
  for (i in 1:length(mlb.teams)) {
    index = grep(mlb.teams[i], standings.teams)
    if(length(index) < 1){
      index = grep("Florida Marlins", standings.teams)
      # the Miami Marlins used to be known as the Florida Marlins
    }
    team.ratings[i, 3:4] = standings.stats[index, 7:8] %>%
      as.numeric() # convert from character/string to numeric
  }
  mlb.ratings = mlb.ratings %>%
    rbind(team.ratings) # add to the master data frame
}
mlb.ratings = mlb.ratings %>%
  mutate(Pyth = RS ^ 1.83 / (RS ^ 1.83 + RA ^ 1.83)) %>% # calculate pythagorean win pct
  mutate(Log.Rate = -log(1 / Pyth - 1)) # use logit function to convert to a scale better for a linear model

# inverse: wpct = 1/(exp(b.rate - a.rate) + 1)
```

Step 4: Analyze each team's schedule and travel patterns

I scraped each team's schedule from the 2010-2019 MLB seasons. I created a data frame recording each team's series openers and how far they had to travel from the previous series and how many rest days they had.

```
# library(stringr)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(geosphere)
FirstRowAsHeader = function(df){
  for(i in 1:ncol(df)){
      names(df)[i] = as.character(df[1, i])
    }
  # names(df) = df[1, ]
  df = df[-c(1), ]
  return(df)
}
GetSiteDist = function(hosts){
  # hosts is a vector of length 2 representing the two teams' names
  coords1 = master.table[match(hosts[1], mlb.teams), c("Longitude", "Latitude")]
  coords2 = master.table[match(hosts[2], mlb.teams), c("Longitude", "Latitude")]
  return(distHaversine(coords1, coords2, r = 3959))
}
error.urls = NULL
opening.games.data = data.frame()
# pb = txtProgressBar(0, length(yearRange) * length(mlb.teams), style = 3)
for (yr in yearRange) {
  # print(yr)
  for (i in 1:length(mlb.teams)) {
    team.id = espn.ids[i]
    # print(mlb.teams[i])
    for (szn.half in 1:2) {
      error.caught = FALSE
      sched.url = paste0("https://www.espn.com/mlb/team/schedule/_/name/", team.id, "/season/", yr, "/seasontype/
2/half/", szn.half)
      sched.html = rawToChar(GET(sched.url)$content)
      tryCatch({
        sched.table = html_table(html_nodes(read_html(sched.url), "table"))[[1]] %>%
          FirstRowAsHeader()
      }, error = function(e){
        error.urls <<- c(error.urls, sched.url)
        error.caught <<- TRUE
        print(paste(mlb.teams[i], yr, "half", szn.half, "failed"))
      })
      if(error.caught){
        next()
      }
      home.strs = gregexpr(">vs<", sched.html)[[1]]
      away.strs = gregexpr(">@<", sched.html)[[1]]
      home.rows = grep("vs", sched.table$OPPONENT)
      away.rows = grep("@", sched.table$OPPONENT)
      sched.table$Perpective = mlb.teams[i]
      sched.table$Opp.Tm = NA
      sched.table$Host = NA
      for (j in 1:length(home.strs)) {
        opp.start = regexpr("mlb/team/_/name", substr(sched.html, home.strs[j], home.strs[j] + 200))
        opp.id = substr(sched.html, home.strs[j], home.strs[j] + 200) %>%
          str_match("mlb/team/_/name/[a-z]{2,3}/") %>% # find url with id
          str_replace("^.{16}", "") %>% # remove beginning of string
          str_replace(".{1}$", "") # remove last "/"
        sched.table[home.rows[j], "Opp.Tm"] = mlb.teams[match(opp.id, espn.ids)]
        sched.table[home.rows[j], "Host"] = mlb.teams[i]
      }
      for (j in 1:length(away.strs)) {
        opp.start = regexpr("mlb/team/_/name", substr(sched.html, away.strs[j], away.strs[j] + 200))
        opp.id = substr(sched.html, away.strs[j], away.strs[j] + 200) %>%
          str_match("mlb/team/_/name/[a-z]{2,3}/") %>% # find url with id
          str_replace("^.{16}", "") %>% # remove beginning of string
          str_replace(".{1}$", "") # remove last "/"
        sched.table[away.rows[j], c("Opp.Tm", "Host")] = mlb.teams[match(opp.id, espn.ids)]

