**fun\_warning <- function(data, no.warn.print = FALSE, text = NULL){**

*# AIM*

*# evaluate an instruction written between "" and return the first of the warning messages if ever exist*

*# WARNING*

*# Only the first warning message is returned*

*# REQUIRED FUNCTIONS FROM CUTE\_LITTLE\_R\_FUNCTION*

*# fun\_check()*

*# ARGUMENTS*

*# data: character string to evaluate*

*# no.warn.print: logical. Print a message saying that no warning reported?*

*# text: character string added to the warning message (even if no warning exists)*

*# RETURN*

*# the warning message or NULL if no warning message*

*# EXAMPLES*

*# fun\_warning(data = "wilcox.test(c(1,1), 2:3)", no.warn.print = FALSE, text = NULL)*

*# DEBUGGING*

*# data = "wilcox.test(c(1,1), 2:3)" ; no.warn.print = FALSE ; text = NULL # for function debugging*

*# function name*

**function.name <- paste0(as.list(match.call(expand.dots=FALSE))[[1]], "()")**

*# end function name*

*# required function checking*

**if(length(find("fun\_check", mode = "function")) == 0){**

**tempo.cat <- paste0("\n\n================\n\nERROR IN ", function.name, ": REQUIRED fun\_check() FUNCTION IS MISSING IN THE R ENVIRONMENT\n\n================\n\n")**

**stop(tempo.cat)**

**}**

*# end required function checking*

*# argument checking*

*# argument checking with fun\_check()*

**arg.check <- NULL** *# for function debbuging*

**checked.arg.names <- NULL** *# for function debbuging: used by r\_debugging\_tools*

**ee <- expression(arg.check <- c(arg.check, tempo$problem) , checked.arg.names <- c(checked.arg.names, tempo$param.name))**

**tempo <- fun\_check(data = data, class = "character", length = 1, fun.name = function.name) ; eval(ee)**

**tempo <- fun\_check(data = no.warn.print, class = "logical", length = 1, fun.name = function.name) ; eval(ee)**

**if( ! is.null(text)){**

**tempo <- fun\_check(data = text, class = "character", length = 1, fun.name = function.name) ; eval(ee)**

**}**

**if(any(arg.check) == TRUE){**

**stop()** *# nothing else because print = TRUE by default in fun\_check()*

**}**

*# end argument checking with fun\_check()*

*# source("C:/Users/Gael/Documents/Git\_versions\_to\_use/debugging\_tools\_for\_r\_dev-v1.2/r\_debugging\_tools-v1.2.R") ; eval(parse(text = str\_basic\_arg\_check\_dev)) ; eval(parse(text = str\_arg\_check\_with\_fun\_check\_dev)) # activate this line and use the function (with no arguments left as NULL) to check arguments status and if they have been checked using fun\_check()*

*# end argument checking*

*# main code*

**warn.options.ini <- options()$warn**

*# last warning cannot be used because suppressWarnings() does not modify last.warning present in the base evironment (created at first warning in a new R session), or warnings() # to reset the warning history : unlockBinding("last.warning", baseenv()) ; assign("last.warning", NULL, envir = baseenv())*

**options(warn = 2)** *# put warnings and error messages into error*

**tempo.try <- try(eval(parse(text = data)), silent = TRUE)**

**if(any(grepl(x = tempo.try, pattern = "(converted from warning)"))){**

**if( ! all(class(tempo.try) == "try-error") & length(tempo.try) != 1){**

**stop(paste0("\n\n================\n\nINTERNAL CODE ERROR IN ", function.name, "\nTHE tempo.try OBJECT IS NOT OF EXPECTED CLASS \"try-error\" AND LENGTH 1:\nCLASS: ", paste0(class(tempo.try), collapse = " "), "\nLENGTH: ", paste0(length(tempo.try), collapse = " "), "\n\n================\n\n"))** *# message for developers*

**}**

**if(length(tempo.try[[1]]) > 1){**

**stop(paste0("\n\n================\n\nINTERNAL CODE ERROR IN ", function.name, "\nTHE FIRST COMPARTMENT OF tempo.try OBJECT IS NOT OF EXPECTED LENGTH 1:\n",paste(tempo.try[[1]], collapse = " "), "\n\n================\n\n"))** *# message for developers*

**}**

**tempo.try[[1]] <- gsub(x = tempo.try[[1]], pattern = "Error i", replacement = "I")**

**tempo.warning <- gsub(x = tempo.try[[1]], pattern = "\\(converted from warning\\)| \*\n \*", replacement = "")**

**output <- paste0("WARNING REPORTED (FIRST ONE ONLY)", ifelse(is.null(text), "", " "), text, ":\n", tempo.warning)** *# try() take only the first message*

**}else if(no.warn.print == TRUE){**

**output <- paste0("NO WARNING REPORTED", ifelse(is.null(text), "", " "), text)**

**}else{**

**output <- NULL**

**}**

**options(warn = warn.options.ini)** *# restore initial setting*

**return(output)**

**}**