Learning Goals

1. Using a `for()` loop.
2. Using an `if()` statement.
3. Familiarity with the code for Colley and Massey rankings.
4. Ability to adapt the code to find rankings for all NCAA I teams or teams in other sports.

`for()` loops and `if()` statements.

A `for()` loop allows us to specify that an operation should be repeated a finite number of times.

Syntax `for(`*name* `in` *vector*`){do}`

Example To create a vector showing the first ten powers of 2, we first create a vector of zeros of length 10 below, and then we replace each entry by the corresponding power of two with a `for()` loop.

```
>  Twopow<-mat.or.vec(10,1)
>  Twopow

 [1] 0 0 0 0 0 0 0 0 0 0

> for(i in 1:10){
+   Twopow[i]<-2^i
+ }
>  Twopow

 [1]    2    4    8   16   32   64  128  256  512 1024
```

An `if()` statement allows us to control which statements are executed.

Syntax `if(condition){commands to be carried out when the condition is true}`
or
`if(condition1){commands to be carried out when the condition1 is true}`
`else{commands to be carried out when the condition1 is false}`

Example To create a vector of length 10 showing 2^i if $i \leq 5$ and 3^i if $i > 5$, we first create a vector of zeros of length 10 below, and then we replace each entry by the appropriate power of two or three with a `for()` loop and an `if()` statement.

```
> Mixedpow<-mat.or.vec(10,1)
>  Mixedpow

 [1] 0 0 0 0 0 0 0 0 0 0

> for(i in 1:10){
+  if(i>5){Mixedpow[i]<-3^i}
+   else{Mixedpow[i]<-2^i}
+ }
> Mixedpow

 [1]     2     4     8    16    32   729  2187  6561 19683 59049
```

A Nested Loop To fill up a matrix, we need to work with the values of two variables, we can use a nested loop (a loop within a loop) in this case.

Example Create a 5×5 matrix with (i, j) entry equal to $i - j$. (Think about what this should look like first). We use our trick of first creating a 5×5 matrix with 0's everywhere. We let i denote the row and j denote the column below. We need to specify a value for every i and j, with both variables running from $1 - 5$. The idea is to run a loop through the j values for every row i.

```
> m<-mat.or.vec(5,5)
> m

     [,1] [,2] [,3] [,4] [,5]
[1,]    0    0    0    0    0
[2,]    0    0    0    0    0
[3,]    0    0    0    0    0
[4,]    0    0    0    0    0
[5,]    0    0    0    0    0

> for(i in 1:5){
+   for(j in 1:5){m[i,j]<-i - j}
+ }
> m

     [,1] [,2] [,3] [,4] [,5]
[1,]    0   -1   -2   -3   -4
[2,]    1    0   -1   -2   -3
[3,]    2    1    0   -1   -2
[4,]    3    2    1    0   -1
[5,]    4    3    2    1    0
```

The above tools (along with the syntax for creating conditional statements for `if()` statements) allow us to communicate with the computer and have it perform long calculations much faster than we could ourselves. Below, we will apply these tools to create the Colley and Massey matrices for the games played already in the ACC tournament.

Code for Colley's Method

Our goal here is to write a script file that will download and update our data and our rankings for us each time we run it. We will use data for the ACC tournament (You can substitute your own favorite tournament as soon as you understand the script).

Colley Rankings for the teams in the ACC

Downloading The Data First we will download the tournament data into our workspace from Massey's website: `http://masseyratings.com/`

1. Click on `Data` on the lower right of the page to expand that section. This opens a page with a list of sports by year. If you click on `Format` at the top of the table, it gives you a guide to the format of the data. The details we need are shown below:

- The data provided are intended for use with computer databases, spreadsheets, or other software programs.
- Data are provided as-is, with no guarantees of accuracy.
- Please reference MasseyRatings.com if you use any of this data in your calculations or research.
- Intra - list games if both teams are members of the selected conference
- Inter - list games that involve a team outside the selected conference
    - Matlab Games - CSV file with the following fields:
        - days since 1-1-0000
        - YYYYMMDD
        - team1 index (zero if a non-conference team)
        - homefield1 (1=home, -1=away, 0=neutral)
        - score1
        - team2 index (zero if a non-conference team)
        - homefield2 (1=home, -1=away, 0=neutral)
        - score2
    - Matlab Teams - CSV file with the team index, name.

