1 football data co uk

Ing. Peter Tomko, M.A.

27/8/2020

```r
library(rvest)
library(stringr)
library(dplyr)
library(data.table)
library(xml2)
library(tidyverse)
library(DBI)
library(lubridate)

# - initialize the connection
con <- DBI::dbConnect(odbc::odbc(), "betting_ds")
```

Note: MIGHT BE REFRESHED ANYTIME.

Introduction

The document provides an automated way of refreshing the data necessary for the project. There are two types of data source - historical match data `www.football-data.co.uk/data.php` and actual odds provided by `Tipsport`.

Get Seasons to be refreshed

After creating the database schema with appropriate tables, the query is created in order to update only relevant part of the data. It is not required to download all data each time, when the document is knitted - i.e. the incremental upload is implemented.

```sql
select * from t_match_calendar;
```

Historical Data Download - Matches Results

The data are located on the website `https://www.football-data.co.uk/data.php` in the `.csv` format. The procedure implemented within this document is following

- get seasons that will be updated
- get all links of `.csv` files for the major leagues
- download data using `tidy` framework
- write the data into corresponding tables

The data sources contain information about the match played, league and results with extra information about number of shots, cards etc. Detail description of data source might be found in the `README.md` document Branch Johnson and Johnson case study.

```
start_url <- c("https://www.football-data.co.uk/data.php")
start_webpage <- read_html(start_url)

links_all <- start_webpage %>%
  html_nodes("a") %>%
  html_attr("href") %>%
  str_subset("\\m.php") %>%
  str_subset(string = ., pattern = "https://") %>%
  str_subset(string = ., pattern = "scam", negate = TRUE) %>%
  str_subset(string = ., pattern = "betting", negate = TRUE) %>%
  str_subset(string = ., pattern = "downloadm", negate = TRUE) %>%
  str_subset(string = ., pattern = "gambling", negate = TRUE) %>%
  unique()

# ----- Get file links -----
links_data <-
  data.frame(name_league = links_all) %>%
  group_by(name_league) %>%
  mutate(file_link =
           map(name_league, function(i_link){

             data.frame(file_link =
                          html(i_link) %>%
                          html_nodes("a") %>%
                          html_attr("href") %>%
                          str_subset("\\.csv") %>%
                          as.character())
           })) %>%
  unnest(c(file_link)) %>%
  as.data.frame() %>%

  # - subset only links to be refreshed (INCREMENTAL REFRESH)
  left_join(., c_match_calendar %>%
              filter(created_at > Sys.Date() - 360) %>%
              mutate(is_refreshed = 1) %>%
              select(file_link, is_refreshed) %>%
              distinct()) %>%

  as.data.frame() %>%
  filter(is.na(is_refreshed)) %>%
  select(-is_refreshed) %>%
  as.data.frame()

# ----- Download Files -----
master_data <-
  links_data %>%
  group_by(file_link) %>%
  mutate(orig_data =
           map(file_link,
               function(i_link) {

                 data <-
```

```
                   # - read data
                   read.csv(paste("https://www.football-data.co.uk/",
                                  i_link, sep = ""),
                            stringsAsFactors = FALSE) %>%
                   as.data.frame()

                 # - return
                 return(data)})) %>%

  # - get season and league
  rowwise() %>%
  mutate(season = str_split(file_link, pattern = "/")[[1]][2],
         league =
           str_replace(str_split(file_link, pattern = "/")[[1]][3],
                       pattern = ".csv", replacement = "")) %>%
  select(-name_league, -file_link) %>%
  as.data.frame()

rm(start_webpage)
rm(start_url)
rm(links_all)
```

Data Pre-Processing

Applied procedures to process the data (in the order, how they are applied in the `tidy` pipeline)

- full-time result, i.e. `FTR` needs to have only admissible values `H, D, A`
- some `.csv` files contains columns `HT` and `AT` instead of standard `HomeTeam` and `AwayTeam`, therefore the synchronization is necessary
- `created_at` represents date of a match. The column is not standard in each `.csv` files, therefore it is created based on the `get_day`, `get_month` and `get_year`.
- using `FTR = ifelse("FTR" %in% names(.), FTR, NA)`, each column is expected to be present within the data, otherwise missing values are added.
- all columns expected to be numeric, except `created_at`, `HomeTeam`, `AwayTeam` and `FTR`
- using `gather()` function, the data are transformed to long format, where `HomeTeam` and `AwayTeam` creates new column, i.e. `is_home`

```
master_data <-
  master_data %>%
  group_by(file_link) %>%
  mutate(prep_data =
           map(orig_data,
               function(i_data) {

                 data <-

                   i_data %>%
                   as.data.frame() %>%

                   # - remove unimportant rows
                   filter(FTR %in% c("H", "D", "A")) %>%

