# Define functions for testing

test\_message <- function(a){

message("this is test from function message.") return(a)

}

test\_warning <- function(a){

warning("this is test from function warning.") return(a)

}

test\_error <- function(a){

stop("this is test from function error.") return(a)

}

# Capture logs of a R function

A function is developed to capture error, warning and message into a list.

capture\_log1 <- function(f) { function(...) {

logs <- list()

add\_log <- function(type, message) { new\_l <- logs

new\_log <- list(timestamp = format(Sys.time(), tz = 'UTC', format = '%Y-%m-%d %H:%M:%S'),

type = type, message = message)

new\_l[[length(new\_l) + 1]] <- new\_log logs <<- new\_l

}

res <- withCallingHandlers( tryCatch(f(...), error=function(e) {

add\_log("error", conditionMessage(e)) NULL

}), warning=function(w) { add\_log("warning", conditionMessage(w)) invokeRestart("muffleWarning")

}, message = function(m) { add\_log("message", conditionMessage(m)) invokeRestart("muffleMessage")

})

list(res, logs = logs)

}

}

capture\_log1(test\_message)(1)

## [[1]]

## [1] 1 ##

## $logs

## $logs[[1]]

## $logs[[1]]$timestamp

## [1] "2020-10-21 06:52:52" ##

## $logs[[1]]$type ## [1] "message" ##

## $logs[[1]]$message

## [1] "this is test from function message.\n" capture\_log1(test\_warning)(1)

## [[1]]

## [1] 1 ##

## $logs

## $logs[[1]]

## $logs[[1]]$timestamp

## [1] "2020-10-21 06:52:52" ##

## $logs[[1]]$type ## [1] "warning" ##

## $logs[[1]]$message

## [1] "this is test from function warning." capture\_log1(test\_error)(1)

## [[1]] ## NULL ##

## $logs

## $logs[[1]]

## $logs[[1]]$timestamp

## [1] "2020-10-21 06:52:52" ##

## $logs[[1]]$type ## [1] "error"

##

## $logs[[1]]$message

## [1] "this is test from function error."

The only problem is the function cannot capture print and cat.

# Send logs into database through restAPI in real time

In the next step, I would like to POST logs into batabase through restAPI in real time, but

not too frequent to reduce overhead of web server (e.g. 10s as minimum time interval). In this case, all unsent logs generated by R function are cached in the memory until next POST time. However, unsent logs might be lost if the function is finished before the next POST time. A special final log, which starts with a random string (e.g. GtBRVWpNGunZRJAt), can be used to POST all unsent logs. All unsent logs are also required to POST into dataset when an error is

happening.

post\_log <- function(id, data) { # post to restAPI here

# ...

}

#' Capture log and post by restAPI #'

#' @param f A function

#' @param id The id to POST to restAPI #' @param post Whether to post message #'

#' @return A list with result of function f and all logs #' @export

capture\_log2 <- function(f, id, post = FALSE) { function(...) {

logs <- list() remain\_logs <- list() post\_time <- NULL

add\_log <- function(type, message) { new\_l <- logs

# Only post message if the time interval is more than 10 s # and contain the last message key (GtBRVWpNGunZRJAt)

# and type equals to stop is\_post <- FALSE

if (is.null(post\_time)) { is\_post <- TRUE

} else {

time\_interval <- as.numeric(Sys.time()) - as.numeric(post\_time)

if (type == 'error' | time\_interval > 10) { is\_post <- TRUE

}

}

message

if (grepl("^GtBRVWpNGunZRJAt:", message)) { is\_post <- TRUE

message <- gsub("^GtBRVWpNGunZRJAt:(.\*)", '\\1',

}

new\_log <- list(id = id,

timestamp = format(Sys.time(), tz = 'UTC', format = '%Y-%m-%d %H:%M:%S'),

type = type, message = message)

if (post) {

tryCatch({

new\_log

new\_remain\_logs <- remain\_logs new\_remain\_logs[[length(new\_remain\_logs) + 1]] <-

if (is\_post) {

# Function to post logs through restAPI post\_log(id = id,

data = new\_remain\_logs) remain\_logs <<- list()

post\_time <<- Sys.time()

} else {

remain\_logs <<- new\_remain\_logs

}

}, error = function(e) { print(e)

})

}

new\_l[[length(new\_l) + 1]] <- new\_log logs <<- new\_l

}

res <- withCallingHandlers( tryCatch(f(...), error=function(e) {

add\_log("error", conditionMessage(e)) NULL

}), warning=function(w) { add\_log("warning", conditionMessage(w)) invokeRestart("muffleWarning")

}, message = function(m) { add\_log("message", conditionMessage(m)) invokeRestart("muffleMessage")

})

list(res, logs = logs)

}

}

test\_final\_message <- function(a) { message('GtBRVWpNGunZRJAt:This is a final message')

}

capture\_log2(test\_message, 1, post = TRUE)(1) capture\_log2(test\_warning, 1, post = TRUE)(1) capture\_log2(test\_error, 1, post = TRUE)(1) capture\_log2(test\_final\_message, 1, post = TRUE)(1)