      }
      {
        na.index = which(is.na(sched.table$Opp.Tm))
        if(length(na.index) > 0){
          marlins.index = na.index[grep("Florida", sched.table[na.index, "OPPONENT"])]
          sched.table[marlins.index, "Opp.Tm"] = "Miami Marlins"
          sched.table[na.index[which(is.na(sched.table[marlins.index, "Host"]))], "Host"] = "Miami Marlins"
        }
      } # florida marlins
      sched.table = sched.table %>%
        slice(grep("(W|L)\\d{1,2}-\\d{1,2}", sched.table$RESULT))
      sched.table$Game.Date = sched.table$DATE %>%
        str_replace("^.{5}", "") %>%
        paste(yr) %>%
        mdy()
      sched.table$Series.Game = NA
      sched.table$Days.Rest = NA
      for (j in 1:nrow(sched.table)) {
        if(j == 1){
          sched.table[j, "Series.Game"] = 1
        } else {
          if(sched.table[j, "Opp.Tm"] == sched.table[j - 1, "Opp.Tm"] &
             sched.table[j, "Host"] == sched.table[j - 1, "Host"]){
            sched.table[j, "Series.Game"] = sched.table[j - 1, "Series.Game"] + 1
          } else {
            sched.table[j, "Series.Game"] = 1
          }
          sched.table[j, "Days.Rest"] = sched.table[j, "Game.Date"] - sched.table[j - 1, "Game.Date"]
        }
      }
      first.games = sched.table %>%
        filter(Series.Game == 1)
      first.games$Travel.Dist = NA
      for (j in 2:nrow(first.games)) {
        first.games[j, "Travel.Dist"] = GetSiteDist(first.games[(j-1):j, "Host"])
      }
      first.games$Series.Num = 1:nrow(first.games)
      opening.games.data = opening.games.data %>%
        rbind(first.games[-1,])
    }
    # setTxtProgressBar(pb, (match(yr, yearRange) - 1) * length(mlb.teams) + i)
  }
}
# close(pb)
```

Step 5: Organize all relevant information into a table

I organized the data from step 4 into a table, combining the data of both the home and away teams. The data in this table includes the away and home teams, whether the home team won the opening game, the strength rating of each team, each team's travel distance from the previous series, each team's rest from the previous series, and how far apart the two teams' home stadiums are (to estimate the away fan base travel distance).

```
home.games = which(opening.games.data$Perpective == opening.games.data$Host)
openers.cols = c("Date", "Away.Tm", "Home.Tm", "Home.Win", "Away.Rate", "Home.Rate", "Away.Travel", "Home.Travel"
, "Away.Rest", "Home.Rest", "Dist")
openers.analysis = matrix(ncol = length(openers.cols), nrow = length(home.games)) %>%
  as.data.frame()
names(openers.analysis) = openers.cols
openers.analysis[, c("Date", "Home.Tm", "Away.Tm", "Home.Rest", "Home.Travel")] = opening.games.data[home.games,
c("Game.Date", "Host", "Opp.Tm", "Days.Rest", "Travel.Dist")]
openers.analysis$Home.Win = opening.games.data[home.games, "RESULT"] %>%
  as.character() %>%
  str_detect("^W")
for (i in 1:nrow(openers.analysis)) {
  this.row = openers.analysis[i, ]
  opp.index = which(opening.games.data$Game.Date == (this.row$Date) & opening.games.data$Host == this.row$Home.Tm
& opening.games.data$Perpective == this.row$Away.Tm)
  if(length(opp.index) != 1){
    next()
  }
  openers.analysis[i, c("Away.Travel", "Away.Rest")] = opening.games.data[opp.index, c("Travel.Dist", "Days.Rest"
)]
  game.yr = year(this.row$Date)
  openers.analysis[i, "Away.Rate"] = mlb.ratings[which(mlb.ratings$Season == game.yr & mlb.ratings$Team == this.r
ow$Away.Tm), "Log.Rate"]
  openers.analysis[i, "Home.Rate"] = mlb.ratings[which(mlb.ratings$Season == game.yr & mlb.ratings$Team == this.r
ow$Home.Tm), "Log.Rate"]
  openers.analysis[i, "Dist"] = GetSiteDist(this.row[1, c("Away.Tm", "Home.Tm")])
}
# lookup away team's travel/rest data and both teams' ratings
# throw out games where the home team was found but not the away team
```

Step 6: Analyze factors behind who wins the first game of the series

I ran several logistic regression models to estimate the impact of each of the factors listed in the table from step 5 have on determining whether the home team will win the opening game of an MLB series.