```

```r
# - Home Team and Away Team (some .csv files have AT and HT)
rowwise() %>%
mutate(HomeTeam =
         ifelse("HT" %in% names(.), HT, HomeTeam),
       AwayTeam =
         ifelse("AT" %in% names(.), AT, AwayTeam)) %>%

# - create standardized date column (created_at - match date)
rowwise() %>%
mutate(get_day = str_split(Date, "/")[[1]][1],
       get_month = str_split(Date, "/")[[1]][2],
       get_year = str_split(Date, "/")[[1]][3]) %>%
mutate(year_adj =
         ifelse(get_year %in% c("93", "94", "95", "96",
                                "97", "98", "99"),
                paste("19", get_year, sep = ""),
                get_year)) %>%
mutate(year_adj =
         ifelse(str_length(year_adj) < 3,
                paste("20", get_year, sep = ""),
                year_adj)) %>%
mutate(created_at =
         as.Date(paste(get_day, get_month, year_adj,
                       sep = "/"),
                 format = "%d/%m/%Y")) %>%
distinct() %>%

# - ensure the columns exist
rowwise() %>%
mutate(FTR = ifelse("FTR" %in% names(.), FTR, NA),
       FTHG = ifelse("FTHG" %in% names(.), FTHG, NA),
       FTAG = ifelse("FTAG" %in% names(.), FTAG, NA),
       HS = ifelse("HS" %in% names(.), HS, NA),
       AS = ifelse("AS" %in% names(.), AS, NA),
       HST = ifelse("HST" %in% names(.), HST, NA),
       AST = ifelse("AST" %in% names(.), AST, NA),
       HF = ifelse("HF" %in% names(.), HF, NA),
       AF = ifelse("AF" %in% names(.), AF, NA),
       HC = ifelse("HC" %in% names(.), HC, NA),
       AC = ifelse("AC" %in% names(.), AC, NA),
       HY = ifelse("HY" %in% names(.), HY, NA),
       AY = ifelse("AY" %in% names(.), AY, NA),
       HR = ifelse("HR" %in% names(.), HR, NA),
       AR = ifelse("AR" %in% names(.), AR, NA),
       B365H = ifelse("B365H" %in% names(.), B365H, NA),
       B365D = ifelse("B365D" %in% names(.), B365D, NA),
       B365A = ifelse("B365A" %in% names(.), B365A, NA),
       BWH = ifelse("BWH" %in% names(.), BWH, NA),
       BWD = ifelse("BWD" %in% names(.), BWD, NA),
       BWA = ifelse("BWA" %in% names(.), BWA, NA),
       IWH = ifelse("IWH" %in% names(.), IWH, NA),
       IWD = ifelse("IWD" %in% names(.), IWD, NA),
       IWA = ifelse("IWA" %in% names(.), IWA, NA),
       PSH = ifelse("PSH" %in% names(.), PSH, NA),
       PSD = ifelse("PSD" %in% names(.), PSD, NA),
       PSA = ifelse("PSA" %in% names(.), PSA, NA),
       WHH = ifelse("WHH" %in% names(.), WHH, NA),
       WHD = ifelse("WHD" %in% names(.), WHD, NA),
       WHA = ifelse("WHA" %in% names(.), WHA, NA),
       VCH = ifelse("VCH" %in% names(.), VCH, NA),
       VCD = ifelse("VCD" %in% names(.), VCD, NA),
       VCA = ifelse("VCA" %in% names(.), VCA, NA),
       B365.2.5 = ifelse("B365.2.5" %in% names(.),
                         `B365.2.5`, NA),
       B365.2.5.1 = ifelse("B365.2.5.1" %in% names(.),
                           `B365.2.5.1`, NA),
       P.2.5 = ifelse("P.2.5" %in% names(.), `P.2.5`, NA),
       P.2.5.1 = ifelse("P.2.5.1" %in% names(.),
                        `P.2.5.1`, NA),
       GB.2.5 =
         ifelse("GB.2.5" %in% names(.), `GB.2.5`, NA),
       GB.2.5.1 = ifelse("GB.2.5.1" %in% names(.),
                         `GB.2.5.1`, NA)) %>%

# - select relevant columns
select(created_at, HomeTeam, AwayTeam,
       FTR, FTHG, FTAG,
       HS, AS, HST, AST, HF, AF, HC, AC, HY, AY, HR, AR,
       B365H, B365D, B365A, BWH, BWD, BWA,
       IWH, IWD, IWA, PSH, PSD, PSA,
       WHH, WHD, WHA, VCH, VCD, VCA,
       B365.2.5, B365.2.5.1, P.2.5, P.2.5.1,
       GB.2.5, GB.2.5.1) %>%
distinct() %>%

# - create match id
group_by(created_at) %>%
mutate(match_id = row_number()) %>%
as.data.frame() %>%

# - get numeric values at selected columns
mutate_at(vars(-created_at, -HomeTeam, -AwayTeam, -FTR),
          as.numeric) %>%

# - gather by team and is_home
gather(., is_home, team,
       -created_at, -match_id, -FTR,

       # - Match Statistics
       -FTHG, -FTAG, -HS, -AS, -HST, -AST, -HF, -AF,
       -HC, -AC, -HY, -AY, -HR, -AR,

       # - Bet Odds
       -B365H, -B365D, -B365A, -BWH, -BWD, -BWA,
       -IWH, -IWD, -IWA, -PSH, -PSD, -PSA,
       -WHH, -WHD, -WHA, -VCH, -VCD, -VCA,
```

```r
                          -B365.2.5, -B365.2.5.1,
                          -P.2.5, -P.2.5.1,
                          -GB.2.5, -GB.2.5.1) %>%

                   # - dummy is_home
                   mutate(is_home =
                            recode(is_home,
                                   "HomeTeam" = 1, "AwayTeam" = 0)) %>%
                   as.data.frame()

                 return(data)}))

rm(links_data)
```

