Preprocess\_IndividualFiles\_10Nov19.R

tylercampbell

2019-11-21

#############################################################################  
#### Introduction ####  
  
## Title: F1 Preprocessing - Individual Source Files  
## Author: Tyler Campbell  
## Date: Nov 2019  
  
## IMPORTANT:  
 # If running full script (and not wanting to overwrite files), avoid 'Write files' section at bottom  
  
## Description:  
 # This R script performs cleaning of individual raw F1 source data files  
 # Sections are arranged in alphabetical order and contain section-specific notes  
 # A separate script will be created for joining the data frames  
  
#############################################################################  
  
  
  
  
#### Markdown ####  
  
# ignore this section for normal processing  
# load blank workspace when compiling/knitting  
# extra dots also are added to file paths for compilation purposes, remove when running script  
  
  
  
  
#### Load packages ####  
  
library(tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.2.1 ──

## ✔ ggplot2 3.2.1 ✔ purrr 0.3.0  
## ✔ tibble 2.1.3 ✔ dplyr 0.8.3  
## ✔ tidyr 1.0.0 ✔ stringr 1.4.0  
## ✔ readr 1.3.1 ✔ forcats 0.4.0

## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()

library(lubridate)

##   
## Attaching package: 'lubridate'

## The following object is masked from 'package:base':  
##   
## date

library(chron)

##   
## Attaching package: 'chron'

## The following objects are masked from 'package:lubridate':  
##   
## days, hours, minutes, seconds, years

#### Load files ####  
  
# define input path  
path\_in <- "../Raw Source Data"  
  
# read files  
circuits <- read.csv(file.path(path\_in, "circuits.csv"), stringsAsFactors = F)  
constructorResults <- read.csv(file.path(path\_in, "constructorResults.csv"), stringsAsFactors = F)  
constructors <- read.csv(file.path(path\_in, "constructors.csv"), stringsAsFactors = F)  
constructorStandings <- read.csv(file.path(path\_in, "constructorStandings.csv"), stringsAsFactors = F)  
drivers <- read.csv(file.path(path\_in, "drivers.csv"), stringsAsFactors = F)  
driverStandings <- read.csv(file.path(path\_in, "driverStandings.csv"), stringsAsFactors = F)  
lapTimes <- read.csv(file.path(path\_in, "lapTimes.csv"), stringsAsFactors = F)  
pitStops <- read.csv(file.path(path\_in, "pitStops.csv"), stringsAsFactors = F)  
qualifying <- read.csv(file.path(path\_in, "qualifying.csv"), stringsAsFactors = F, skipNul = TRUE, na.strings = c("","NULL","NA")) # also convert embedded NULLs and blanks to NA))  
races <- read.csv(file.path(path\_in, "races.csv"), stringsAsFactors = F, skipNul = TRUE, na.strings = c("","NULL","NA")) # also convert embedded NULLs and blanks to NA))  
results <- read.csv(file.path(path\_in, "results.csv"), stringsAsFactors = F, na.strings = c("","NULL","NA")) # also convert embedded NULLs and blanks to NA))  
seasons <- read.csv(file.path(path\_in, "seasons.csv"), stringsAsFactors = F)  
status <- read.csv(file.path(path\_in, "status.csv"), stringsAsFactors = F)  
  
  
  
  
#### Clean data ####  
  
## circuits -----------------------------------------------------------------  
  
# inspect file for completeness  
colSums(circuits == "" | circuits == "NULL" | is.na(circuits))

## circuitId circuitRef name location country lat   
## 0 0 0 0 0 0   
## lng alt url   
## 0 72 0

# remove columns: 'name', 'alt' ('name' is dirtier and lengthier than 'circuitRef' & 'alt' appears to have no meaning)  
# keep columns: 'url', 'lat', 'long' in case wanting to scrape wiki pages and/or calculate distance (e.g. from home)  
circuits <- subset(circuits, select = -c(name, alt))  
  
# clean dirty location names (caused by accent characters)  
circuits[c(4,18,20), "location"] <- c("Montmelo", "Sao Paulo", "Nurburg")  
  
# rename columns  
circuits <- circuits %>%  
 rename("circuit\_name" = "circuitRef",  
 "circuit\_city" = "location",  
 "circuit\_country" = "country",  
 "circuit\_lat" = "lat",  
 "circuit\_long" = "lng",  
 "circuit\_url" = "url"  
 )  
  
  
  
  
## constructorResults -------------------------------------------------------  
  
# 'D' in 'status' column represents 'Disqualified' due to Spygate scandal in 2007 season (https://en.wikipedia.org/wiki/2007\_Formula\_One\_espionage\_controversy)  
  