```
indep.vars = names(openers.analysis)[(match("Home.Win", names(openers.analysis)) + 1):ncol(openers.analysis)]
model.formula = paste0("Home.Win ~ ", paste(indep.vars, collapse = "+")) %>% as.formula()
all.model = glm(model.formula, data = openers.analysis, family = binomial())
summary(all.model)
```

```
##
## Call:
## glm(formula = model.formula, family = binomial(), data = openers.analysis)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7603  -1.1959   0.8759   1.0999   1.5852
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  9.534e-02  8.376e-02   1.138    0.255
## Away.Rate   -8.404e-01  8.213e-02 -10.232   <2e-16 ***
## Home.Rate    8.826e-01  8.196e-02  10.769   <2e-16 ***
## Away.Travel  3.025e-05  4.725e-05   0.640    0.522
## Home.Travel -3.196e-05  3.692e-05  -0.866    0.387
## Away.Rest   -7.557e-03  5.560e-02  -0.136    0.892
## Home.Rest    8.707e-02  5.565e-02   1.565    0.118
## Dist        -6.180e-05  3.978e-05  -1.554    0.120
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 10443  on 7563  degrees of freedom
## Residual deviance: 10192  on 7556  degrees of freedom
##   (6 observations deleted due to missingness)
## AIC: 10208
##
## Number of Fisher Scoring iterations: 4
```

```
# only the teams' ratings are statistically significant in this model

# next we will eliminate rest days from the model and instead just examine the teams' ratings, travel, and home c
ity distance
rate.travel.dist = paste0("Home.Win ~ ", paste(indep.vars[c(1:4, 7)], collapse = "+")) %>%
  as.formula() %>%
  glm(data = openers.analysis, family = binomial())
summary(rate.travel.dist)
```

```
##
## Call:
## glm(formula = ., family = binomial(), data = openers.analysis)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7327  -1.1965   0.8762   1.0993   1.6060
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.967e-01  4.652e-02   4.228 2.36e-05 ***
## Away.Rate   -8.387e-01  8.209e-02 -10.216  < 2e-16 ***
## Home.Rate    8.820e-01  8.193e-02  10.765  < 2e-16 ***
## Away.Travel  3.710e-05  4.686e-05   0.792   0.4286
## Home.Travel -2.440e-05  3.665e-05  -0.666   0.5056
## Dist        -6.556e-05  3.967e-05  -1.653   0.0984 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 10443  on 7563  degrees of freedom
## Residual deviance: 10195  on 7558  degrees of freedom
##   (6 observations deleted due to missingness)
## AIC: 10207
##
## Number of Fisher Scoring iterations: 4
```

```
# again, only the ratings are statistically significant

# I run a model taking into account ratings, rest and distance (eliminating travel)
rate.rest.dist = paste0("Home.Win ~ ", paste(indep.vars[c(1:2, 5:7)], collapse = "+")) %>%
  as.formula() %>%
  glm(data = openers.analysis, family = binomial())
summary(rate.rest.dist)
```

```
##
## Call:
## glm(formula = ., family = binomial(), data = openers.analysis)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7474  -1.1957   0.8758   1.1000   1.5624
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  9.662e-02  8.292e-02   1.165    0.244
## Away.Rate   -8.406e-01  8.212e-02 -10.237   <2e-16 ***
## Home.Rate    8.822e-01  8.195e-02  10.765   <2e-16 ***
## Away.Rest   -3.960e-03  5.538e-02  -0.072    0.943
## Home.Rest    8.315e-02  5.533e-02   1.503    0.133
## Dist        -5.053e-05  3.362e-05  -1.503    0.133
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 10443  on 7563  degrees of freedom
## Residual deviance: 10193  on 7558  degrees of freedom
##   (6 observations deleted due to missingness)
## AIC: 10205
##
## Number of Fisher Scoring iterations: 4
```

