Round Pairing

ajw

Set-up

Set the round that you are pairing.

```
round <- 4
```

This runs the backing functions, which can be found here.

```
source('pair_func.R')
```

Read in data from google sheets.

```
teams <- 'https://docs.google.com/spreadsheets/d/1REb82IzLPC3S7n93CntfAaqBETdSCDXoO6DKS9VV0ro/pubhtml?g
tab   <- 'https://docs.google.com/spreadsheets/d/1REb82IzLPC3S7n93CntfAaqBETdSCDXoO6DKS9VV0ro/pubhtml?g

teams <- readSpreadsheet(teams)
df <- cleanTab(readSpreadsheet(tab))

tab <- data.frame(
  team = df$TEAM,
  side = df[[paste0('R', round - 1, '_SIDE')]],
  wpb  = as.numeric(df[[paste0('R', round - 1, '_RT_WPB')]]),
  pb   = as.numeric(df[[paste0('R', round - 1, '_RT_PB')]]),
  pd   = as.numeric(df[[paste0('R', round - 1, '_RT_PD')]]),
  stringsAsFactors = F
)

tab$rank <- rankWPB(wpb = 'wpb',
                    pb  = 'pb',
                    pd  = 'pd', dat = tab, r = round)
```

`teams` lists out past pairings, which are used to check impermissibles.

University

Team

Team Name

Team Number

UPenn

UPenn 1

A

UPenn

UPenn 2

B

Harvard 1

Harvard 1

C

Harvard 2

Harvard 2

D

Hamline

Hamline

E

Rochester

Rochester

F

`tab` shows the values on which we pair.

team

side

wpb

pb

pd

rank

A

D

38.28

7.84

10.00

6.00

B

P

44.67

8.61

26.00

4.00

C

D

45.87

8.80

-12.00

3.00

D

P

19.56

6.86

-30.00

18.00

E

D

7.90

4.05

-95.00

20.00

F

P

34.36

7.69

19.00

9.00

Defining Impermissibles

We store a list of impermissible match-ups, which are defined either as teams from the same school or teams that have previously faced one another. Here's a sample of that list.

Team1

Team2

A

B

B

A

A

K

B

L

C

M

D

N

Pairing

Now we pair the teams. If the round is side-constrained, we rank and pair by sides. Otherwise, we take all the teams togther.

```
pair <- data.frame(P_team = rep(NA, nrow(df)/2),
                   D_team = NA)

#if round is side-constrained:
if(round %in% c(2,4)){

  #rank sides separately
  needP <- tab[tab$side == 'D', ]
  needD <- tab[tab$side == 'P', ]

  #pair highest versus highest
  pair <- teamMeta(pair, needP[order(needP$rank), ], side = 'P', round = round)
  pair <- teamMeta(pair, needD[order(needD$rank), ], side = 'D', round = round)

} else {

  #rank teams together
  tab <- tab[order(tab$rank), ]

  #pair 1 vs 2, 3 vs 4, etc
  pair <- teamMeta(pair, tab[c(T, F), ], side = 'P', round = round)
  pair <- teamMeta(pair, tab[c(F, T), ], side = 'D', round = round)

}
```

Here's the pairings, which may or may not include impermissibles.

P_team

D_team

P_WPB

P_pb

P_pd

P_rank

D_WPB

D_pb

D_pd

D_rank

R

B

52.65

10.18

67.00

1
44.67
8.61
26.00
1
K
Q
48.58
9.19
26.00
2
41.16
8.48
30.00
2
C
M
45.87
8.80
-12.00
3
38.23
7.97
50.00
3
A
H
38.28
7.84
10.00
4
37.89
7.66
4.00
4
P

F
34.26
7.39
-11.00
5
34.36
7.69
19.00
5
L
G
29.19
8.04
-5.00
6
30.33
7.92
2.00
6
S
N
23.81
7.50
9.00
7
30.07
7.43
12.00
7
I
O
23.30
6.05
-38.00
8
23.86

7.26

25.00

8

J

D

21.11

6.40

-27.00

9

19.56

6.86

-30.00

9

E

T

7.90

4.05

-95.00

10

11.38

4.66

-62.00

10

Finding and Resolving Impermissibles

We loop through each row and see whether the matchup is in the list of impermissibles we created earlier.

```
# Find impermissibles
pair$impermiss <- findImpermiss(pair, impermiss)

# Set value to store swaps
swaps <- data.frame(Team1 = NA, Team2 = NA, final = NA)
```

And then we actually try to resolve them.

```
# If there are no impermissibles,
if(sum(pair$impermiss) == 0){

  writeLines('No impermissibles!')
```

```
} else {

  # Create value to store ranks after moving teams around
  pair$newP_rank <- pair$P_rank
  pair$newD_rank <- pair$D_rank

  # resolve impermissibles
  while (sum(pair$impermiss) > 0){

    # Print pairings at start
    writeLines('Current List of Pairings')
    printTab(xtable(addColor(pair)))

    # set trial_x = highest trial with impermissible
    trial_x <- pair[min(which(pair$impermiss == T)), ]

    # Compare swap distances based on WPB, PB, and PD
    possSwaps <- compareDist(all = tab, x = trial_x, pair = pair, round = round)

    repeat{

      # Set proposed_swap = minimum distance swap
      proposedSwap <- possSwaps[1, ]

      # proposed Swap
      writeLines('Proposed Swap')
      printTab(xtable(proposedSwap))

      # If it's allowed
      if(!paste0(proposedSwap$p, proposedSwap$d) %in% swaps$final){
        break # Move on
      }

      # If it's not allowed, remove proposed_swap from possible
      writeLines('Proposed swap is not possible!')
      possSwaps <- possSwaps[-1, ]
    }

    #  make proposed_swap
    pair <- makeSwap(newSwap = proposedSwap, old = trial_x, dat = pair)

    #  insert proposed_swap in SWAP
    swaps <- insertSwap(neW = proposedSwap, dat = swaps)

    #  set n = number of impermissibles
    pair$impermiss <- findImpermiss(pair, impermiss)
    }
  }
```

