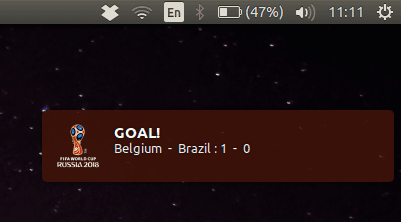
Here in Belgium, World Cup fever is at fever pitch, but with matches starting  
during work hours, how is a desk worker supposed to follow along? By leaving  
the R environment? Blasphemy.

Today we show how to use R to generate live desktop notifications for The  
Beautiful Game.



A notification system preview, free of local bias.

**Overview**

We break the process of producing a live score notification into the following  
steps:

1. Get the score
2. Check if the score has changed
3. If yes, show a notification
4. Repeat steps 1-3 every minute

**Step 1: Getting the Score**

FIFA [provides an API](https://api.fifa.com/api/v1/live/football/)  
with detailed information about matches. The API provides a list of each day’s  
matches in JSON. A full description of the fields is provided in the [API  
documentation](https://api.qa.fifa.com/Help/ResourceModel?modelName=LiveMatch).

For the purposes of this exercise, we need the scores  
(AggregateHomeTeamScore, AggregateAwayTeamScore, HomeTeamPenaltyScore,  
AwayTeamPenaltyScore), and team names (HomeTeam.TeamName,  
AwayTeam.TeamName).  
Additionally, we subset the data to the active World Cup matches by filtering to  
matches with IdSeason of 254645 (the World Cup competition  
ID) and MatchStatus of 3 (the live match status ID).

As functions, this looks like:

readLiveMatchScore <- function() {

# reading in the API data

worldcupDataDF <-

jsonlite::fromJSON("https://api.fifa.com/api/v1/live/football/")$Results

# which World Cup match is currently active?

worldcupMatchIdx <- which(worldcupDataDF$IdSeason == 254645 &

worldcupDataDF$MatchStatus == 3)

if (length(worldcupMatchIdx) != 1) { # no matches or more than 1 match

liveScore <- NULL

} else {

# get the score

liveScore <- unlist(worldcupDataDF[worldcupMatchIdx,

c("AggregateHomeTeamScore", "AggregateAwayTeamScore",

"HomeTeamPenaltyScore", "AwayTeamPenaltyScore")])

homeTeam <- worldcupDataDF$HomeTeam$TeamName[[worldcupMatchIdx]]$Description

awayTeam <- worldcupDataDF$AwayTeam$TeamName[[worldcupMatchIdx]]$Description

names(liveScore) <- rep(c(homeTeam, awayTeam), 2)

}

liveScore

}

scoreAsString <- function(matchScore, penalties = FALSE) {

out <- paste(names(matchScore)[1], " - ", names(matchScore)[2], ":",

matchScore[1], " - ", matchScore[2])

if (penalties && matchScore[1] == matchScore[2])

out <- paste0(out, " (pen. ", matchScore[3], " - ", matchScore[4], ")" )

out

}

**Step 2: Check If the Score Has Changes**

To check if the score has changed, we store the previous score and  
check if it differs from the current score. If there is a change, we send a  
notification.

checkScoreAndNotify <- function(prevScore = NULL) {

newScore <- readLiveMatchScore()

if (is.null(newScore) && is.null(prevScore)) {

# nothing to do here

} else if (is.null(newScore) && !is.null(prevScore)) {

# end of the game

sendNotification(title = "Match ended", message = scoreAsString(prevScore, TRUE))

} else if (is.null(prevScore) && !is.null(newScore)) {

# start of the game

sendNotification(title = "Match started", message = scoreAsString(newScore))

} else if (!is.null(prevScore) && !is.null(newScore) && !identical(newScore, prevScore)) {

# change in the score

sendNotification(title = "GOAL!", message = scoreAsString(newScore))

}

return(newScore)

}

**Step 3: Display Notification**

To create a notification, we use the notifier R  
package (now archived on CRAN).  
It can be installed via devtools:

devtools::install\_version("notifier")

or via the CRAN Archive by giving the URL:

url <- "https://cran.rstudio.com/src/contrib/Archive/notifier/notifier\_1.0.0.tar.gz"

install.packages(url, type = "source", repos = NULL)

To spice up the notification, we add the World Cup logo in the  
notification area.