2. From the `Data` page, click on the `2019` next to `College Basketball`.

3. At the top of the page, click on `NCAA` and at the top of the next page, click on `NCAA DI`.

4. Now click on `Atlantic Coast` at the top of the page.

5. Click on `All` at the top of the page. This shows all of the games the teams have played in both the Pre-Season and during the Season.

6. Click on `Intra` and `Scheduled` for Conference games.

7. To download the `Games` data, click on `Matlab Games` and then click on `Go`. This opens the page we want to download.

8. We create a csv file in our workspace called `Games.csv` with the commands below.
   We create a named object with the url of the page:

```
> g<-"https://www.masseyratings.com/scores.php?s=305972&sub=10423&all=1&mode=2&sch=on&format=1
```

   Then we download from that url:

```
> download.file(g,"Games.csv","curl")
```

   Note You may need to replace "curl" here by a different method for your computer. Check the options for `download.file` under `Help` in lower right panel in your R project.
   Before we import the data to R, it is best to have a look at your data with excel. Notice that at the end it lists the games that have not been played already with scores of 0 for each team. This will make a difference to our Colley matrix, so we need to erase that data. It is probably easiest to erase this data in excel before you import to R. Note also there are no headers in your csv file, so we will have to(ought to) add names to our data frame when we create it.

```
> A<-read.csv("Games.csv", header=FALSE)
> head(A)

      V1       V2 V3 V4 V5 V6 V7 V8
1 737425 20190101 14  1 81 10 -1 66
2 737427 20190103  8 -1 87  7  1 82
3 737429 20190105 14  1 77  1 -1 66
4 737429 20190105  9 -1 85 11  1 60
5 737429 20190105 12 -1 72 10  1 62
6 737429 20190105  3  1 87  2 -1 68
```

9. The teams are indexed by number in our data set `A`, so we also need to download the file containing the team indices. Go back to the previous webpage and click on `Intra`, `Scheduled` and `Matlab Teams` at the top. Then click on the `Go` button. This brings you to a page with the team indices on it.

10. We create a csv file in our workspace called `Teams.csv` and a data frame called `my_teams` in R with the commands below.

```
> t<-"https://www.masseyratings.com/scores.php?s=305972&sub=10423&all=1&mode=2&sch=on&format=2
> download.file(t,"Teams.csv","curl")
> my_teams<-read.csv("Teams.csv", header=FALSE)
> head(my_teams)

  V1             V2
1  1  Boston_College
2  2         Clemson
3  3            Duke
4  4      Florida_St
5  5    Georgia_Tech
6  6      Louisville
```

We now create vectors to name the columns of these data files(we see what is in the columns from the printout of the format information from Massey's site above) and assign names to the columns.

```
> names<-c("days", "YYYYMMDD","team1","homefield1", "score1", "team2", "homefield2", "score2")

> colnames(A)<-names
> head(A)

    days YYYYMMDD team1 homefield1 score1 team2 homefield2 score2
1 737425 20190101    14          1     81    10         -1     66
2 737427 20190103     8         -1     87     7          1     82
3 737429 20190105    14          1     77     1         -1     66
4 737429 20190105     9         -1     85    11          1     60
5 737429 20190105    12         -1     72    10          1     62
6 737429 20190105     3          1     87     2         -1     68

> namest<-c("teamindex", "teamname")
> colnames(my_teams)<-namest
> my_teams

   teamindex        teamname
1          1  Boston_College
2          2         Clemson
3          3            Duke
4          4      Florida_St
5          5    Georgia_Tech
6          6      Louisville
7          7        Miami_FL
8          8        NC_State
9          9  North_Carolina
10        10      Notre_Dame
11        11      Pittsburgh
12        12        Syracuse
13        13        Virginia
14        14   Virginia_Tech
15        15     Wake_Forest
```

Example: The first line of information above tells us that on Jan. 01 2019, team 14 (Virginia Tech) played team 10 (Notre Dame) at Virginia Tech. The score was : team 14: 81, Team 10 : 66.

Creating The Colley Matrix Now we need to create two matrices:

1. `C` will be the Colley matrix with $2+t_i$ in the (i,i) position on the diagonal where t_i is the number of games played by team i, and $-n_{i,j}$ in the (i,j) position $(i \neq j)$ where $n_{i,j}$ is the number of times that team i played team j. (This will be a $T \times T$ matrix where T is the number of teams.)

