Unsurprisingly, there are literally hundreds of different ranking systems for College Basketball and other sports, some created by large news organizations and others by passionate fans. The focus here today is to describe two popular methods, the [Rating Percentage Index (RPI)](https://en.wikipedia.org/wiki/Rating_percentage_index) and the Simple Rating System (SRS), that you can easily calculate and track yourself using functional programming in R. Then, in future posts to be released throughout the tournament season over the next few weeks, I will be able to use the rating systems described here to answer some interesting questions about tournament teams in the past and what is happening in 2018.

For this, we will need the following R packages, and a small dataset outlining an example season.

library(readr) # for reading in data

library(dplyr) # for data cleaning

library(purrr) # for functional programming

library(forcats) # for recoding factors

library(limSolve) # for solving linear equations

# Read in data example

ex\_data <- read\_csv("../data/SmallExampleResults.csv")

ex\_data

## # A tibble: 6 x 5

## WTeamID WScore LTeamID LScore WLoc

##

## 1 UConn 64 Kansas 57 H

## 2 UConn 82 Duke 68 H

## 3 UConn 72 Wisconsin 71 A

## 4 Kansas 69 UConn 62 H

## 5 Duke 81 Wisconsin 70 H

## 6 Kansas 62 Wisconsin 52 A

**Ratings Percentage Index (RPI)**

The RPI is one of the most widely known (and widely criticized) rating systems used in College Basketball. It is a weighted formula, where 25% comes from a team’s own winning percentage (WP), 50% from its opponents’ winning percentage (OWP), and 25% from its opponents’ opponents’ winning percentage (OOWP). In other words, it is trying to use a team’s win/loss record and their strength of schedule to infer their rating. The main criticisms of the RPI is that it fails to take into account margin of victory and it places too much weight on facing strong opponents (to the point where it clearly rewards teams for losing badly to many strong opponents and penalizes teams for playing well against weaker ones). That being said, it is still one of the biggest factors used by the committee for tournament selection and seeding, so it’s worth looking at and understanding.

Though calculating the RPI seems simple on the surface, there are two additional caveats that make it a bit more challenging (and a great example for functional programming).

1. An additional weighting factor (1.4 for away wins/losses, 0.6 for home wins/losses ) is included for the reference team’s WP calculation, but not for OWP and OOWP
2. For OWP and OOWP, the reference team *is not included* in the win/loss record, making both of these calculations for the reference team unique to that team. In other words (and using the example data above), Kansas as an opponent will have a different WP for when calculating the OWP for UConn than it would for Wisconsin. If you are still confused, I highly recommend checking out the example [here](https://en.wikipedia.org/wiki/Rating_percentage_index#Basketball_formula), which walks you through the calculation step-by-step.

So putting it all together, our R implementation requires the creation of three main functions, one each for WP, OWP, and OOWP. Below is the WP function, which is able to calculation a team’s winning percentage in two different ways. The first way occurs if a team should be excluded from the calculation, which will come in handy for the OWP and OOWP functions. The second is the weighted formula for home and away wins/losses, which is relevant for the WP part of the RPI formula for a given team. Here, we see that while UConn’s WP is 75% (3/4), its weighted WP is actually 81% (2.6/3.2).

# Team winning percentage

calc\_wp <- function(game\_data, team\_id, exclusion\_id = NULL){

games\_played <- game\_data[game\_data$WTeamID == team\_id | game\_data$LTeamID == team\_id, ]

if(!is.null(exclusion\_id)){

games\_played <-

games\_played[games\_played$WTeamID != exclusion\_id & games\_played$LTeamID != exclusion\_id, ]

wp <- sum(games\_played$WTeamID == team\_id)/length(games\_played$WTeamID)

} else{

wwins <- 1.4 \* sum(games\_played$WTeamID == team\_id & games\_played$WLoc == "A") +

0.6 \* sum(games\_played$WTeamID == team\_id & games\_played$WLoc == "H") +

sum(games\_played$WTeamID == team\_id & games\_played$WLoc == "N")

wlosses <- 1.4 \* sum(games\_played$LTeamID == team\_id & games\_played$WLoc == "A") +

