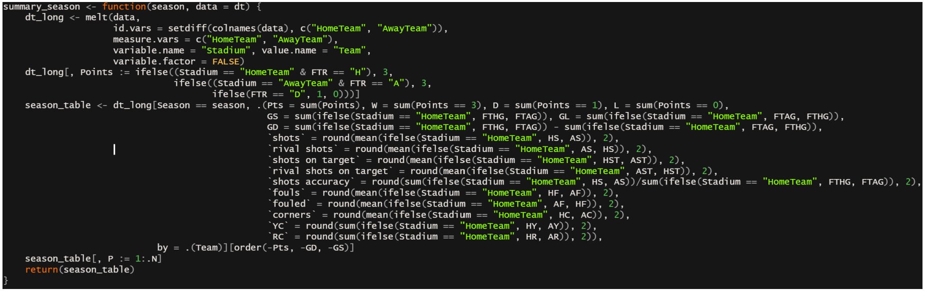
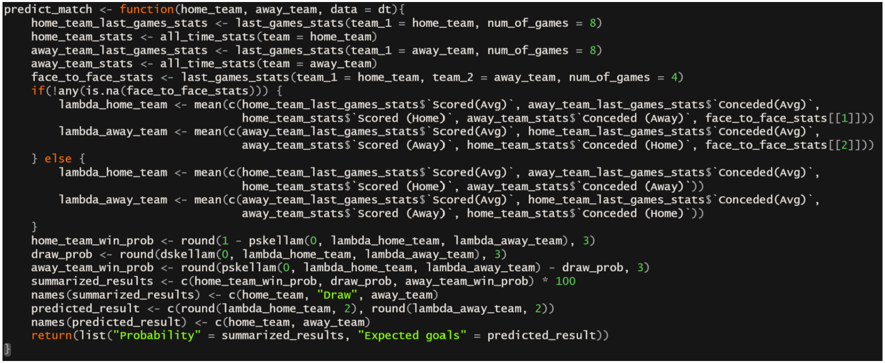
Premier League

Stanisław Banaszek, Mateusz Drobina, Dominik Mika, Adrian Płoszczyca, Jakub Sobkowiak

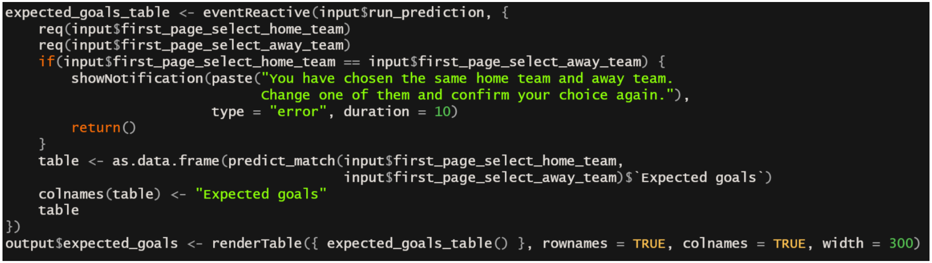
1 lutego 2021

Premier League

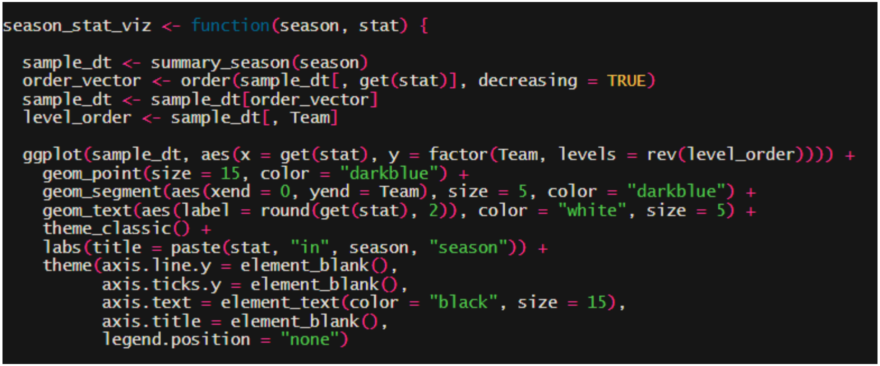

```r
summary_season <- function(season, data = dt) {
    dt_long <- melt(data,
                    id.vars = setdiff(colnames(data), c("HomeTeam", "AwayTeam")),
                    measure.vars = c("HomeTeam", "AwayTeam"),
                    variable.name = "Stadium", value.name = "Team",
                    variable.factor = FALSE)
    dt_long[, Points := ifelse((Stadium == "HomeTeam" & FTR == "H"), 3,
                               ifelse((Stadium == "AwayTeam" & FTR == "A"), 3,
                                      ifelse(FTR == "D", 1, 0)))]
    season_table <- dt_long[Season == season, .(Pts = sum(Points), W = sum(Points == 3), D = sum(Points == 1), L = sum(Points == 0),
                                                GS = sum(ifelse(Stadium == "HomeTeam", FTHG, FTAG)), GL = sum(ifelse(Stadium == "HomeTeam", FTAG, FTHG)),
                                                GD = sum(ifelse(Stadium == "HomeTeam", FTHG, FTAG)) - sum(ifelse(Stadium == "HomeTeam", FTAG, FTHG)),
                                                `shots` = round(mean(ifelse(Stadium == "HomeTeam", HS, AS)), 2),
                                                `rival shots` = round(mean(ifelse(Stadium == "HomeTeam", AS, HS)), 2),
                                                `shots on target` = round(mean(ifelse(Stadium == "HomeTeam", HST, AST)), 2),
                                                `rival shots on target` = round(mean(ifelse(Stadium == "HomeTeam", AST, HST)), 2),
