DATA607: Project1

Neil Shah

2/13/2020

#DATA607: Project 1

Introduction:

This project will demonstrate the ues of R in reading in the parsing of a text-file, which is then lodaded into a dataframe. This is a very common task in data munging.

I collaborated with Jeff Shamp;while our final codes might be different we bounced ideas off each other, and he was very helpful in developing a strategy.

Initilaizing

Loading a helpful package

```
library('stringr')
```

The text file was saved locally on my computer.

But it can also be found here
'https://raw.githubusercontent.com/shahneilp/DATA607/master/Project%201/tournamentinfo.txt
(https://raw.githubusercontent.com/shahneilp/DATA607/master/Project%201/tournamentinfo.txt)'

```
filepath <- "tournamentinfo.txt"

data <- readLines('https://raw.githubusercontent.com/shahneilp/DATA607/master/Project%201/tourna
mentinfo.txt')
```

Note–I chose to read the file directly into memory; normally this would be the standard due if this was a HUGE file but it isn't and I reasoned that most traditional computers have the RAM to handle it. For big data files I would do it line by line.

Initial Analysis

Opening the text file I see there are two types of rows.

```
data(head)

-----------------------------------------------------------------------------------------
 Pair | Player Name                     |Total|Round|Round|Round|Round|Round|Round|Round|
 Num  | USCF ID / Rtg (Pre->Post)       | Pts |  1  |  2  |  3  |  4  |  5  |  6  |  7  |
-----------------------------------------------------------------------------------------
    1 | GARY HUA                        |6.0  |W  39|W  21|W  18|W  14|W   7|D  12|D   4|
   ON | 15445895 / R: 1794   ->1817     |N:2  |W    |B    |W    |B    |W    |B    |W    |
-----------------------------------------------------------------------------------------
    2 | DAKSHESH DARURI                 |6.0  |W  63|W  58|L   4|W  17|W  16|W  20|W   7|
   MI | 14598900 / R: 1553   ->1663     |N:2  |B    |W    |B    |W    |B    |W    |B    |
-----------------------------------------------------------------------------------------
```

1. Data containing words with normal white space and punctuation that needs to be extracted.
2. Hyphen separating rows that I can safely eliminate.
3. The pertinent data starts on line 5
4. Player data is every 3rd line

Furthermore:

1. ID are stored as an 8 digit number
2. Total pts are a two digit float
3. States are two capital letter preceded/followed by a white space
4. Names are all capital letters with whitespace–they are precded by numbers.
5. Pre-scores can be a 3-4 digit number and includea P–they come after a R: and before a ->
6. Each row is 89 characters
7. The player schedule is the opponents rank but can also include games not played.

I will exploit these formatting to create reg-ex to pull said values and create vectors.

Initial RegEx

1. IDs are 8 digit numbers without any white space–this expression in regex is the following:

```
idregex <- "[0-9]{8}"
```

Matching exactly an 8 digit number

2. Total pts are two digit number with floats (decimal) place

```
pointsregex <- "\\d\\.\\d"
```

3. Matching the states–capital letters preceded by a white space on the start of a string with exactly 2 characters.

```
stateregex <- "^\\s+[A-Z]{2}"
```

4. Names are capital strings, at least two characters with white space and preceeded by a number and bar |.

```
nameregex <- '[0-9] \\| ([A-Z]+\\s){2,}'
```

5. Scores are 3-4 digit number with possible P string–but are preceeded by a R:

```
scoreregex <- "R:\\s*([0-9P]+){1,}"
```

Creating Initial vectors

Using the regex expressions I can make column vectors with our data.

```
states <- unlist(str_extract_all(unlist(data), stateregex))
names <- unlist(str_extract_all(unlist(data), nameregex))
points <- unlist(str_extract_all(unlist(data), pointsregex))
ids <- unlist(str_extract_all(unlist(data), idregex))
scores <-unlist(str_extract_all(unlist(data), scoreregex))
```

Post Processing

Let’s just check our columns

```
> head(names,2)
[1] "1 | GARY HUA "        "2 | DAKSHESH DARURI "
> head(states,2)
[1] "   ON" "   MI"
> head(points,2)
[1] "6.0" "6.0"
> head(scores,2)
[1] "R: 1794" "R: 1553"
> head(ids,2)
[1] "15445895" "14598900
```

Let’s remove the whitespace from states, the RL from scores and the number | from names.