# inspect file for completeness  
colSums(constructorResults == "" | constructorResults == "NULL" | is.na(constructorResults))

## constructorResultsId raceId constructorId   
## 0 0 0   
## points status   
## 0 11125

# inspect distribution of 'status' (see above description of 'D') & remove column  
table(constructorResults$status)

##   
## D NULL   
## 17 11125

constructorResults <- subset(constructorResults, select = -status)  
  
# convert 'points' to int type  
constructorResults$points <- as.integer(constructorResults$points)  
  
# rename columns  
constructorResults <- constructorResults %>%  
 rename("constructorResult\_pointsPerRace" = "points")  
  
  
  
  
## constructors -------------------------------------------------------------  
  
# again keeping 'url' column incase of desire to scrape  
  
# inspect for completeness  
colSums(constructors == "" | constructors == "NULL" | is.na(constructors))

## constructorId constructorRef name nationality url   
## 0 0 0 0 0   
## X   
## 208

# remove columns: 'name', 'X' ('X' is entirely NA values)  
constructors <- subset(constructors, select = -c(X, name))  
  
# rename columns  
constructors <- constructors %>%  
 rename("constructor\_name" = "constructorRef",  
 "constructor\_nationality" = "nationality",  
 "constructor\_url" = "url"  
 )  
  
  
  
  
## constructorStandings -----------------------------------------------------  
  
# inspect file for completeness  
colSums(constructorStandings == "" | constructorStandings == "NULL" | is.na(constructorStandings))

## constructorStandingsId raceId constructorId   
## 0 0 0   
## points position positionText   
## 0 0 0   
## wins X   
## 0 11896

# remove column: 'positionText' ('positionText' is identical to 'position' with the exception of 'E' for excluded due to "Spygate" scandal (see constructorResults section))  
# remove column: 'X' ('X' is entirely NA values)  
constructorStandings <- subset(constructorStandings, select = -c(positionText, X))  
  
# convert 'points' to int type  
constructorStandings$points <- as.integer(constructorStandings$points)  
  
# rename columns  
constructorStandings <- constructorStandings %>%  
 rename("constructorStanding\_runningTotalPointsInSeason" = "points",  
 "constructorStanding\_runningPositionInSeason" = "position",  
 "constructorStanding\_runningTotalWinsInSeason" = "wins"  
 )  
  
  
  
  
## drivers ------------------------------------------------------------------  
  
# inspect file for completeness  
colSums(drivers == "" | drivers == "NULL" | is.na(drivers))

## driverId driverRef number code forename surname   
## 0 0 804 757 0 0   
## dob nationality url   
## 1 0 1

# remove column: 'number' ('number' is arbitrary driver/car number)  
# remove columns: 'code' 'forename', 'surname' ('code' serves no purpose $ forename' and 'surname' are messier and lengthier than 'driverRef')  
drivers <- subset(drivers, select = -c(number, code, forename, surname))  
  
# clean dob column and convert to date type variable using lubridate function  
drivers[415, "dob"] <- "12/08/1993"  
drivers$dob <- dmy(drivers$dob)

## Warning: 7 failed to parse.

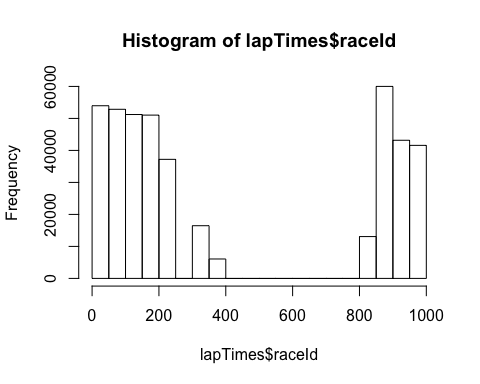
# manually handle the 7 records that failed to parse and re-run lubridate  
drivers[c(590, 704, 742, 751, 761, 787, 792), "dob"] <-  
 c("1899-08-03", "1898-11-01", "1896-12-28", "1899-10-15", "1899-10-13", "1898-06-09", "1898-10-18")  
drivers$dob <- ymd(drivers$dob)  
  
# rename columns  
drivers <- drivers %>%  
 rename("driver\_name" = "driverRef",  
 "driver\_dob" = "dob",  
 "driver\_nationality" = "nationality",  
 "driver\_url" = "url"  
 )  
  
  
  
  
## driverStandings ----------------------------------------------------------  
  
# thought: if you could associate a race with its round #, then driver with most points after final round wins championship  
  