                                                `shots accuracy` = round(sum(ifelse(Stadium == "HomeTeam", HS, AS))/sum(ifelse(Stadium == "HomeTeam", FTHG, FTAG)), 2),
                                                `fouls` = round(mean(ifelse(Stadium == "HomeTeam", HF, AF)), 2),
                                                `fouled` = round(mean(ifelse(Stadium == "HomeTeam", AF, HF)), 2),
                                                `corners` = round(mean(ifelse(Stadium == "HomeTeam", HC, AC)), 2),
                                                `YC` = round(sum(ifelse(Stadium == "HomeTeam", HY, AY)), 2),
                                                `RC` = round(sum(ifelse(Stadium == "HomeTeam", HR, AR)), 2)),
                            by = .(Team)][order(-Pts, -GD, -GS)]
    season_table[, P := 1:.N]
    return(season_table)
}
```



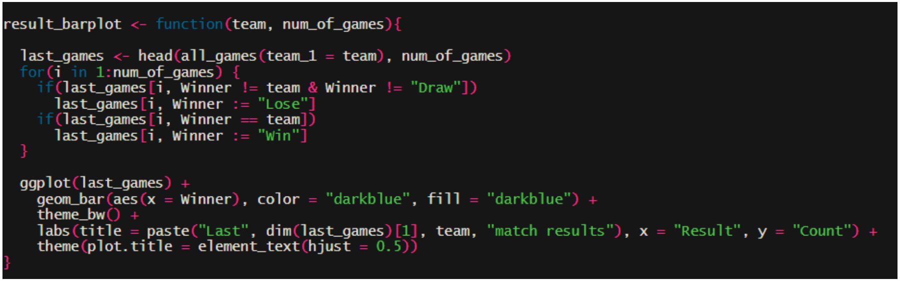
```r
predict_match <- function(home_team, away_team, data = dt){
    home_team_last_games_stats <- last_games_stats(team_1 = home_team, num_of_games = 8)
    home_team_stats <- all_time_stats(team = home_team)
    away_team_last_games_stats <- last_games_stats(team_1 = away_team, num_of_games = 8)
    away_team_stats <- all_time_stats(team = away_team)
    face_to_face_stats <- last_games_stats(team_1 = home_team, team_2 = away_team, num_of_games = 4)
    if(!any(is.na(face_to_face_stats))) {
        lambda_home_team <- mean(c(home_team_last_games_stats$`Scored(Avg)`, away_team_last_games_stats$`Conceded(Avg)`,
                                   home_team_stats$`Scored (Home)`, away_team_stats$`Conceded (Away)`, face_to_face_stats[[1]]))
