NHL Regular Season Sports Reference Scraper

Adam Kiehl

7/8/20

This is a script that scrapes NHL regular season data from Hockey Reference (https://www.hockey-reference.com). Specifically, it extracts team-aggregated statistics and individual skater/goalie statistics (https://www.hockey-reference.com/teams/) for every playoff between 1999-2019. This code was originally sourced from Lyft Research Science Manager, Sean Taylor (https://github.com/seanjtaylor/learning-the-draft) and was modified with assistance from Colorado State University Ph.D student Connor Gibbs (https://github.com/ConGibbs10) and Philip Bulsink's Hockey and Chemistry Blog (https://pbulsink.github.io/blog/2016-12-26/scraping_player_data.html).

Variable Setup

Vectors and a mapping were compiled for later use in URL indexing. A `years` vector purposefully doesn't include the 2005 season due to a season-cancelling labor lockout. A `franchises` vector denotes teams by their standard abbreviations according to Sports Reference. A mapping was created between the two vectors in the form of `franchise/year`. Finally, corrections were made for the following:
1. Columbus Blue Jackets and Minnesota Wild being added as expansion teams in 2000
2. The Mighty Ducks of Anaheim being renamed to the Anaheim Ducks in 2006
3. The Phoenix Coyotes being renamed to the Arizona Coyotes in 2014
4. The Atlanta Thrashers becoming the Winnipeg Jets in 2011
5. The Vegas Golden Knights being added as an expansion team in 2017

```
years <- c('1999', '2000', '2001', '2002', '2003', '2004', '2006', '2007', '2008', '2009', '2010', '201
           '2015', '2016', '2017', '2018', '2019')
franchises <- c('ANA', 'PHX', 'BOS', 'BUF', 'CGY', 'CAR', 'CHI', 'COL', 'CBJ', 'DAL', 'DET', 'EDM', 'FL
                'NSH', 'NJD', 'NYI', 'NYR', 'OTT', 'PHI', 'PIT', 'SJS', 'STL', 'TBL', 'TOR', 'VAN', 'WS
map <- c()
for (i in franchises) {
  for (j in years) {
    map <- c(map, str_c(i, '/', j))
  }
}
map <- map[map %notin% c('CBJ/1999', 'CBJ/2000', 'MIN/1999', 'MIN/2000')]
map <- map[map %notin% c('ANA/1999', 'ANA/2000', 'ANA/2001', 'ANA/2002', 'ANA/2003', 'ANA/2004', 'ANA/2
map <- c(map, c('MDA/1999', 'MDA/2000', 'MDA/2001', 'MDA/2002', 'MDA/2003', 'MDA/2004', 'MDA/2006'))
map <- map[map %notin% c('PHX/2015', 'PHX/2016', 'PHX/2017', 'PHX/2018', 'PHX/2019')]
map <- c(map, c('ARI/2015', 'ARI/2016', 'ARI/2017', 'ARI/2018', 'ARI/2019'))
map <- map[map %notin% c('WPG/1999', 'WPG/2000', 'WPG/2001', 'WPG/2002', 'WPG/2003', 'WPG/2004', 'WPG/2
                         'WPG/2010', 'WPG/2011')]
map <- c(map, c('ATL/2000', 'ATL/2001', 'ATL/2002', 'ATL/2003', 'ATL/2004', 'ATL/2006', 'ATL/2007', 'AT
                'ATL/2011'))
map <- c(map, c('VEG/2018', 'VEG/2019'))
```

A `headers` list was created containing headers for each table of interest. These headers replace the ones scraped from Sports Reference for consistency and naming ease.

```
headers <- list()
headers[['skaters']] <- c('rank', 'player', 'age', 'pos', 'games', 'goals', 'assists', 'points', 'plus.
                          'even.goals','pp.goals', 'pk.goals', 'gw.goals', 'even.assists', 'pp.assists'
                          'shots', 'shot.percentage', 'toi', 'avg.toi', 'dps', 'ops', 'ps')
headers[['goalies']] <- c('rank', 'player', 'age', 'games', 'starts', 'wins', 'losses', 'ties', 'goals'
                          'save.percentage', 'avg.goals', 'shutouts', 'mins', 'quality.starts', 'quality
                          'really.bad.starts', 'goals.allowed.percentage', 'goals.saved.above.average',
                          'adjusted.goals.against.average', 'gps')
headers[['team_stats']] <- c('rank', 'team', 'age', 'games', 'wins', 'losses', 'ot.losses', 'points', '
                             'goals', 'goals.against', 'srs', 'sos', 'total.goals', 'power.play.goals',
                             'power.play.opportunities', 'power.play.percentage', 'power.play.goals.aga
                             'power.play.opportunities.against', 'pk.percentage', 'short.goals', 'short
                             'shots', 'shot.percentage', 'shots.against', 'save.percentage', 'pdo', 'sh
```