# inspect file for completeness  
colSums(driverStandings == "" | driverStandings == "NULL" | is.na(driverStandings))

## driverStandingsId raceId driverId points   
## 0 0 0 0   
## position positionText wins   
## 0 0 0

# remove column: 'positionText' ('positionText' is identical to 'position' with the exception of 'D' for disqualification from the championship due to intentional collision)  
driverStandings <- subset(driverStandings, select = -positionText)  
  
# convert 'points' to int type  
driverStandings$points <- as.integer(driverStandings$points)  
  
# rename columns  
driverStandings <- driverStandings %>%  
 rename("driverStanding\_runningTotalPointsInSeason" = "points",  
 "driverStanding\_runningPositionInSeason" = "position",  
 "driverStanding\_runningTotalWinsInSeason" = "wins"  
 )  
  
  
  
  
## lapTimes -----------------------------------------------------------------  
  
# file is missing lap times from at least races 400-800 (by raceID)  
# consider excluding lap 25 of raceId 847 (2011 Canadian Grand Prix had "torrential rains" that caused a single lap to clock in at over 2 hrs)  
# consider excluding file altogether  
  
# inspect file for completeness  
colSums(lapTimes == "" | lapTimes == "NULL" | is.na(lapTimes))

## raceId driverId lap position time   
## 0 0 0 0 0   
## milliseconds   
## 0

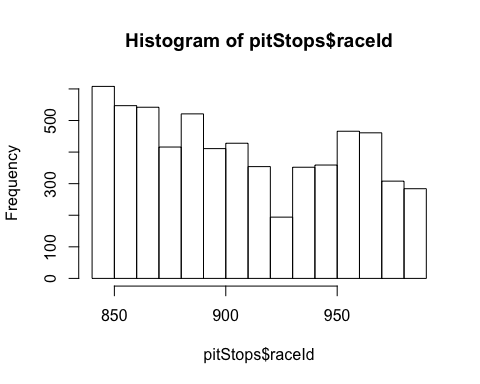
hist(lapTimes$raceId)



# remove column: 'time' ('time' == 'milliseconds')  
lapTimes <- subset(lapTimes, select = -time)  
  
# create an average lap time column  
lapTimes <- lapTimes %>%  
 group\_by(raceId, driverId) %>%  
 mutate(lapTime\_avgMillisec = as.integer(mean(milliseconds)))  
  
# ungroup columns and convert back to data frame  
lapTimes <- lapTimes %>%  
 ungroup() %>%  
 as.data.frame()  
  
# rename columns  
lapTimes <- lapTimes %>%  
 rename("lapTime\_positionInLap" = "position",  
 "lapTime\_millisec" = "milliseconds"  
 )  
  
  
  
  
## pitStops -----------------------------------------------------------------  
  
# file excludes races before 2011 and after 2017  
# consider aggregating 'milliseconds' per driver per race to obtain a single total pit time per race  
# consider obtaining largest number of 'stop' per driver per race to get total number of pit stops per race  
# consider excluding file altogether  
  
# inspect file for completeness  
colSums(pitStops == "" | pitStops == "NULL" | is.na(pitStops))

## raceId driverId stop lap time   
## 0 0 0 0 0   
## duration milliseconds   
## 0 0

hist(pitStops$raceId)



# remove column: 'duration' ('duration' == 'milliseconds')  
pitStops <- subset(pitStops, select = -duration)  
  
# create a total number of stops per race column  
pitStops <- pitStops %>%  
 group\_by(raceId, driverId) %>%  
 arrange(raceId, driverId, stop) %>%  
 mutate(pitStop\_totalStops = last(stop))  
  
# create an average pitstop duration column  
pitStops <- pitStops %>%  
 group\_by(raceId, driverId) %>%  
 mutate(pitStop\_avgDuration = as.integer(mean(milliseconds)))  
  
# ungroup columns and convert back to data frame  
pitStops <- pitStops %>%  
 ungroup() %>%  
 as.data.frame()  
  
# rename columns  
pitStops <- pitStops %>%  
 rename("pitStop\_stopNum" = "stop",  
 "pitStop\_lap" = "lap",  
 "pitStop\_timeOfDay" = "time",  
 "pitStop\_durationMillisec" = "milliseconds"  
 )  
  
  
  