For scores–we can just remove the R: portion and keep the number before the P–this is usually a 3,4 digit number. For states we just need to match exactly two capital letters For names –we can just pull the names with spaces and match at least two cases.

Also let’s convert the numerical columns to such.

```
> scores <-unlist(str_extract_all(unlist(scores), '[0-9]{3,4}'))
> head(scores,2)
[1] "1794" "1553"

> states <- unlist(str_extract_all(unlist(states), '[A-Z]{2}'))
> head(states,2)
[1] "ON" "MI"

> names <- unlist(str_extract_all(unlist(names), "([A-Z]+\\s){2,}"))
> head(names,2)
[1] "GARY HUA "        "DAKSHESH DARURI "


points <-as.numeric(points)
scores <-as.numeric(scores)
```

Data Validation

If we pulled this data correctly they should all have the same entries/lenght

```
> identical(length(states),length(names),length(points),length(scores))
[1] TRUE
```

Assembling the data frame.

Note: I included ID even though project isn’t asking it

```
chessdf <- data.frame(names,states,ids,points,scores)
> head(chessdf)
                 names states      ids points scores
1            GARY HUA      ON 15445895    6.0   1794
2     DAKSHESH DARURI      MI 14598900    6.0   1553
```

Opponent rankings

The opponent played is a series of up to 7 entries, delineated by the rank of an opponent and can include empty values after a D/H. This is tricky since pulling numbers before the | could miss games played. Since we know there is a possibly (max) of 7 games played and 64 players–our end result should be a 64*7 matrix that can include blank/0 values.

Going through the data–the values of the game out come can be W,L,D,H,U,X,B. Some of these have numbers (meaning a game was played) and some are blank [B,U,D,H,X]. To complicate things the line after each row also contains W,B with blanks.

I noticed that the only data I needed was from each player row so I sliced the data and did an initial regex to find everything between |

```
dataslice <- data[seq(5,length(data),3)]
```

For simplicity I’m going to make matrix/vector called opp to do the calculations.

First to regex anything after a |WDBHXLU

```
opp <- unlist(str_extract_all(unlist(dataslice), "\\|[WDBDXHLU]\\s\\s(.*)"))
```

Replace values of U,H,X,B with 0 to make my life easier–these imply a game not played.

```
opp<-str_replace_all(opp,'H','0')
opp<-str_replace_all(opp,'U','0')
opp<-str_replace_all(opp,'B','0')
opp<-str_replace_all(opp,'X','0')
```

And finally pulling just the numerical values.

```
opp<- unlist(str_extract_all(unlist(opp), "[0-9]{1,2}"))
> head(opp)
[1] "39" "21" "18" "14" "7"  "12"
> length(opp)
[1] 448

opp <- as.numeric(opp)
```