Uploads for `t_match_calendar` and `t_match_stats`

Prepare and write into `t_match_calendar`

`t_match_calendar` table contains information about the downloaded matches - columns are self-explanatory from the table creation script.

```r
t_match_calendar <-
  master_data %>%
  group_by(file_link) %>%
  mutate(cal_data = map(prep_data, function(i_data){
    temp <- i_data %>%

      # - get Away Team
      select(created_at, team, match_id, is_home) %>%
      filter(is_home == 0) %>%
      rename(AwayTeam = team) %>%
      select(-is_home) %>%
      distinct() %>%
      as.data.frame() %>%

      # - join Home Team
      left_join(., i_data %>%
                  select(created_at, team, match_id, is_home) %>%
                  filter(is_home == 1) %>%
                  rename(HomeTeam = team) %>%
                  select(-is_home) %>%
                  distinct() %>%
                  as.data.frame())

    return(temp)
  })) %>%
  select(season, league, cal_data) %>%
  unnest(c(cal_data)) %>%

  select(-match_id) %>%
  as.data.frame() %>%

  # - subset rows to be uploaded
  left_join(., c_match_calendar %>%
              mutate(is_selected = 1) %>%
              rename(HomeTeam = home_t,
                     AwayTeam = away_t)) %>%
  as.data.frame() %>%

  # - get starting match index
  mutate(start_id = coalesce(max(match_id), 1)) %>%
  filter(is.na(is_selected)) %>%
  select(-is_selected) %>%

  # - create index column
  as.data.frame() %>%
  mutate(match_id = start_id + row_number() - 1) %>%
  select(-start_id) %>%

  # - rename columns
  rename(home_t = HomeTeam,
         away_t = AwayTeam) %>%
  as.data.frame()

if(nrow(t_match_calendar) > 0) {
  dbWriteTable(con, "t_match_calendar", t_match_calendar,
               row.names = F, append = T)}
```

```
FALSE Error in result_insert_dataframe(rs@ptr, values, batch_rows): nanodbc/nanodbc.cpp:1617: 23505: ERI
FALSE Error while executing the query   RROR: duplicate key value violates unique constraint "t_match_ca
FALSE Error while executing the query
```

Prepare and write into t_match_stats

The basic building block of the table are columns - `created_at`, `team`, `is_home`, `season`, `league`, `match_id` (added in order to be able to identify `HomeTeam` and `AwayTeam`). For this column basis the numerical attributes are added - for example `HS` (shots by `HomeTeam`), `HST` (shots on target by `HomeTeam`) etc. Furthermore, the bookmakers odds are included - for example `B365H` (odds by B365 for `HomeTeam` win), `B365.2.5` (odds by B365 that the match and with less than 2.5 goals) etc.

```
t_match_stats <-

  # - unnest
  master_data %>%
  select(season, league, prep_data) %>%
  unnest(c(prep_data)) %>%
  as.data.frame() %>%

  # remove columns
  select(-file_link) %>%
  as.data.frame() %>%

  # - keep rows that should be updated
  left_join(., t_match_calendar %>%
              select(created_at, home_t, away_t, league, season) %>%
              mutate(is_selected = 1) %>%
              gather(., is_home, team, -created_at, -season, -league) %>%
              mutate(is_home = recode(is_home, "home_t" = 1, "away_t" = 0),
```

```r
                     is_selected = 1)) %>%
  as.data.frame() %>%
  filter(is_selected == 1) %>%
  as.data.frame() %>%
  select(-is_selected) %>%
  rename_at(names(.), tolower)

if(nrow(t_match_stats) > 0) {
  dbWriteTable(con, "t_match_stats", t_match_stats,
               row.names = F, append = T)}
```