0.6 \* sum(games\_played$LTeamID == team\_id & games\_played$WLoc == "H") +

sum(games\_played$LTeamID == team\_id & games\_played$WLoc == "N")

wp <- wwins/(wwins + wlosses)

}

return(wp)

}

calc\_wp(ex\_data, team\_id = "UConn")

## [1] 0.8125

Here we have the function to calculate the second part of the RPI formula, OWP. In order to perform the calculation, we need to calculate the winning percentage of each opponent on a given team’s schedule. This is where functional programming comes in, which allows us to apply the function calc\_wp element-wise to a vector of opponents. Here, I am using the map family of functions (specifically map\_dbl, as I am returning a double) in the purrr package, which I find to be more user-friendly than the apply family equivalents. At any rate, once I get the WP for each opponent, I simply return the mean. In this case, UConn’s OWP is 75%.

# Opponents winning percentage (exclusive of the reference team)

Library(Purrr)

calc\_owp <- function(game\_data, team\_id){

opp\_games <- game\_data[game\_data$WTeamID == team\_id | game\_data$LTeamID == team\_id, ]

opps <- if\_else(opp\_games$WTeamID == team\_id, opp\_games$LTeamID, opp\_games$WTeamID)

owp <- opps %>%

map\_dbl(~ calc\_wp(game\_data, team\_id = .x, exclusion\_id = team\_id))

return(mean(owp))

}

calc\_owp(ex\_data, team\_id = "UConn")

## [1] 0.75

Finally, the OOWP calculation is actually quite simple to compute now that the calc\_owp function has been created. It is identical in structure to that function, except instead of calling calc\_wp on the vector of opponents, we call calc\_owp on that vector. This returns the OWP for the vector of opponents, which is the OOWP. For the case of UConn, their OOWP is 51%.

# Opponents opponents winning percentage

calc\_oowp <- function(game\_data, team\_id){

opp\_games <- game\_data[game\_data$WTeamID == team\_id | game\_data$LTeamID == team\_id, ]

opps <- if\_else(opp\_games$WTeamID == team\_id, opp\_games$LTeamID, opp\_games$WTeamID)

oowp <- opps %>%

map\_dbl(~ calc\_owp(game\_data, team\_id = .x))

return(mean(oowp))

}

calc\_oowp(ex\_data, team\_id = "UConn")

## [1] 0.5138889

Putting it all together, we just need a final function that creates the linear combination of WP, OWP, and OOWP defined above. For UConn, this value is 0.7066.

# RPI using weighted formula

calc\_rpi <- function(game\_data, team\_id){

rpi <- 0.25 \* calc\_wp(game\_data, team\_id) +

0.5 \* calc\_owp(game\_data, team\_id) +

0.25 \* calc\_oowp(game\_data, team\_id)

return(round(rpi, 4))

}

calc\_rpi(ex\_data, team\_id = "UConn")

## [1] 0.7066

Again using the map\_dbl function, we can easily apply our RPI function to all teams we have in our dataset and get an idea of how teams relate to one another. In this simple example, we see that UConn and Kansas are close in rating (which makes sense given they went 1-1 against the other, with both wins for each team coming at home). There is also a clear difference between the top two teams and the bottom two.

ex\_teams <- unique(c(ex\_data$WTeamID, ex\_data$LTeamID))

data\_frame(Team = ex\_teams,

RPI = map\_dbl(ex\_teams, ~ calc\_rpi(ex\_data, team\_id = .x)))

## # A tibble: 4 x 2

## Team RPI

##

## 1 UConn 0.7066

## 2 Kansas 0.6830

## 3 Duke 0.4340

## 4 Wisconsin 0.3403

**The Simple Rating System (SRS)**

While the RPI is fairly straightforward to understand, the relative scores between teams have very little meaning. Because it does not take margin of victory into account, it is hard to say how teams would actually match up to one another, and how that match up might differ on a home, away, or neutral court. This is where a system that explicitly takes margin of victory into account comes in handy, and the SRS is one of these systems.