  
## qualifying ---------------------------------------------------------------  
  
# 'position' should be the same as grid in results file  
# consider excluding file altogether because of NAs  
  
# inspect file for completeness   
colSums(qualifying == "" | qualifying == "NULL" | is.na(qualifying))

## qualifyId raceId driverId constructorId number   
## 0 0 0 0 0   
## position q1 q2 q3   
## 0 119 3864 5338

# remove column: 'number' ('number' is arbitrary driver/car number)  
qualifying <- subset(qualifying, select = -number)  
  
# correct erroneous record in q2  
qualifying[5663, "q2"] <- "1:48.552"  
  
# convert 'q1':'q3' to milliseconds  
qualifying <- qualifying %>%  
 separate(q1, into = c("q1\_minuteSecond", "q1\_decimal"), sep = "\\.") %>%  
 separate(q1\_minuteSecond, into = c("q1\_minute", "q1\_second"), convert = TRUE) %>%  
 mutate(q1\_milliseconds = paste0(((q1\_minute \* 60) + q1\_second), q1\_decimal))  
  
qualifying <- qualifying %>%  
 separate(q2, into = c("q2\_minuteSecond", "q2\_decimal"), sep = "\\.") %>%  
 separate(q2\_minuteSecond, into = c("q2\_minute", "q2\_second"), convert = TRUE) %>%  
 mutate(q2\_milliseconds = paste0(((q2\_minute \* 60) + q2\_second), q2\_decimal))  
  
qualifying <- qualifying %>%  
 separate(q3, into = c("q3\_minuteSecond", "q3\_decimal"), sep = "\\.") %>%  
 separate(q3\_minuteSecond, into = c("q3\_minute", "q3\_second"), convert = TRUE) %>%  
 mutate(q3\_milliseconds = paste0(((q3\_minute \* 60) + q3\_second), q3\_decimal))  
  
# convert 'NANA's (caused by concatenation) to NAs  
qualifying[qualifying == "NANA"] <- NA  
  
# remove columns: 'q1\_minute' through 'q3\_decimal' (temp columns created for parsing)  
qualifying <- subset(qualifying, select = -c(q1\_minute:q3\_decimal))  
  
# rename columns  
qualifying <- qualifying %>%  
 rename("qualifying\_finishPosition" = "position",  
 "qualifying\_q1Millisec" = "q1\_milliseconds",  
 "qualifying\_q2Millisec" = "q2\_milliseconds",  
 "qualifying\_q3Millisec" = "q3\_milliseconds"  
 )  
  
  
  
  
## races --------------------------------------------------------------------  
  
# race time is empty from 1950 to 2005  
# consider removing 'time'  
# remove 'name' which will be redundant after merge (circuit table contains ref)  
  
# inspect file for completeness  
colSums(races == "" | races == "NULL" | is.na(races))

## raceId year round circuitId name date time   
## 0 0 0 0 0 0 731   
## url   
## 0

# convert 'date' to date type  
races$date <- ymd(races$date)  
  
# convert 'time' to time type  
races$time <- times(races$time)  
  
# rename columns  
races <- races %>%  
 rename("race\_year" = "year",  
 "race\_round" = "round",  
 "race\_name" = "name",  
 "race\_date" = "date",  
 "race\_time" = "time",  
 "race\_url" = "url"  
 )  
  
  
  
  
## results ------------------------------------------------------------------  
  
# 'time' is inconsistently represented as minutes:seconds beyond 1 hr for first place driver, then the gap for next several drivers  
# no time is recorded for racers greater than 1 lap behind the winner  
# consider just including 'time' for winner, otherwise it's reflected as "gap" and often +1 laps rather than a time  
# 'position' = raceFinish, 'positionText' = raceFinish or descriptor of retired/disqualified etc., 'positionOrder' = raceFinish or order of retired/disqualified  
# positionText: D=Disqualified, E=Excluded, F=Did Not (/failed to) Qualify, N=Not Classified, R=Retired, W=Withrew  
# consider converting 'positionText' to boolean (i.e. Finished, Did not finish)  
# fastest lap data is empty or NA from 1950 to 2004  
# 'grid' is mostly a duplicate of position in qualifying table, but is likely more reliable  
  
# inspect file for completeness  
colSums(results == "" | results == "NULL" | is.na(results))

## resultId raceId driverId constructorId   
## 0 0 0 0   
## number grid position positionText   
## 6 0 10550 0   
## positionOrder points laps time   
## 0 0 0 17773   
## milliseconds fastestLap rank fastestLapTime   
## 17774 18394 18246 18394   
## fastestLapSpeed statusId   
## 18394 0

# remove columns: 'position', 'number', 'time', 'fastestLap', 'rank', 'fastestLapTime', 'fastestLapSpeed'  
# ('position' is redundant, 'number' is arbitrary driver/car number, 'time' is represented more cleanly in 'milliseconds')  
results <- subset(results, select = -c(position, number, time))  
  