A `franchise.map` data frame was created mapping team names to their respective team codes. These codes will replace team names later for a standardized vocabulary between tables.

```
franchise.map <- list()
franchise.map[['team.code']] <- c('ANA', 'ARI', 'ATL', 'PHX', 'BOS', 'BUF', 'CGY', 'CAR', 'CHI', 'COL',
                                  'FLA', 'LAK', 'MDA','MIN', 'MTL', 'NSH', 'NJD', 'NYI', 'NYR', 'OTT',
                                  'TBL', 'TOR', 'VAN', 'WSH', 'WPG', 'VEG')
franchise.map[['team.name']] <- c('Anaheim Ducks', 'Arizona Coyotes', 'Atlanta Thrashers', 'Phoenix Coy
                                  'Buffalo Sabres', 'Calgary Flames', 'Carolina Hurricanes', 'Chicago Bl
                                  'Colorado Avalanche', 'Columbus Blue Jackets', 'Dallas Stars', 'Detroi
                                  'Edmonton Oilers', 'Florida Panthers', 'Los Angeles Kings', 'Mighty Du
                                  'Minnesota Wild', 'Montreal Canadiens', 'Nashville Predators', 'New Jo
                                  'New York Islanders', 'New York Rangers', 'Ottawa Senators', 'Philadel
                                  'Pittsburgh Penguins', 'San Jose Sharks', 'St. Louis Blues', 'Tampa Ba
                                  'Toronto Maple Leafs', 'Vancouver Canucks', 'Washington Capitals', 'Wi
                                  'Vegas Golden Knights')
franchise.map <- data.frame(franchise.map)
```

Function Setup

The `parse_teams` function is used to discriminate between all the tables found at a given URL. All the tables of a page and a table ID of interest are passed to the function and the table is returned with a revised header and a removed footer. This will be used specifically in the scraping of team data.

```
parse_teams <- function(tables, tbl.id) {
  results = list()
  for (tbl in tables) {
    id <- html_attr(tbl, 'id')
    if (id %in% tbl.id) {
      df <- html_table(tbl) %>%
        as_tibble(.name_repair = 'universal') %>%
        slice(., -n())
      results[[tbl.id]] <- df
    }
    return(bind_rows(results))
```

```
  }
}
```

The `parse_skaters` function is used to discriminate between all the tables found at a given URL. All the tables of a page and a table ID of interest are passed to the function and the table is returned with a revised header and a removed footer. This will be used specifically in the scraping of skater data.

```
parse_skaters <- function(tables, tbl.id) {
  results = list()
  for (tbl in tables) {
    id <- html_attr(tbl, 'id')
    if (id %in% tbl.id) {
      df <- html_table(tbl) %>%
        as_tibble(.name_repair = 'universal') %>%
        slice(., -n())
      if(ncol(df) == length(headers[[tbl.id]])) {
        colnames(df) <- headers[[tbl.id]]
      } else {

      }
      results[[tbl.id]] <- df
    }
  }
  return(bind_rows(results))
}
```

The `parse_goalies` function is used to discriminate between all the tables found at a given URL. All the tables of a page and a table ID of interest are passed to the function and the table is returned with a revised header and a removed footer. This will be used specifically in the scraping of goalie data.

```
parse_goalies <- function(tables, tbl.id) {
  results = list()
  for (tbl in tables) {
    id <- html_attr(tbl, 'id')
    if (id %in% tbl.id) {
      df <- html_table(tbl) %>%
        as_tibble(.name_repair = 'universal') %>%
        slice(., -c(1, n()))
      if(ncol(df) == length(headers[[tbl.id]])) {
        colnames(df) <- headers[[tbl.id]]
      } else {

      }
      results[[tbl.id]] <- df
    }
  }
  return(bind_rows(results))
}
```