The main idea is that a team’s strength is comprised of its margin of victory in games and strength of schedule derived from the margin of victory of teams in other games. It also allows for the estimation of a global parameter that yields the effect of where the game was played (i.e., home vs. away). The key benefit of this system is that it is both interpretable and easy to estimate. Because of this, it is a common tool used in sports reporting, like in some of the articles by [538](https://fivethirtyeight.com/tag/simple-rating-system/).

That being said, there is no such thing as a free lunch, and the SRS’s simplicity is not without drawbacks. Namely, it ignores wins/losses, is slightly biased to offensive-oriented and faster pace teams where larger victories are more common, and it weights all games in a schedule equally.

The calculation of the SRS relies on three matrices. The first, G(ames), is an MxN+1 matrix where M is each game in a season, N are the teams that played in the season (with an additional column for home/away designation), and each element has a 1 to indicate the winner, a -1 to indicate the loser, and a 0 if the teams were not involved in that game. The additional column uses 1/-1 to indicate a game was home/away. The second, R(atings), is a N+1×1 matrix of rating scores to be estimated. The third, S(scores), is an Mx1 matrix with the margin of victory for each game. Then, we just set up the equation GR = S and solve for R using a matrix solver

With this in mind, we can quickly use functional programming again to iterate over all the teams in our dataset and use a custom function, transform\_wl, to create a series of column vectors with the relevant win/loss information to comprise our G matrix.

all\_teams <- unique(c(ex\_data$WTeamID, ex\_data$LTeamID))

# Function to create column vector of wins/losses for each team in every game

transform\_wl <- function(game\_data, team\_id){

col\_w <- if\_else(game\_data$WTeamID == team\_id, 1, 0) %>%

na\_if(0)

col\_l <- if\_else(game\_data$LTeamID == team\_id, -1, 0) %>%

na\_if(0)

col\_all <- coalesce(col\_w, col\_l) %>%

tbl\_df()

return(col\_all)

}

# Replace NAs with 0 and cbind home/away column

srs\_ex <- map(all\_teams, ~ transform\_wl(ex\_data, team\_id = .x)) %>%

bind\_cols() %>%

setNames(all\_teams) %>%

replace(is.na(.), 0) %>%

mutate(loc = fct\_recode(ex\_data$WLoc, "1" = "H", "-1" = "A", "0" = "N")) %>%

mutate(loc = as.numeric(as.character(loc))) %>%

select(loc, everything()) %>%

as.matrix()

srs\_ex

## loc UConn Kansas Duke Wisconsin

## [1,] 1 1 -1 0 0

## [2,] 1 1 0 -1 0

## [3,] -1 1 0 0 -1

## [4,] 1 -1 1 0 0

## [5,] 1 0 0 1 -1

## [6,] -1 0 1 0 -1

The matrix S is much easier to create with a few simple steps, then we can use the lsei function in the limSolve package to estimate the parameters of interest.

In interpreting the results of the SRS, everything can be related to the expected performance relative to an average team on a neutral court. This time, we see that Kansas is actually rated higher than UConn, likely due to them having a much better margin of victory against Wisconsin, the weakest team in the schedule. Additionally, there seems to be about a 7 point advantage to playing at home. The great thing about this system is that it is straightforward to predict the outcomes of future matches. For example, if Duke played Kansas at home for their next game, we would expect Duke to be 2 point underdogs (-3 – 6 + 6.9).

Library(limsolve)

scorediff\_ex <- ex\_data %>%

mutate(scorediff = WScore - LScore) %>%

select(scorediff) %>%

as.matrix()

results\_ex <- lsei(srs\_ex, scorediff\_ex)

data\_frame(Team = colnames(srs\_ex),

SRS = results\_ex[[1]])

## # A tibble: 5 x 2

## Team SRS

##

## 1 loc 6.873239

## 2 UConn 4.204225

## 3 Kansas 6.021127

## 4 Duke -3.007042

## 5 Wisconsin -7.218310

Hopefully this post provided some background into some rating systems in College Basketball and how these metrics can be easily calculated for any team (and any sport) using a few custom functions in R. In my next post, I will use these systems to look at how tournament teams have performed in the past to get some insight into what might happen in 2018.