Current List of Pairings

P_team

D_team

P_WPB

P_pb

P_pd

P_rank

D_WPB

D_pb

D_pd

D_rank

impermiss

newP_rank

newD_rank

R

B

52.645

10.179

67

1

44.67

8.612

26

1

TRUE

1

1

K

Q

48.576

9.193

26

2

41.161

8.483

30

2

FALSE

2

2
C
M
45.87
8.796
-12
3
38.235
7.969
50
3
TRUE
3
3
A
H
38.277
7.838
10
4
37.889
7.656
4
4
TRUE
4
4
P
F
34.257
7.388
-11
5
34.364
7.688
19

5
TRUE
5
5
L
G
29.189
8.04
-5
6
30.331
7.925
2
6
FALSE
6
6
S
N
23.807
7.505
9
7
30.069
7.433
12
7
FALSE
7
7
I
O
23.301
6.055
-38
8

23.863

7.263

25

8

FALSE

8

8

J

D

21.115

6.403

-27

9

19.563

6.862

-30

9

FALSE

9

9

E

T

7.904

4.055

-95

10

11.379

4.655

-62

10

FALSE

10

10

Proposed Swap

p

d

dist_WPB

dist_PB

dist_PD

dist_rank

cat

R

Q

3.51

0.13

4.00

1

Keep P, Swap D

Current List of Pairings

P_team

D_team

P_WPB

P_pb

P_pd

P_rank

D_WPB

D_pb

D_pd

D_rank

impermiss

newP_rank

newD_rank

R

Q

52.645

10.179

67

1

41.161

8.483

30

2

FALSE

1

1

K

B

48.576

9.193

26

2

44.67

8.612

26

1

FALSE

2

2

C

M

45.87

8.796

-12

3

38.235

7.969

50

3

TRUE

3

3

A

H

38.277

7.838

10

4

37.889

7.656
4
4
TRUE
4
4
P
F
34.257
7.388
-11
5
34.364
7.688
19
5
TRUE
5
5
L
G
29.189
8.04
-5
6
30.331
7.925
2
6
FALSE
6
6
S
N
23.807
7.505

9
7
30.069
7.433
12
7
FALSE
7
7
I
O
23.301
6.055
-38
8
23.863
7.263
25
8
FALSE
8
8
J
D
21.115
6.403
-27
9
19.563
6.862
-30
9
FALSE
9
9
E

T

7.904

4.055

-95

10

11.379

4.655

-62

10

FALSE

10

10

Proposed Swap

p

d

dist_WPB

dist_PB

dist_PD

dist_rank

cat

C

H

0.35

0.31

46.00

1

Keep P, Swap D

Current List of Pairings

P_team

D_team

P_WPB

P_pb

P_pd

P_rank

D_WPB

D_pb

D_pd

D_rank

impermiss

newP_rank

newD_rank

R

Q

52.645

10.179

67

1

41.161

8.483

30

2

FALSE

1

1

K

B

48.576

9.193

26

2

44.67

8.612

26

1

FALSE

2

2

C

H

45.87

8.796

-12

3
37.889
7.656
4
4
FALSE
3
3
A
M
38.277
7.838
10
4
38.235
7.969
50
3
FALSE
4
4
P
F
34.257
7.388
-11
5
34.364
7.688
19
5
TRUE
5
5
L
G

29.189
8.04
-5
6
30.331
7.925
2
6
FALSE
6
6
S
N
23.807
7.505
9
7
30.069
7.433
12
7
FALSE
7
7
I
O
23.301
6.055
-38
8
23.863
7.263
25
8
FALSE
8

8

J

D

21.115

6.403

-27

9

19.563

6.862

-30

9

FALSE

9

9

E

T

7.904

4.055

-95

10

11.379

4.655

-62

10

FALSE

10

10

Proposed Swap

p

d

dist_WPB

dist_PB

dist_PD

dist_rank

cat

P

M

3.87

0.28

31.00

1

Keep P, Swap D

Final Pairings

Here are final pairings, which are resolved for impermissibles.

P_team

D_team

P_WPB

P_pb

P_pd

P_rank

D_WPB

D_pb

D_pd

D_rank

impermiss

newP_rank

newD_rank

R

Q

52.645

10.179

67

1

41.161

8.483

30

2

FALSE

1

1

K

B
48.576
9.193
26
2
44.67
8.612
26
1
FALSE
2
2
C
H
45.87
8.796
-12
3
37.889
7.656
4
4
FALSE
3
3
A
F
38.277
7.838
10
4
34.364
7.688
19
5
FALSE

4
4
P
M
34.257
7.388
-11
5
38.235
7.969
50
3
FALSE
5
5
L
G
29.189
8.04
-5
6
30.331
7.925
2
6
FALSE
6
6
S
N
23.807
7.505
9
7
30.069
7.433

12
7
FALSE
7
7
I
O
23.301
6.055
-38
8
23.863
7.263
25
8
FALSE
8
8
J
D
21.115
6.403
-27
9
19.563
6.862
-30
9
FALSE
9
9
E
T
7.904
4.055
-95

10

11.379

4.655

-62

10

FALSE

10

10