```
# as before, only the ratings are statistically significant

# I eliminate distance from the previous model and run one for ratings and rest
rate.rest = paste0("Home.Win ~ ", paste(indep.vars[c(1:2, 5:6)], collapse = "+")) %>%
  as.formula() %>%
  glm(data = openers.analysis, family = binomial())
summary(rate.rest)
```

```
##
## Call:
## glm(formula = ., family = binomial(), data = openers.analysis)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7410  -1.1967   0.8789   1.0985   1.5733
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.046836   0.075994   0.616    0.538
## Away.Rate   -0.840684   0.082099 -10.240   <2e-16 ***
## Home.Rate    0.881121   0.081941  10.753   <2e-16 ***
## Away.Rest   -0.004441   0.055371  -0.080    0.936
## Home.Rest    0.083596   0.055317   1.511    0.131
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 10443  on 7563  degrees of freedom
## Residual deviance: 10196  on 7559  degrees of freedom
##   (6 observations deleted due to missingness)
## AIC: 10206
##
## Number of Fisher Scoring iterations: 4
```

```
# the p-value for away rest is much higher than than of home rest (which is not quite significant at 0.131)

# I eliminate away rest and run a model for ratings and only the home team's rest
rate.homerest = paste0("Home.Win ~ ", paste(indep.vars[c(1:2, 6)], collapse = "+")) %>%
  as.formula() %>%
  glm(data = openers.analysis, family = binomial())
summary(rate.homerest)
```

```
##
## Call:
## glm(formula = ., family = binomial(), data = openers.analysis)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7416  -1.1966   0.8786   1.0987   1.5747
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.04400    0.06725   0.654   0.5130
## Away.Rate   -0.84064    0.08210 -10.240   <2e-16 ***
## Home.Rate    0.88122    0.08193  10.755   <2e-16 ***
## Home.Rest    0.08127    0.04712   1.725   0.0845 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 10443  on 7563  degrees of freedom
## Residual deviance: 10196  on 7560  degrees of freedom
##   (6 observations deleted due to missingness)
## AIC: 10204
##
## Number of Fisher Scoring iterations: 4
```

```
# the p-value for home rest is 0.0845, which is not quite below the 0.05 threshold for statistical significance

# I run another model to see if fan base distance is significant
rate.dist = paste0("Home.Win ~ ", paste(indep.vars[c(1:2, 7)], collapse = "+")) %>%
  as.formula() %>%
  glm(data = openers.analysis, family = binomial())
summary(rate.dist)
```

```
##
## Call:
## glm(formula = ., family = binomial(), data = openers.analysis)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7221  -1.1949   0.8767   1.0982   1.5871
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  2.028e-01  4.062e-02   4.993 5.95e-07 ***
## Away.Rate   -8.387e-01  8.208e-02 -10.219  < 2e-16 ***
## Home.Rate    8.813e-01  8.192e-02  10.758  < 2e-16 ***
## Dist        -5.071e-05  3.361e-05  -1.509    0.131
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 10443  on 7563  degrees of freedom
## Residual deviance: 10196  on 7560  degrees of freedom
##   (6 observations deleted due to missingness)
## AIC: 10204
##
## Number of Fisher Scoring iterations: 4
```

```
# again, only the ratings are significant

# here is a model with just the teams' ratings
rate.model = paste0("Home.Win ~ ", paste(indep.vars[c(1:2)], collapse = "+")) %>%
  as.formula() %>%
  glm(data = openers.analysis, family = binomial())
summary(rate.model)
```

```
##
## Call:
## glm(formula = ., family = binomial(), data = openers.analysis)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7168  -1.1965   0.8781   1.0970   1.5982
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.15280    0.02344   6.517 7.16e-11 ***
## Away.Rate   -0.83878    0.08206 -10.222  < 2e-16 ***
## Home.Rate    0.88023    0.08191  10.746  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 10443  on 7563  degrees of freedom
## Residual deviance: 10199  on 7561  degrees of freedom
##   (6 observations deleted due to missingness)
## AIC: 10205
##
## Number of Fisher Scoring iterations: 4
```

Only the two teams' ratings (and the intercept) were found to be statistically significant. We fail to reject the null hypothesis that travel, rest, and distance between teams do not contribute to determining the winner of an MLB series opener