        lambda_away_team <- mean(c(away_team_last_games_stats$`Scored(Avg)`, home_team_last_games_stats$`Conceded(Avg)`,
                                   away_team_stats$`Scored (Away)`, home_team_stats$`Conceded (Home)`, face_to_face_stats[[2]]))
    } else {
        lambda_home_team <- mean(c(home_team_last_games_stats$`Scored(Avg)`, away_team_last_games_stats$`Conceded(Avg)`,
                                   home_team_stats$`Scored (Home)`, away_team_stats$`Conceded (Away)`))
        lambda_away_team <- mean(c(away_team_last_games_stats$`Scored(Avg)`, home_team_last_games_stats$`Conceded(Avg)`,
                                   away_team_stats$`Scored (Away)`, home_team_stats$`Conceded (Home)`))
    }
    home_team_win_prob <- round(1 - pskellam(0, lambda_home_team, lambda_away_team), 3)
    draw_prob <- round(dskellam(0, lambda_home_team, lambda_away_team), 3)
    away_team_win_prob <- round(pskellam(0, lambda_home_team, lambda_away_team) - draw_prob, 3)
    summarized_results <- c(home_team_win_prob, draw_prob, away_team_win_prob) * 100
    names(summarized_results) <- c(home_team, "Draw", away_team)
    predicted_result <- c(round(lambda_home_team, 2), round(lambda_away_team, 2))
    names(predicted_result) <- c(home_team, away_team)
    return(list("Probability" = summarized_results, "Expected goals" = predicted_result))
}
```

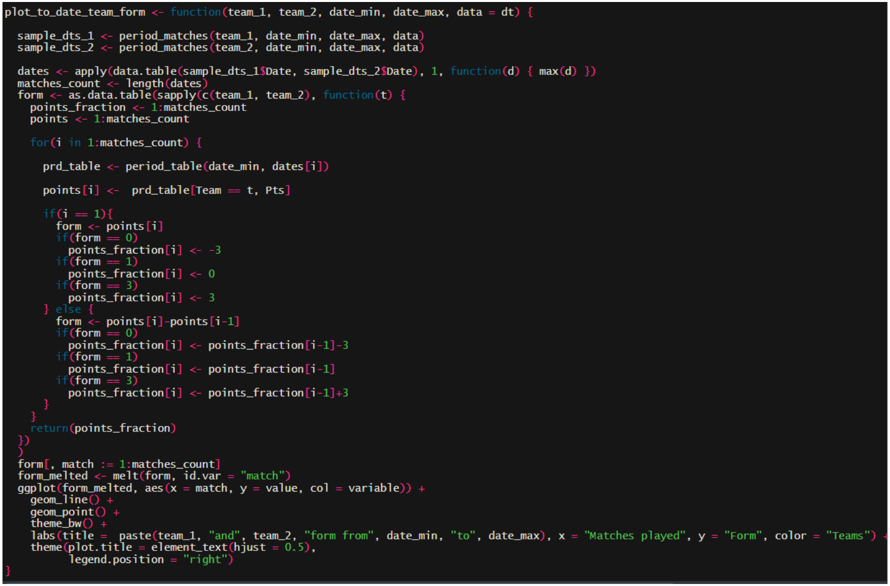


```r
expected_goals_table <- eventReactive(input$run_prediction, {
    req(input$first_page_select_home_team)
    req(input$first_page_select_away_team)
    if(input$first_page_select_home_team == input$first_page_select_away_team) {
        showNotification(paste("You have chosen the same home team and away team.