Team Data

Here, scraping for the team-aggregated data is performed. https://www.hockey-reference.com/teams.html is indexed by team and then by year according to the map defined above. The `team_stats` table is extracted

from each year's page using the `parse_tables` function. A `team_reg.rds` file is written to document the data's structure and contents.

```
if(!file.exists('./data/team_reg.rds')){
  nodes <- list()
  for(mapping in map) {
    url <- paste('https://www.hockey-reference.com/teams/', mapping, '.html', sep ='')
    doc <- read_html(url)
    html.page <- doc %>%
        html_nodes('table') %>%
        parse_teams('team_stats')
    my.table <- html.page %>%
        mutate(., url = url)
    nodes[[mapping]] <- my.table
  }
  teams.table <- bind_rows(nodes, .id = 'mapping')
  write_rds(teams.table, './data/team_reg.rds', compress = 'xz')
}
```

Here, team-aggregated data is read from the `team_reg.rds` file, cleaned to include only fields of interest, and written to a `team_reg.csv` file. Additionally, an inner join was performed to match team names to their team codes and the team names were discarded.

```
if(!file.exists('./data/team_reg.csv')){
  team_reg <- read_rds('./data/team_reg.rds') %>%
    mutate(., url = if_else(str_sub(url, 1, 5) == 'http:',
                            str_c('https:', str_sub(url, 6, -1)),
                            url),
           key = ifelse(is.na(url),
                        year,
                        url),
           team = str_sub(mapping, end = 3),
           year = str_sub(mapping, start=5)) %>%
  arrange(., year, team)
  team_reg <- inner_join(team_reg, franchise.map, by = c('Team' = 'team.name')) %>%
    select(., year, team = team.code, games = GP, wins = W, points = PTS, goals = GF, goals.against = GA
    mutate_if(., is.character, str_trim) %>%
    write.csv('./data/team_reg.csv')
}
```

The result of team data scraping is shown below.

```
team_reg <- read_csv('./data/team_reg.csv')
head(team_reg)
```

```
## # A tibble: 6 x 21
##      X1  year team  games  wins points goals goals.against team.dps team.ops
##   <dbl> <dbl> <chr> <dbl> <dbl>  <dbl> <dbl>         <dbl>    <dbl>    <dbl>
## 1     1  1999 BOS      82    39     91   214           181     45.9       NA
## 2     2  1999 BUF      82    37     91   207           175     45.3       NA
## 3     3  1999 CAR      82    34     86   210           202     38.7       NA
## 4     4  1999 CGY      82    30     72   211           234     30.6       NA
## 5     5  1999 CHI      82    29     70   202           248     28.0       NA
```

```
## 6     6  1999 COL      82    44     98   239           205     38.5       NA
## # ... with 11 more variables: team.gps <dbl>, dps.avg <dbl>, ops.avg <dbl>,
## #   gps.avg <dbl>, team.ps <dbl>, dps.prop <dbl>, dps.prop.adj <dbl>,
## #   ops.prop <dbl>, ops.prop.adj <dbl>, dps.full.prop <dbl>,
## #   dps.full.prop.adj <dbl>
```

Skater Data

Here, scraping for the individual skater data is performed. https://www.hockey-reference.com/teams.html is indexed by team and then by year according to the map defined above. The `skaters` table is extracted from each year's page using the `parse_tables` function. HTML comment brackets are removed to unmask hidden tables. A `skaters_reg.rds` file is written to document the data's structure and contents.

```
if(!file.exists('./data/skaters_reg.rds')){
  nodes <- list()
  for(mapping in map) {
    url <- paste('https://www.hockey-reference.com/teams/', mapping, '.html', sep ='')
    doc <- read_html(url)
    html.page <- doc %>%
        html_nodes('table') %>%
        parse_skaters('skaters')
    my.table <- html.page %>%
        mutate(., url = url)
    nodes[[mapping]] <- my.table
  }
  teams.table <- bind_rows(nodes, .id = 'mapping')
  write_rds(teams.table, './data/skaters_reg.rds', compress = 'xz')
}
```

Here, individual skater data is read from the `skaters_reg.rds` file, cleaned to include only fields of interest, and written to a `skaters_reg.csv` file.

```
if(!file.exists('./data/skaters_reg.csv')){
  skaters_reg <- read_rds('./data/skaters_reg.rds') %>%
    mutate(., url = if_else(str_sub(url, 1, 5) == 'http:',
                            str_c('https:', str_sub(url, 6, -1)),
                            url),
           key = ifelse(is.na(url),
                        year,
                        url),
           team = str_sub(mapping, end = 3),
           year = str_sub(mapping, start=5)) %>%
  arrange(., year, team) %>%
  select(., year, team, player, pos, goals, assists, plus.minus, toi, dps.sr = dps, ops.sr = ops) %>%
  mutate_if(., is.character, str_trim) %>%
  filter(pos %in% c('RW', 'C', 'LW', 'D')) %>%
  write.csv('./data/skaters_reg.csv')
}
```