2. `B` will be an $T \times 1$ matrix with $1 + \frac{w_i - l_i}{2}$ in the ith position where w_i and l_i denote wins and losses respectively for team i.

Intermediary Matrices: We start by creating two intermediary matrices.

- `N` will be a $T \times T$ matrix where the $(i,j), i \neq j$ entry is the number of times team i has played team j.

- `WL` will be a matrix with T rows and 2 columns where the ith row has wins for team i in the first column and losses for team i in the second column.

For ACC data we have imported: Here we have T = Number of teams = 15.
N will be a 15×15 matrix where the (2, 3) entry will be the number of times Team 2(Clemson) has played Team 3(Duke). The (2,2) entry will be 0 since Clemson has not played Clemson.
`WL` will be a matrix with 15 rows and 2 columns. Row 10 will have the number of wins for Notre Dame(Team 10) in the first column and the number of losses for Notre Dame in the second.

To create the matrices `N` and `WL`, we first create a matrix of the correct size with zeros everywhere and then using a `for()` loop to change the entries to the desired ones.
Step 1 We create a T by T matrix called `N`, where T is the number teams in the tournament. We also create a $T \times 2$ matrix called `WL` with zeros everywhere. Note that T is the number of rows in the `my_teams` data file.

```
> T<-nrow(my_teams)
> N<-mat.or.vec(T,T)
> WL<-mat.or.vec(T,2)
```

Step 2 The following `for()` loop goes through the data in `A` row by row and for each row it adds a 1 to the (k, l) entry and the (l, k) entry in `N` where k is the number of the team in column 3 and l is the number of the team in column 6. Thus, it counts the games played between the teams as it goes. If the score in column 5 is bigger than the score in column 8 we add a win to the first column of `WL` for the team in column 3 and a loss to the second column of `WL` for the team in column 6. Otherwise, do the opposite.

For example for line 1 (i = 1)

```
> A[1,]

    days YYYYMMDD team1 homefield1 score1 team2 homefield2 score2
1 737425 20190101    14          1     81    10         -1     66
```

It will perform the actions:

1. `N[A[1,3],A[1,6]]<-N[A[1,3],A[1,6]]+1` and
   `N[A[1,6],A[1,3]]<-N[A[1,6],A[1,3]]+1`.
   That is, it sees that a game has been played between Team 14 and Team 10, so it will add 1 to the existing value of N[14,10] and the existing value of N[10,14].

2. It will also check the scores, if `A[1,5]>A[1,8]` (which it is) then we add 1 (for a win) to team 14's wins with `WL[A[1,3],1]<-WL[A[1,3],1]+1` and we add 1 (for a loss) to team 10's losses with `WL[A[1,6],2]<-WL[A[1,6],2]+1`. Otherwise, team 10 must have beaten team 14 and we do the opposite. We are fortunate that there are no draws in College Basketball, otherwise we would have to add a line of code to say what to do with a Draw.

```
> for(i in 1:nrow(A)){
+   N[A[i,3],A[i,6]]<-N[A[i,3],A[i,6]]+1
+   N[A[i,6],A[i,3]]<-N[A[i,6],A[i,3]]+1
+   if(A[i,5]>A[i,8]){
+     WL[A[i,3],1]<-WL[A[i,3],1]+1
+     WL[A[i,6],2]<-WL[A[i,6],2]+1
+   }
+   else{
+     WL[A[i,6],1]<-WL[A[i,6],1]+1
+     WL[A[i,3],2]<-WL[A[i,3],2]+1
+   }
+ }
```

Lets have a look at the matrices N and WL after we run the code:

```
> options(width=110)
> N
```

