**################################ OUTLINE ################################**

[################ CREATION OF THE CHECKING DATASET 1](#_Toc55753659)

[######## object list tl25\_20201108 (25 objects created 20201108) 1](#_Toc55753660)

[################ END CREATION OF THE CHECKING DATASET 3](#_Toc55753661)

[################ Object analysis 3](#_Toc55753662)

[######## fun\_check() #### check class, type, length, etc., of objects 3](#_Toc55753663)

[######## fun\_info() #### recover object information 4](#_Toc55753664)

[######## fun\_head() #### head of the left or right of big 2D objects 5](#_Toc55753665)

[######## fun\_tail() #### tail of the left or right of big 2D objects 5](#_Toc55753666)

[######## fun\_comp\_1d() #### comparison of two 1D datasets (vectors, factors, 1D tables) 5](#_Toc55753667)

[######## fun\_comp\_2d() #### comparison of two 2D datasets (row & col names, dimensions, etc.) 5](#_Toc55753668)

[######## fun\_comp\_list() #### comparison of two lists 5](#_Toc55753669)

[######## fun\_test() #### test combinations of argument values of a function 5](#_Toc55753670)

[################ Object modification 5](#_Toc55753671)

[######## fun\_name\_change() #### check a vector of character strings and modify any string if present in another vector 5](#_Toc55753672)

[######## fun\_df\_remod() #### remodeling a data frame to have column name as a qualitative values and vice-versa 6](#_Toc55753673)

[######## fun\_merge() #### merge the columns of two 2D objects, by common rows 6](#_Toc55753674)

[######## fun\_round() #### rounding number if decimal present 6](#_Toc55753675)

[######## fun\_mat\_rotate() #### 90° clockwise matrix rotation 6](#_Toc55753676)

[######## fun\_mat\_num2color() #### convert a numeric matrix into hexadecimal color matrix 6](#_Toc55753677)

[######## fun\_mat\_op() #### assemble several matrices with operation 6](#_Toc55753678)

[######## fun\_mat\_inv() #### return the inverse of a square matrix 6](#_Toc55753679)

[######## fun\_mat\_fill() #### fill the empty half part of a symmetric square matrix 6](#_Toc55753680)

[######## fun\_permut() #### progressively breaks a vector order 6](#_Toc55753681)

[################ Graphics management 6](#_Toc55753682)

[######## fun\_width() #### window width depending on classes to plot 6](#_Toc55753683)

[######## fun\_open() #### open a GUI or pdf graphic window 6](#_Toc55753684)

[######## fun\_prior\_plot() #### set graph param before plotting (erase axes for instance) 7](#_Toc55753685)

[######## fun\_scale() #### select nice label numbers when setting number of ticks on an axis 7](#_Toc55753686)

[######## fun\_post\_plot() #### set graph param after plotting (axes redesign for instance) 7](#_Toc55753687)

[######## fun\_close() #### close specific graphic windows 7](#_Toc55753688)

[################ Standard graphics 7](#_Toc55753689)

[######## fun\_empty\_graph() #### text to display for empty graphs 7](#_Toc55753690)

[################ gg graphics 8](#_Toc55753691)

[######## fun\_gg\_palette() #### ggplot2 default color palette 8](#_Toc55753692)

[######## fun\_gg\_just() #### ggplot2 justification of the axis labeling, depending on angle 8](#_Toc55753693)

[######## fun\_gg\_point\_rast() #### ggplot2 raster scatterplot layer 9](#_Toc55753694)

[######## fun\_gg\_scatter() #### ggplot2 scatterplot + lines (up to 6 overlays totally) 9](#_Toc55753695)

[######## fun\_gg\_bar() #### ggplot2 mean barplot + overlaid dots if required 14](#_Toc55753696)

[######## fun\_gg\_boxplot() #### ggplot2 boxplot + background dots if required 14](#_Toc55753697)

[######## fun\_gg\_prop() #### ggplot2 proportion barplot 21](#_Toc55753698)

[######## fun\_gg\_dot() #### ggplot2 categorial dotplot + mean/median 21](#_Toc55753699)

[######## fun\_gg\_violin() #### ggplot2 violins 21](#_Toc55753700)

[######## fun\_gg\_line() #### ggplot2 lines + background dots and error bars 21](#_Toc55753701)

[######## fun\_gg\_heatmap() #### ggplot2 heatmap + overlaid mask if required 21](#_Toc55753702)

[######## fun\_gg\_empty\_graph() #### text to display for empty graphs 22](#_Toc55753703)

[################ Graphic extraction 22](#_Toc55753704)

[######## fun\_trim() #### display values from a quantitative variable and trim according to defined cut-offs 22](#_Toc55753705)

[######## fun\_segmentation() #### segment a dot cloud on a scatterplot and define the dots from another cloud outside the segmentation 22](#_Toc55753706)

[################ Import 22](#_Toc55753707)

[######## fun\_pack() #### check if R packages are present and import into the working environment 22](#_Toc55753708)