The result of skater data scraping is shown below.

```
skaters_reg <- read_csv('./data/skaters_reg.csv')
head(skaters_reg)
```

```
## # A tibble: 6 x 12
##    year team  player pos   goals assists plus.minus   toi   dps   ops dps.sr
##   <dbl> <chr> <chr>  <chr> <dbl>   <dbl>      <dbl> <dbl> <dbl> <dbl>  <dbl>
## 1  1999 BOS   Jason~ C        23      53          5  1835  2.65    NA    5.3
## 2  1999 BOS   Dmitr~ LW       29      42         11  1562  2.75    NA    6.1
## 3  1999 BOS   Ray B~ D        10      47         -7  2391  5.82    NA    4.5
## 4  1999 BOS   Serge~ LW       25      26         -6  1294  1.69    NA    4.3
## 5  1999 BOS   Joe T~ C        16      25          3  1243  1.78    NA    2.5
## 6  1999 BOS   Anson~ C        24      16          7  1030  1.65    NA    3.8
## # ... with 1 more variable: ops.sr <dbl>
```

Goalie Data

Here, scraping for the individual goalie data is performed. https://www.hockey-reference.com/teams.html is indexed by team and then by year according to the map defined above. The `goalies_reg` table is extracted from each year's page using the `parse_tables` function. HTML comment brackets are removed to unmask hidden tables. A `goalies_reg.csv` file is written to document the data's structure and contents.

```
if(!file.exists('./data/goalies_reg.rds')){
  nodes <- list()
  for(mapping in map) {
    url <- paste('https://www.hockey-reference.com/teams/', mapping, '.html', sep ='')
    doc <- read_html(url)
    html.page <- doc %>%
        html_nodes('table') %>%
        parse_goalies('goalies')
    my.table <- html.page %>%
        mutate(., url = url)
    nodes[[mapping]] <- my.table
  }
  teams.table <- bind_rows(nodes, .id = 'mapping')
  write_rds(teams.table, './data/goalies_reg.rds', compress = 'xz')
}
```

Here, individual goalie data is read from the `goalies_reg.rds` file, cleaned to include only fields of interest, and written to a `goalies_reg.csv` file.

```
if(!file.exists('./data/goalies_reg.csv')){
  goalies_reg <- read_rds('./data/goalies_reg.rds') %>%
    mutate(., url = if_else(str_sub(url, 1, 5) == 'http:',
                            str_c('https:', str_sub(url, 6, -1)),
                            url),
           key = ifelse(is.na(url),
                        year,
                        url),
           team = str_sub(mapping, end = 3),
           year = str_sub(mapping, start=5)) %>%
  arrange(., year, team) %>%
  select(., year, team, player, goals, shots, mins, gps.sr = gps) %>%
```

```
  mutate_if(., is.character, str_trim) %>%
  write.csv('./data/goalies_reg.csv')
}
```

The result of goalie data scraping is shown below.

```
goalies_reg <- read_csv('./data/goalies_reg.csv')
head(goalies_reg)
```

```
## # A tibble: 6 x 8
##    year team  player         goals shots  mins    gps gps.sr
##   <dbl> <chr> <chr>          <dbl> <dbl> <dbl>  <dbl>  <dbl>
## 1  1999 BOS   Byron Dafoe      133  1800  4001 14.1     14.1
## 2  1999 BOS   Robbie Tallas     43   421   987  2.00     2
## 3  1999 BUF   Dominik Hasek*   119  1877  3817 16.8     16.8
## 4  1999 BUF   Dwayne Roloson    42   460   911  2.73     2.7
## 5  1999 BUF   Martin Biron      10   120   281  0.816    0.8
## 6  1999 CAR   Arturs Irbe      135  1753  3643 13.1     13.1
```

Note: This script only writes .rds and .csv files if they do not already exist in the `data` subdirectory. To attempt a fresh scrape, first delete the .rds and .csv files of interest and then run this file. After, the data can be easily drawn into other scripts using the `read_csv` function.