                               Change one of them and confirm your choice again."),
                         type = "error", duration = 10)
        return()
    }
    table <- as.data.frame(predict_match(input$first_page_select_home_team,
                                         input$first_page_select_away_team)$`Expected goals`)
    colnames(table) <- "Expected goals"
    table
})
output$expected_goals <- renderTable({ expected_goals_table() }, rownames = TRUE, colnames = TRUE, width = 300)
```

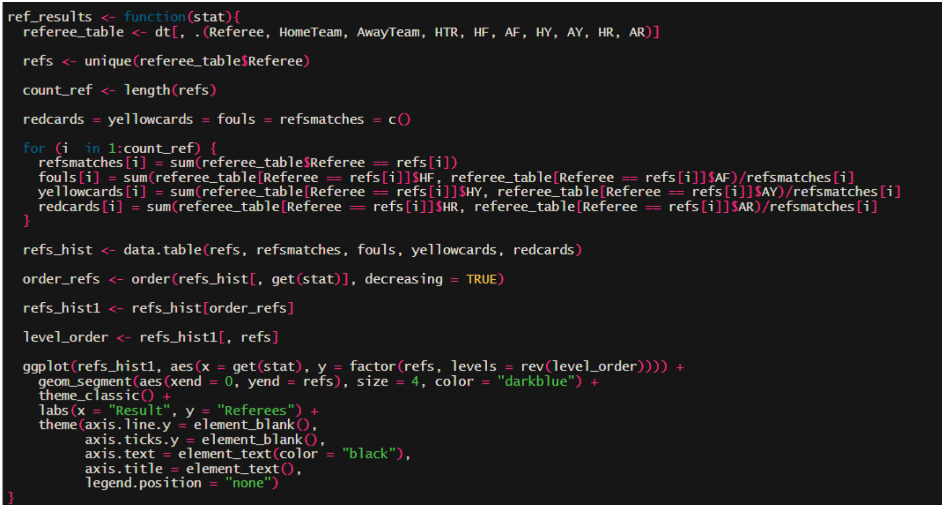


```
season_stat_viz <- function(season, stat) {

  sample_dt <- summary_season(season)
  order_vector <- order(sample_dt[, get(stat)], decreasing = TRUE)
  sample_dt <- sample_dt[order_vector]
  level_order <- sample_dt[, Team]

  ggplot(sample_dt, aes(x = get(stat), y = factor(Team, levels = rev(level_order)))) +
    geom_point(size = 15, color = "darkblue") +
    geom_segment(aes(xend = 0, yend = Team), size = 5, color = "darkblue") +
    geom_text(aes(label = round(get(stat), 2)), color = "white", size = 5) +
    theme_classic() +
    labs(title = paste(stat, "in", season, "season")) +
    theme(axis.line.y = element_blank(),
          axis.ticks.y = element_blank(),
          axis.text = element_text(color = "black", size = 15),
          axis.title = element_blank(),
          legend.position = "none")
```

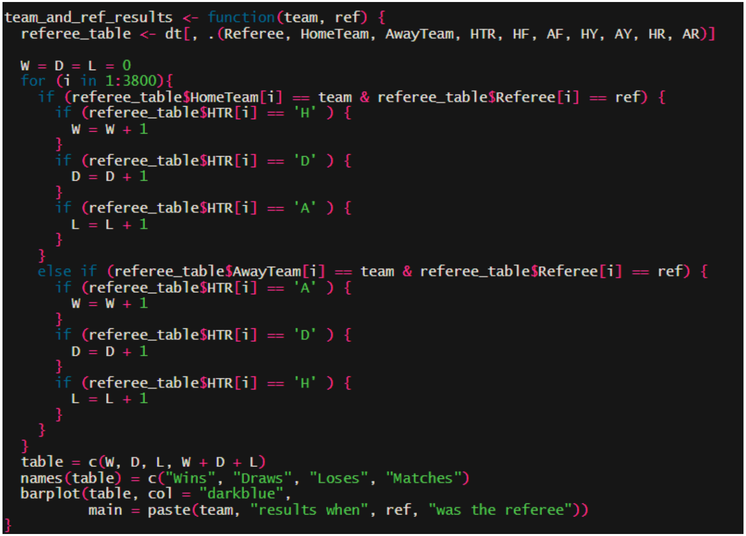

```r
result_barplot <- function(team, num_of_games){

  last_games <- head(all_games(team_1 = team), num_of_games)
  for(i in 1:num_of_games) {
    if(last_games[i, Winner != team & Winner != "Draw"])
      last_games[i, Winner := "Lose"]
    if(last_games[i, Winner == team])
      last_games[i, Winner := "Win"]
  }

  ggplot(last_games) +
    geom_bar(aes(x = Winner), color = "darkblue", fill = "darkblue") +
    theme_bw() +
    labs(title = paste("Last", dim(last_games)[1], team, "match results"), x = "Result", y = "Count") +
    theme(plot.title = element_text(hjust = 0.5))
}
```



```r
plot_to_date_team_form <- function(team_1, team_2, date_min, date_max, data = dt) {

  sample_dts_1 <- period_matches(team_1, date_min, date_max, data)
  sample_dts_2 <- period_matches(team_2, date_min, date_max, data)

  dates <- apply(data.table(sample_dts_1$Date, sample_dts_2$Date), 1, function(d) { max(d) })
  matches_count <- length(dates)
  form <- as.data.table(sapply(c(team_1, team_2), function(t) {
    points_fraction <- 1:matches_count
    points <- 1:matches_count

    for(i in 1:matches_count) {

      prd_table <- period_table(date_min, dates[i])

      points[i] <-  prd_table[Team == t, Pts]

      if(i == 1){
        form <- points[i]
        if(form == 0)
          points_fraction[i] <- -3
        if(form == 1)
          points_fraction[i] <- 0
        if(form == 3)
          points_fraction[i] <- 3
      } else {
        form <- points[i]-points[i-1]
        if(form == 0)
          points_fraction[i] <- points_fraction[i-1]-3