# get the logo from FIFA website

download.file("https://api.fifa.com/api/v1/picture/tournaments-sq-4/254645\_w",

"logo.png")

sendNotification <- function(title = "", message) {

notifier::notify(title = title, msg = message, image = normalizePath("logo.png"))

}

**Step 4: Repeat Every Minute**

We use the later package to query the scores API repeatedly without blocking the R session. we write a recursive function to query the scores. The previous score is tracked using a global variable.

getScoreUpdates <- function() {

prevScore <<- checkScoreAndNotify(prevScore)

later::later(getScoreUpdates, delay = 60)

}

**All Together Now**

To run this entire process, we simply initialize the  
global prevScore variable and launch the recursive function  
getScoreUpdates:

prevScore <- NULL

getScoreUpdates()

**Complete script**

## 0. preparatory steps

if (!require("notifier", character.only = TRUE)) {

url <- "https://cloud.r-project.org/src/contrib/Archive/notifier/notifier\_1.0.0.tar.gz"

install.packages(url, type = "source", repos = NULL)

}

if (!require("later", character.only = TRUE)) {

install.packages("later")

}

download.file("https://api.fifa.com/api/v1/picture/tournaments-sq-4/254645\_w", "logo.png")

## 1. get match score

readLiveMatchScore <- function() {

# reading in the API data

worldcupDataDF <-

jsonlite::fromJSON("https://api.fifa.com/api/v1/live/football/")$Results

# which World Cup match is currently active?

worldcupMatchIdx <- which(worldcupDataDF$IdSeason == 254645 &

worldcupDataDF$MatchStatus == 3)

if (length(worldcupMatchIdx) != 1) { # no matches or more than 1 match

liveScore <- NULL

} else {

# get the score

liveScore <- unlist(worldcupDataDF[worldcupMatchIdx,

c("AggregateHomeTeamScore", "AggregateAwayTeamScore",

"HomeTeamPenaltyScore", "AwayTeamPenaltyScore")])

homeTeam <- worldcupDataDF$HomeTeam$TeamName[[worldcupMatchIdx]]$Description

awayTeam <- worldcupDataDF$AwayTeam$TeamName[[worldcupMatchIdx]]$Description

names(liveScore) <- rep(c(homeTeam, awayTeam), 2)

}

liveScore

}

scoreAsString <- function(matchScore, penalties = FALSE) {

out <- paste(names(matchScore)[1], " - ", names(matchScore)[2], ":",

matchScore[1], " - ", matchScore[2])

if (penalties && matchScore[1] == matchScore[2])

out <- paste0(out, " (pen. ", matchScore[3], " - ", matchScore[4], ")" )

out

}

## 2. check for score changes

checkScoreAndNotify <- function(prevScore = NULL) {

newScore <- readLiveMatchScore()

if (is.null(newScore) && is.null(prevScore)) {

# nothing to do here

} else if (is.null(newScore) && !is.null(prevScore)) {

# end of the game

sendNotification(title = "Match ended", message = scoreAsString(prevScore, TRUE))

} else if (is.null(prevScore) && !is.null(newScore)) {

# start of the game

sendNotification(title = "Match started", message = scoreAsString(newScore))

} else if (!is.null(prevScore) && !is.null(newScore) && !identical(newScore, prevScore)) {

# change in the score

sendNotification(title = "GOAL!", message = scoreAsString(newScore))

}

return(newScore)

}

## 3. send notification

sendNotification <- function(title = "", message) {

notifier::notify(title = title, msg = message, image = normalizePath("logo.png"))

}

## 4. check score every minute

getScoreUpdates <- function() {

prevScore <<- checkScoreAndNotify(prevScore)

later::later(getScoreUpdates, delay = 60)

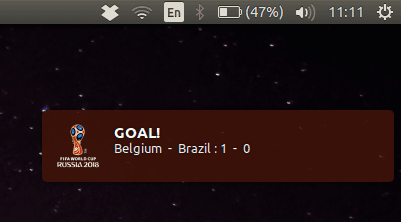
}

## 5. launch everything

prevScore <- NULL

getScoreUpdates()

**Wrap-Up**



That’s our quick take on generating live score notifications using R. By using  
a different API or alternative competition codes, this approach can be  
generalized to generate notifications for other settings.