# convert positionText column to finish description  
results <- mutate(results, result\_finishDescription =   
 ifelse(results$positionText == "D", "Disqualified",  
 ifelse(results$positionText == "E", "Excluded",  
 ifelse(results$positionText == "F", "FailedToFinish",  
 ifelse(results$positionText == "N", "NotClassified",  
 ifelse(results$positionText == "R", "Retired",  
 ifelse(results$positionText == "W", "Withdrew",  
 "Finished"  
 )))))))  
results <- subset(results, select = -positionText)  
  
# convert fastestLapTime to milliseconds  
results <- results %>%  
 separate(fastestLapTime, into = c("fLT\_minuteSecond", "fLT\_decimal"), sep = "\\.") %>%  
 separate(fLT\_minuteSecond, into = c("fLT\_minute", "fLT\_second"), convert = TRUE) %>%  
 mutate(fLT\_milliseconds = paste0(((fLT\_minute \* 60) + fLT\_second), fLT\_decimal, "00"))  
  
# convert 'NANA00's (caused by concatenation) to NAs  
results$fLT\_milliseconds[results$fLT\_milliseconds == "NANA00"] <- NA  
  
# convert 'points', 'fastestLapSpeed', and 'fLT\_milliseconds' to appropriate number types  
results$points <- as.integer(results$points)  
results$fLT\_milliseconds <- as.integer(results$fLT\_milliseconds)  
results$fastestLapSpeed <- as.double(results$fastestLapSpeed)

## Warning: NAs introduced by coercion

# remove columns: 'q1\_minute' through 'q3\_decimal' (temp columns created for parsing)  
results <- subset(results, select = -c(fLT\_minute:fLT\_decimal))  
  
# rename columns  
results <- results %>%  
 rename("result\_startingGridPosition" = "grid",  
 "result\_finishOrder" = "positionOrder",  
 "result\_pointsEarned" = "points",  
 "result\_lapsCompleted" = "laps",  
 "result\_finishTimeMillisec" = "milliseconds",  
 "result\_fastestLapTimeMillisec" = "fLT\_milliseconds",  
 "result\_fastestLap" = "fastestLap",  
 "result\_fastestLapRank" = "rank",  
 "result\_fastestLapSpeed" = "fastestLapSpeed"  
 )  
  
  
  
  
## seasons ------------------------------------------------------------------  
  
# somewhat irrelevant file unless wanting to scrape wiki pages  
  
# inspect file for completeness  
colSums(seasons == "" | seasons == "NULL" | is.na(seasons))

## year url   
## 0 0

# convert year to date type  
seasons$year <- as.character(seasons$year)  
seasons$year <- as.Date(paste(seasons$year, 1, 1, sep = "-")) # beginning of year  
seasons$year <- as.Date(paste(seasons$year, 12, 31, sep = "-")) # end of year  
  
# rename columns  
seasons <- seasons %>%  
 rename("season\_year" = "year",  
 "season\_url" = "url"  
 )  
  
  
  
  
## status ------------------------------------------------------------------  
  
# inspect file for completeness  
colSums(status == "" | status == "NULL" | is.na(status))

## statusId status   
## 0 0

# rename column  
colnames(status) [colnames(status) == "status"] <- "status\_description"  
  
  
  
  
#### Write files ####  
  
# define output path  
path\_out1 <- "../Processed Data"  
  
# write to csvs  
write.csv(circuits, file.path(path\_out1, "circuits\_clean.csv"), row.names = F)  
write.csv(constructorResults, file.path(path\_out1, "constructorResults\_clean.csv"), row.names = F)  
write.csv(constructors, file.path(path\_out1, "constructors\_clean.csv"), row.names = F)  
write.csv(constructorStandings, file.path(path\_out1, "constructorStandings\_clean.csv"), row.names = F)  
write.csv(drivers, file.path(path\_out1, "drivers\_clean.csv"), row.names = F)  
write.csv(driverStandings, file.path(path\_out1, "driverStandings\_clean.csv"), row.names = F)  
write.csv(lapTimes, file.path(path\_out1, "lapTimes\_clean.csv"), row.names = F)  
write.csv(pitStops, file.path(path\_out1, "pitStops\_clean.csv"), row.names = F)  
write.csv(qualifying, file.path(path\_out1, "qualifying\_clean.csv"), row.names = F)  
write.csv(races, file.path(path\_out1, "races\_clean.csv"), row.names = F)  
write.csv(results, file.path(path\_out1, "results\_clean.csv"), row.names = F)  
write.csv(seasons, file.path(path\_out1, "seasons\_clean.csv"), row.names = F)  
write.csv(status, file.path(path\_out1, "status\_clean.csv"), row.names = F)