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15]
 [1,]    0    1    1    1    1    2    1    1    0     2     1     2     1     1     1
 [2,]    1    0    1    2    2    1    1    1    1     0     2     1     1     1     1
 [3,]    1    1    0    1    1    1    1    1    1     1     1     2     2     1     1
 [4,]    1    2    1    0    2    1    2    1    1     1     1     1     1     0     1
 [5,]    1    2    1    2    0    1    1    0    1     2     1     1     1     2     1
 [6,]    2    1    1    1    1    0    1    1    2     1     2     1     1     1     1
 [7,]    1    1    1    2    1    1    0    1    2     1     0     1     1     1     2
 [8,]    1    1    1    1    0    1    1    0    2     1     2     1     1     1     2
 [9,]    0    1    1    1    1    2    2    2    0     1     1     1     1     1     1
[10,]    2    0    1    1    2    1    1    1    1     0     0     1     2     2     1
[11,]    1    2    1    1    1    2    0    2    1     0     0     2     1     1     1
[12,]    2    1    2    1    1    1    1    1    1     1     2     0     1     1     1
[13,]    1    1    2    1    1    1    1    1    1     2     1     1     0     2     1
[14,]    1    1    1    0    2    1    1    1    1     2     1     1     2     0     1
[15,]    1    1    1    1    1    1    2    2    1     1     1     1     1     1     0
```

```
> WL
```

```
      [,1] [,2]
 [1,]    5   11
 [2,]    7    9
 [3,]   13    3
 [4,]   11    5
 [5,]    5   12
 [6,]   10    7
 [7,]    4   12
 [8,]    8    8
 [9,]   14    2
[10,]    3   13
[11,]    2   14
[12,]   10    7
[13,]   15    2
[14,]   11    5
[15,]    4   12
```

How many times did Clemson play Notre Dame?

Since the (i,j) element of N is the number of times the row team (team i) has played team j, we **can calculate the number of games that team i has played by summing row i of the matrix N.** Conveniently, there is an inbuilt function in R which sums the rows of a matrix called `rowSums()`.

```
> total<-rowSums(N)
> total
```

```
 [1] 16 16 16 16 17 17 16 16 16 16 16 17 17 16 16
```

We have created a vector called `total` which gives us the total number of games played by each team. We now have all of the pieces we need to create the Colley matric C and the matrix B in the Colley equation. We start by creating matrices of zeros of the correct dimensions.

```
> C<-mat.or.vec(T,T)
> B<-mat.or.vec(T,1)
> B

 [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

> dim(B)<-c(T,1)
> B

      [,1]
 [1,]    0
 [2,]    0
 [3,]    0
 [4,]    0
 [5,]    0
 [6,]    0
 [7,]    0
 [8,]    0
 [9,]    0
[10,]    0
[11,]    0
[12,]    0
[13,]    0
[14,]    0
[15,]    0
```

Note we used the `dim()` function to convert the row matrix B to a column matrix. We now use a for loop to fill up the matrices with the correct entries:

```
> for(i in 1:T){
+   B[i,1]<-1+(WL[i,1]-WL[i,2])/2
+   C[i,i]<-2+total[i]
+     for(j in 1:T){
+       if(j!=i){
+       C[i,j]<- -N[i,j]
+     }
+ }
+ }
```

We are now ready to solve for the colley ratings. the order of the entries in the matrix vector is the same as the order of the teams in our data frame `my_teams` so we create a new data frame showing the team name alongside their rating. We then then present them in decreasing order which shows the ranking.

```
> r<-solve(C,B)
> head(r)

          [,1]
[1,] 0.3197776
[2,] 0.4421466
[3,] 0.7842296
[4,] 0.6282029
[5,] 0.3239907
[6,] 0.5674618

> r1<-cbind(my_teams,r)
> head(r1)

  teamindex        teamname         r
1         1  Boston_College 0.3197776
2         2         Clemson 0.4421466
3         3            Duke 0.7842296
4         4      Florida_St 0.6282029
5         5    Georgia_Tech 0.3239907
6         6      Louisville 0.5674618

> r2<-r1[order(-r),]
> r2

   teamindex        teamname         r
13        13        Virginia 0.8346459
9          9  North_Carolina 0.8191359
3          3            Duke 0.7842296
14        14   Virginia_Tech 0.6469561
4          4      Florida_St 0.6282029
6          6      Louisville 0.5674618
12        12        Syracuse 0.5657165
8          8        NC_State 0.4992238
2          2         Clemson 0.4421466
5          5    Georgia_Tech 0.3239907
1          1  Boston_College 0.3197776
7          7        Miami_FL 0.3166737
```

```
15        15     Wake_Forest 0.2797841
10        10      Notre_Dame 0.2617316
11        11      Pittsburgh 0.2103233
```