        if(form == 1)
          points_fraction[i] <- points_fraction[i-1]
        if(form == 3)
          points_fraction[i] <- points_fraction[i-1]+3
      }
    }
    return(points_fraction)
  })
  )
  form[, match := 1:matches_count]
  form_melted <- melt(form, id.var = "match")
  ggplot(form_melted, aes(x = match, y = value, col = variable)) +
    geom_line() +
    geom_point() +
    theme_bw() +
    labs(title =  paste(team_1, "and", team_2, "form from", date_min, "to", date_max), x = "Matches played", y = "Form", color = "Teams") +
    theme(plot.title = element_text(hjust = 0.5),
          legend.position = "right")
}
```



```r
ref_results <- function(stat){
  referee_table <- dt[, .(Referee, HomeTeam, AwayTeam, HTR, HF, AF, HY, AY, HR, AR)]

  refs <- unique(referee_table$Referee)

  count_ref <- length(refs)

  redcards = yellowcards = fouls = refsmatches = c()

  for (i  in 1:count_ref) {
    refsmatches[i] = sum(referee_table$Referee == refs[i])
    fouls[i] = sum(referee_table[Referee == refs[i]]$HF, referee_table[Referee == refs[i]]$AF)/refsmatches[i]
    yellowcards[i] = sum(referee_table[Referee == refs[i]]$HY, referee_table[Referee == refs[i]]$AY)/refsmatches[i]
    redcards[i] = sum(referee_table[Referee == refs[i]]$HR, referee_table[Referee == refs[i]]$AR)/refsmatches[i]
  }

  refs_hist <- data.table(refs, refsmatches, fouls, yellowcards, redcards)

  order_refs <- order(refs_hist[, get(stat)], decreasing = TRUE)

  refs_hist1 <- refs_hist[order_refs]

  level_order <- refs_hist1[, refs]

  ggplot(refs_hist1, aes(x = get(stat), y = factor(refs, levels = rev(level_order)))) +
    geom_segment(aes(xend = 0, yend = refs), size = 4, color = "darkblue") +
    theme_classic() +
    labs(x = "Result", y = "Referees") +
    theme(axis.line.y = element_blank(),
          axis.ticks.y = element_blank(),
          axis.text = element_text(color = "black"),
          axis.title = element_text(),
          legend.position = "none")
}
```



```r
team_and_ref_results <- function(team, ref) {
  referee_table <- dt[, .(Referee, HomeTeam, AwayTeam, HTR, HF, AF, HY, AY, HR, AR)]

  W = D = L = 0
  for (i in 1:3800){
    if (referee_table$HomeTeam[i] == team & referee_table$Referee[i] == ref) {
      if (referee_table$HTR[i] == 'H' ) {
        W = W + 1
      }
      if (referee_table$HTR[i] == 'D' ) {
        D = D + 1
      }
      if (referee_table$HTR[i] == 'A' ) {
        L = L + 1
      }
    }
    else if (referee_table$AwayTeam[i] == team & referee_table$Referee[i] == ref) {
      if (referee_table$HTR[i] == 'A' ) {
        W = W + 1
      }
      if (referee_table$HTR[i] == 'D' ) {
        D = D + 1
      }
      if (referee_table$HTR[i] == 'H' ) {
        L = L + 1
      }
    }
  }
  table = c(W, D, L, W + D + L)
  names(table) = c("Wins", "Draws", "Loses", "Matches")
  barplot(table, col = "darkblue",
          main = paste(team, "results when", ref, "was the referee"))
}
```