We now have 64*7 or 448 entries.

```
> opp
  [1] 39 21 18 14  7 12  4 63 58  4 17 16 20  7  8 61 25 21 11 13 12 23 28  2 26  5 19  1 45 37
12 13  4 14 17 34 29
 [38] 11 35 10 27 21 57 46 13 11  1  9  2  3 32 14  9 47 28 19 25 18 59  8 26  7 20 16 19 55 31
6 25 18 38 56  6  7
 [75]  3 34 26 42 33  5 38  0  1  3 36 27  7  5 33  3 32 54 44  8  1 27  5 31 19 16 30 22 54 33
38 10 15  0 39  2 36
[112]  0 48 41 26  2 23 22  5 47  9  1 32 19 38 10 15 10 52 28 18  4  8 40 49 23 41 28  2  9 43
1 47  3 40 39  6 64
[149] 52 28 15  0 17 40  4 43 20 58 17 37 46 28 47 43 25 60 44 39  9 53  3 24 34 10 47 49 40 17
4  9 32 11 51 13 46
[186] 37 14  6  0 24  4 22 19 20  8 36 50  6 38 34 52 48  0 52 64 15 55 31 61 50 58 55 64 10 30
50 14 61  8 44 18 51
[223] 26 13 60 12 50 36 13 15 51  6 60 37 29 25 11 52 46 38 56  6 57 52 48 13 57 51 33  0 16 28
0  5 34 27  0 23 61
[260] 11 35 29 12  0 18 15  1 54 40 16 44 21 24 20 26 39 59 21 56 22 59 17 58 20  0  0  0 12 50
57 60 61 64 56 21 23
[297] 24 63 59 46 55  0 14 32 53 39 24 59  5 51 60 56 63 55 58 35  7 27 50 64 43 23 18 24 21 61
8 51 25 17 63  0 52
[334]  0 29 35 26 20 63 64 58  0  0 29 42 33 46  0 31 30 27 45 36 57 32 47 33 30 22 19 48 29 35
34  0 25  0 44  0 57
[371]  0 14 39 61  0 15 59 64 62 31 10 30  0 45 43  0 11 35 45  0 40 42  7 36 42 51 35 53  0 31
2 41 23 49  0 45 41
[408]  0  9 40 43 54 44 33 34 45 42 24  0  0 32  3 54 47 42 30 37 55  0  0  0  0  0  0  2 48 49
43 45  0  0 22 30 31
[445] 49 46 42 54
```

Here we got vector that shows the ranking (player) that each player competed against; 0’s imply no opponent and thus not a game played.

Looping over this vector and since chessdf has player scores stored as an index–and keeping mind non-zero indexing.

```
for (r in 1:448) {if (opp[r]==0){opp[r] <-0} else{opp[r]<-as.vector(chessdf$scores)[opp[r]]}}
```

Organizing this into a dataframe

```
opp <-matrix(opp, ncol=7, byrow=TRUE)
opp <-as.data.frame(opp)
> head(opp)
    V1   V2   V3   V4   V5   V6   V7
1 1436 1563 1600 1610 1649 1663 1716
2 1175  917 1716 1629 1604 1595 1649
3 1641  955 1745 1563 1712 1666 1663
4 1363 1507 1553 1579 1655 1564 1794
5 1242  980 1663 1666 1716 1610 1629
6 1399 1602 1712 1438 1365 1552 1563
```

We now have a dataframe with the scores of opponents–non zero values convienantly implies a game was played.

Summing non zero-values

```
> opp <-cbind(opp, Count = rowSums(!opp))
> opp
     V1   V2   V3   V4   V5   V6   V7 Count
1  1436 1563 1600 1610 1649 1663 1716     0
2  1175  917 1716 1629 1604 1595 1649     0
3  1641  955 1745 1563 1712 1666 1663     0
4  1363 1507 1553 1579 1655 1564 1794     0
5  1242  980 1663 1666 1716 1610 1629     0
```

Now simply summing the first 7 elements of the row divided by 7-count, would give us the post rating. I will floor it to keep it as an integer.