```
> r3<-data.frame(r2[,2],r2[,3])
> head(r3)
```

```
          r2...2.   r2...3.
1        Virginia 0.8346459
2  North_Carolina 0.8191359
3            Duke 0.7842296
4   Virginia_Tech 0.6469561
5      Florida_St 0.6282029
6      Louisville 0.5674618
```

To show the Colley rankings(data up to Mar. 05, 2019 for 2018/2019 season) we could use

```
> rank<-1:15
> colley<-data.frame(r3[,1],rank)
> colley
```

```
           r3...1. rank
1         Virginia    1
2   North_Carolina    2
3             Duke    3
4    Virginia_Tech    4
5       Florida_St    5
6       Louisville    6
7         Syracuse    7
8         NC_State    8
9          Clemson    9
10    Georgia_Tech   10
11  Boston_College   11
12        Miami_FL   12
13     Wake_Forest   13
14      Notre_Dame   14
15      Pittsburgh   15
```

Code For Massey's Rankings

Our goal is now to set up the matrix equation for Massey's method using the data. We need to create two matrices:

1. `M` will be the Massey matrix with t_i in the (i, i) position on the diagonal where t_i is the number of games played by team i, and $-n_{i,j}$ in the (i,j) position $(i \neq j)$ where $n_{i,j}$ is the number of times that team i played team j. (This will be a $T \times T$ matrix where T is the number of teams.)

2. `P` will be an $T \times 1$ matrix with the point differential for team i in the ith position when $i < T$ and a 0 in the last position.

Intermediary Matrices: As with Colley, we create two intermediary matrices.

- `N` will be a $T \times T$ matrix where the $(i, j), i \neq j$ entry is the number of times team i has played team j. (already created for Colley's Method)

- `PD` will be a matrix with T rows and 1 column showing the point differential for all teams.

For ACC data we have imported: Here we have T = Number of teams = 15. (Note that T is already defined in R above and N has already been created, in the process of finding Colley's rankings.)
`PD` will be a matrix with 15 rows and 1 column. Row 10 will have the point differential for Notre Dame (total points for minus points against).

To create the matrix `PD`, we use our usual trick in R of creating a matrix of the correct size with zeros everywhere, and then using a for loop to change the entries to the desired ones. We create a $T \times 1$ matrix called `PD` with zeros everywhere. data file.

```
> PD<-mat.or.vec(T,1)
```

The following for loop goes through the data in `A` row by row and for each row. It will calculate the point differential for the game for both teams and add it to their current point differential;

```
> for(i in 1:nrow(A)){
+     PD[A[i,3]]<-PD[A[i,3]]+(A[i,5]-A[i,8])
+     PD[A[i,6]]<-PD[A[i,6]]+(A[i,8]-A[i,5])
+   }
```

We have a look at the matrix PD after we run the code:

```
> PD
 [1]  -89   32  163   78 -157   90  -90  -17  175 -124 -144   11  243   64 -235
```

We have already stored the number of games that each team has played thus far in a vector called `total`. We use these numbers to create the diaginal entries for the first $T-1$ rows of the Massey matrix.
We now create the matrices M and P in the equation we must solve to get the Massey ratings: $Mr = P$. We use our usual trick of first creating a matrix of 0's and then filling in the entries with a function and a loop. We create M with two loops, the first fills in the rows from 1 to T - 1 and the second fills in the last row with 1's. Note: The condition `j!=i` means j is not equal to i.

```
> M<-mat.or.vec(T,T)
> for(i in 1:(T-1)){
+   M[i,i]<-total[i]
+     for(j in 1:T){
+       if(j!=i){
+       M[i,j]<- -N[i,j]
+     }
+ }
+ }
> for(i in 1:T){
+   M[T,i]<-1
+   }
```

Now we create a matrix of 0's for P, note we need to specify the dimensions for a matrix of 0's with a single column because R treats it as a vector.

