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

Following the accepted answer from this question on stackoverflow, a rewrote 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
                }
            }
            if (grepl("^GtBRVWpNGunZRJAt:", message)) {
                is_post <- TRUE
                message <- gsub("^GtBRVWpNGunZRJAt:(.*)", '\\1',
message)
            }
            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_remain_logs <- remain_logs
                    new_remain_logs[[length(new_remain_logs) + 1]]  <-
new_log
```

```
                    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)
```