We can add this to our chessdf

```
chessdf$postscore <-floor((rowSums(opp[1:7])/(7-opp$Count)))
```

The Full Dataset

```
> chessdf
                      names states      ids points scores postscore
1                  GARY HUA     ON 15445895    6.0   1794      1605
2           DAKSHESH DARURI     MI 14598900    6.0   1553      1469
3              ADITYA BAJAJ     MI 14959604    6.0   1384      1563
4       PATRICK H SCHILLING     MI 12616049    5.5   1716      1573
5                HANSHI ZUO     MI 14601533    5.5   1655      1500
6               HANSEN SONG     OH 15055204    5.0   1686      1518
7         GARY DEE SWATHELL     MI 11146376    5.0   1649      1372
8          EZEKIEL HOUGHTON     MI 15142253    5.0   1641      1468
9               STEFANO LEE     ON 14954524    5.0   1411      1523
10                ANVIT RAO     MI 14150362    5.0   1365      1554
11 CAMERON WILLIAM MC LEMAN     MI 12581589    4.5   1712      1467
12           KENNETH J TACK     MI 12681257    4.5   1663      1506
13        TORRANCE HENRY JR     MI 15082995    4.5   1666      1497
14             BRADLEY SHAW     MI 10131499    4.5   1610      1515
15   ZACHARY JAMES HOUGHTON     MI 15619130    4.5   1220      1483
16             MIKE NIKITIN     MI 10295068    4.0   1604      1385
17       RONALD GRZEGORCZYK     MI 10297702    4.0   1629      1498
18            DAVID SUNDEEN     MI 11342094    4.0   1600      1480
19             DIPANKAR ROY     MI 14862333    4.0   1564      1426
20              JASON ZHENG     MI 14529060    4.0   1595      1410
21            DINH DANG BUI     ON 15495066    4.0   1563      1470
22         EUGENE L MCCLURE     MI 12405534    4.0   1555      1300
23                 ALAN BUI     ON 15030142    4.0   1363      1213
24        MICHAEL R ALDRICH     MI 13469010    4.0   1229      1357
25         LOREN SCHWIEBERT     MI 12486656    3.5   1745      1363
26                  MAX ZHU     ON 15131520    3.5   1579      1506
27           GAURAV GIDWANI     MI 14476567    3.5   1552      1221
28              SOFIA ADINA     MI 14882954    3.5   1507      1522
29         CHIEDOZIE OKORIE     MI 15323285    3.5   1602      1313
30       GEORGE AVERY JONES     ON 12577178    3.5   1522      1144
31             RISHI SHETTY     MI 15131618    3.5   1494      1259
32    JOSHUA PHILIP MATHEWS     ON 14073750    3.5   1441      1378
33                  JADE GE     MI 14691842    3.5   1449      1276
34   MICHAEL JEFFERY THOMAS     MI 15051807    3.5   1399      1375
35         JOSHUA DAVID LEE     MI 14601397    3.5   1438      1149
36            SIDDHARTH JHA     MI 14773163    3.5   1355      1388
37     AMIYATOSH PWNANANDAM     MI 15489571    3.5    980      1384
38                BRIAN LIU     MI 15108523    3.0   1423      1539
39            JOEL R HENDON     MI 12923035    3.0   1436      1429
40             FOREST ZHANG     MI 14892710    3.0   1348      1390
41      KYLE WILLIAM MURPHY     MI 15761443    3.0   1403      1248
42                 JARED GE     MI 14462326    3.0   1332      1149
43        ROBERT GLEN VASEY     MI 14101068    3.0   1283      1106
44       JUSTIN D SCHILLING     MI 15323504    3.0   1199      1327
45                DEREK YAN     MI 15372807    3.0   1242      1152
46 JACOB ALEXANDER LAVALLEY     MI 15490981    3.0    377      1357
47              ERIC WRIGHT     MI 12533115    2.5   1362      1392
48             DANIEL KHAIN     MI 14369165    2.5   1382      1355
49         MICHAEL J MARTIN     MI 12531685    2.5   1291      1285
50               SHIVAM JHA     MI 14773178    2.5   1056      1296
51           TEJAS AYYAGARI     MI 15205474    2.5   1011      1356
```

```
52                ETHAN GUO      MI 14918803    2.5    935      1494
53            JOSE C YBARRA      MI 12578849    2.0   1393      1345
54              LARRY HODGE      MI 12836773    2.0   1270      1206
55                ALEX KONG      MI 15412571    2.0   1186      1406
56             MARISA RICCI      MI 14679887    2.0   1153      1414
57               MICHAEL LU      MI 15113330    2.0   1092      1363
58             VIRAJ MOHILE      MI 14700365    2.0    917      1391
59        SEAN M MC CORMICK      MI 12841036    2.0    853      1319
60               JULIA SHEN      MI 14579262    1.5    967      1330
61            JEZZEL FARKAS      ON 15771592    1.5    955      1327
62            ASHWIN BALAJI      MI 15219542    1.0   1530      1186
63     THOMAS JOSEPH HOSMER      MI 15057092    1.0   1175      1350
64                   BEN LI      MI 15006561    1.0   1163      1263
```