```
> P<-mat.or.vec(T,1)
> dim(P)<-c(T,1)
> for(i in 1:(T-1)){
+   P[i]<-PD[i]
+ }
```

```
> options(width=110)
> M
```

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15]
 [1,]   16   -1   -1   -1   -1   -2   -1   -1    0    -2    -1    -2    -1    -1    -1
 [2,]   -1   16   -1   -2   -2   -1   -1   -1   -1     0    -2    -1    -1    -1    -1
 [3,]   -1   -1   16   -1   -1   -1   -1   -1   -1    -1    -1    -2    -2    -1    -1
 [4,]   -1   -2   -1   16   -2   -1   -2   -1   -1    -1    -1    -1    -1     0    -1
 [5,]   -1   -2   -1   -2   17   -1   -1    0   -1    -2    -1    -1    -1    -2    -1
 [6,]   -2   -1   -1   -1   -1   17   -1   -1   -2    -1    -2    -1    -1    -1    -1
 [7,]   -1   -1   -1   -2   -1   -1   16   -1   -2    -1     0    -1    -1    -1    -2
 [8,]   -1   -1   -1   -1    0   -1   -1   16   -2    -1    -2    -1    -1    -1    -2
 [9,]    0   -1   -1   -1   -1   -2   -2   -2   16    -1    -1    -1    -1    -1    -1
[10,]   -2    0   -1   -1   -2   -1   -1   -1   -1    16     0    -1    -2    -2    -1
[11,]   -1   -2   -1   -1   -1   -2    0   -2   -1     0    16    -2    -1    -1    -1
```

```
[12,]   -2   -1   -2   -1   -1   -1   -1   -1   -1    -1    -2    17    -1    -1    -1
[13,]   -1   -1   -2   -1   -1   -1   -1   -1   -1    -2    -1    -1    17    -2    -1
[14,]   -1   -1   -1    0   -2   -1   -1   -1   -1    -2    -1    -1    -2    16    -1
[15,]    1    1    1    1    1    1    1    1    1     1     1     1     1     1     1
```

```
> P
```

```
      [,1]
 [1,]  -89
 [2,]   32
 [3,]  163
 [4,]   78
 [5,] -157
 [6,]   90
 [7,]  -90
 [8,]  -17
 [9,]  175
[10,] -124
[11,] -144
[12,]   11
[13,]  243
[14,]   64
[15,]    0
```

Now we are all set to find our Massey ratings and rankings:

```
> r<-solve(M,P)
> head(r)
```

```
          [,1]
[1,] -5.945011
[2,]  1.556715
[3,] 10.431869
[4,]  3.687725
[5,] -8.551340
[6,]  4.843315
```

```
> r1<-cbind(my_teams,r)
> head(r1)
```

```
  teamindex        teamname         r
1         1  Boston_College -5.945011
2         2         Clemson  1.556715
3         3            Duke 10.431869
4         4     Florida_St  3.687725
5         5    Georgia_Tech -8.551340
6         6      Louisville  4.843315
```

```
> r2<-r1[order(-r),]
> r2
```

```
   teamindex        teamname           r
13        13        Virginia  13.8953462
9          9  North_Carolina  10.5759161
3          3            Duke  10.4318688
6          6      Louisville   4.8433152
4          4      Florida_St   3.6877254
14        14   Virginia_Tech   3.4633690
2          2         Clemson   1.5567153
12        12        Syracuse   0.4464237
8          8        NC_State  -1.1470773
7          7        Miami_FL  -4.8506744
1          1  Boston_College  -5.9450108
10        10      Notre_Dame  -6.7790070
11        11      Pittsburgh  -7.4512319
5          5    Georgia_Tech  -8.5513400
15        15     Wake_Forest -14.1763383
```

```
> r3<-data.frame(r2[,2],r2[,3])
> head(r3)
```

```
          r2...2.   r2...3.
1        Virginia 13.895346
2  North_Carolina 10.575916
3            Duke 10.431869
4      Louisville  4.843315
5      Florida_St  3.687725
6   Virginia_Tech  3.463369
```

To show the Massey rankings(data up to Mar. 05, 2019 for 2018/2019 season) we could use

```
> rank<-1:15
> massey<-data.frame(r3[,1],rank)
> massey
```

```
           r3...1. rank
1         Virginia    1
2   North_Carolina    2
3             Duke    3
4       Louisville    4
```

```
5       Florida_St    5
6    Virginia_Tech    6
7          Clemson    7
8         Syracuse    8
9         NC_State    9
10        Miami_FL   10
11  Boston_College   11
12      Notre_Dame   12
13      Pittsburgh   13
14    Georgia_Tech   14
15     Wake_Forest   15
```