Writing

We can simply now write this dataframe out

```
write.table(chessdf, file = "chessratings.csv", sep = ",", row.names = FALSE)
```

Conclusion

In this Project I sucessfully took a standard text file and pulled the relevant data into a datafrme and outputted a csv. This, I'm sure is a very common task for data scientists–data munging/process is 80% of the job. While the regex was annoying at first–it's really a powerful tool.

Possible improvements
1. Reading line by line–this would be an optimized solution that would help preventing memory issues for big files.
2. Exploiting the fixed lengths to quickly make dataframes–most of the data was at the same interval, I chose to use regex instead to make this more scalable.
3) Better regex–ideally I wouldn't need to post procses my data pulls; I'm sure there are more efficient ways to regex.
4) Combining code–the use of intermediary datafames was to make my life easier but I could of combined this all in one

Appendix: Full code

```
data <- readLines('https://raw.githubusercontent.com/shahneilp/DATA607/master/Project%201/tourna
mentinfo.txt')
idregex <- "[0-9]{8}"
pointsregex <- "\\d\\.\\d"
stateregex <- "^\\s+[A-Z]{2}"
nameregex <- '[0-9] \\| ([A-Z]+\\s){2,}'
scoreregex <- "R:\\s*([0-9P]+){1,}"
states <- unlist(str_extract_all(unlist(data), stateregex))
names <- unlist(str_extract_all(unlist(data), nameregex))
points <- unlist(str_extract_all(unlist(data), pointsregex))
ids <- unlist(str_extract_all(unlist(data), idregex))
scores <-unlist(str_extract_all(unlist(data), scoreregex))
scores <-unlist(str_extract_all(unlist(scores), '[0-9]{3,4}'))
states <- unlist(str_extract_all(unlist(states), '[A-Z]{2}'))
names <- unlist(str_extract_all(unlist(names), "([A-Z]+\\s){2,}"))
points <-as.numeric(points)
scores <-as.numeric(scores)
print('Data Validation:')
print(identical(length(states),length(names),length(points),length(scores)))
rank <-seq(1,length(names))
chessdf <- data.frame(names,states,ids,points,scores)
dataslice <- data[seq(5,length(data),3)]
opp <- unlist(str_extract_all(unlist(dataslice), "\\|[WDBDXHLU]\\s\\s(.*)"))
opp<-str_replace_all(opp,'H','0')
opp<-str_replace_all(opp,'U','0')
opp<-str_replace_all(opp,'B','0')
opp<-str_replace_all(opp,'X','0')
opp<- unlist(str_extract_all(unlist(opp), "[0-9]{1,2}"))
opp <- as.numeric(opp)
for (r in 1:448) {if (opp[r]==0){opp[r] <-0} else{opp[r]<-as.vector(chessdf$scores)[opp[r]]}}
opp <-matrix(opp, ncol=7, byrow=TRUE)
opp <-as.data.frame(opp)
opp <-cbind(opp, Count = rowSums(!opp))
chessdf$postscore <-floor((rowSums(opp[1:7])/(7-opp$Count)))
write.table(chessdf, file = "chessratings.csv", sep = ",", row.names = FALSE)
```