[######## fun\_python\_pack() #### check if python packages are present 22](#_Toc55753709)

[################ Print / Exporting results (text & tables) 22](#_Toc55753710)

[######## fun\_report() #### print string or data object into output file 22](#_Toc55753711)

[######## fun\_get\_message() #### return messages of an expression (that can be exported) 22](#_Toc55753712)

**################################ FUNCTIONS ################################**

################ CREATION OF THE CHECKING DATASET

######## object list tl25\_20201108 (25 objects created 20201108)

**vec1 <- -1:3 # vector of integers**

**vec2 <- 1:3 / 3 # vector of proportions**

**vec3 <- c(1, 2, NA, -Inf) # vector of integers but stored as "double", with NA and Inf**

**vec4 <- "pearson" # vector of characters**

**vec5 <- c("a", "b","a", NA) # vector of characters with NA**

**cpx1 <- as.complex(1) # complex**

**mat1 <- matrix(vec1) # 1D matrix of integers**

**mat2 <- matrix(c(1:5, NA), ncol = 2, dimnames = list(c("ROW1", "ROW2", "ROW3"), c("M1", "M2"))) # 2D matrix of floats with NA**

**df1 <- as.data.frame(mat2) # data.frame**

**l1 <- list(L1 = 1:3, L2 = letters[1:3]) # list**

**fac1 <- factor(rep(letters[4:6], c(4:6))) # factor**

**tab1 <- table(fac1) # 1D table**

**tab2 <- table(fac1, fac1) # 2D table**

**exp1 <- expression("a") # object of class "expression", mode "expression" & type "expression"**

**name1 <- substitute(exp1) # object of class "name", mode "name" & type "symbol"**

**fun1 <- mean # closure function of class "function", mode "function" & type "closure"**

**fun2 <- sum # primitive function of class "function", mode "function" & type "builtin"**

**fun3 <- get("<-") # primitive function of class "function", mode "function" & type "special"**

**env1 <- new.env() # environment**

**s4.1 <- show # S4 object**

**call1 <- call("call1") # object of class "call", mode "call" & type "language"**

**tl25\_20201108 <- list( # test list**

**NULL,**

**NA,**

**1,**

**TRUE,**

**vec1,**

**vec2,**

**vec3,**

**vec4,**

**vec5,**

**cpx1,**

**mat1,**

**mat2,**

**df1,**

**l1,**

**fac1,**

**tab1,**

**tab2,**

**exp1,**

**name1,**

**fun1,**

**fun2,**

**fun3,**

**env1,**

**s4.1,**

**call1**

**)**

**save(list = c(**

**"vec1",**

**"vec2",**

**"vec3",**

**"vec4",**

**"vec5",**

**"cpx1",**

**"mat1",**

**"mat2",**

**"df1",**

**"l1",**

**"fac1",**

**"tab1",**

**"tab2",**

**"exp1",**

**"name1",**

**"fun1",**

**"fun2",**

**"fun3",**

**"env1",**

**"s4.1",**

**"call1",**

**"tl25\_20201108"**

**), file = "C:/Users/gael/Documents/Git\_projects/cute\_little\_R\_functions/other/test\_dataset.20201108.RData")**

################ END CREATION OF THE CHECKING DATASET

################ Object analysis

######## fun\_check() #### check class, type, length, etc., of objects

**# 20201108 checking each argument separately, with default values for the others**

**load("C:/Users/gael/Documents/Git\_projects/cute\_little\_R\_functions/other/test\_dataset.20201108.RData")** *# recover the test list tl file*

*# performed with test\_dataset.RData commited 20201107 (25 different objects)*

*# in green the values that are not the default values*

**for(i0 in 1:14){**

**res <- fun\_test(**

**fun = "fun\_check",**

**arg = c(**

**L1 = "data",**

**L2 = "class",**

**L3 = "typeof",**

**L4 = "mode",**

**L5 = "length",**

**L6 = "prop",**

**L7 = "double.as.integer.allowed",**

**L8 = "options",**

**L9 = "all.options.in.data",**

**L10 = "na.contain",**

**L11 = "neg.values",**

**L12 = "print",**

**L13 = "data.name",**

**L14 = "fun.name"**

**),**

**val = list(**

**L1 = if(i0 == 1){tl25\_20201108}else{"a"},**

**L2 = if(i0 == 2){tl25\_20201108}else{"vector"},**

**L3 = if(i0 == 3){tl25\_20201108}else{list(NULL)},**

**L4 = if(i0 == 4){tl25\_20201108}else{list(NULL)},**

**L5 = if(i0 == 5){tl25\_20201108}else{list(NULL)},**

**L6 = if(i0 == 6){tl25\_20201108}else{FALSE},**

**L7 = if(i0 == 7){tl25\_20201108}else{FALSE},**

**L8 = if(i0 == 8){tl25\_20201108}else{list(NULL)},**

**L9 = if(i0 == 9){tl25\_20201108}else{FALSE},**

**L10 = if(i0 == 10){tl25\_20201108}else{FALSE},**

**L11 = if(i0 == 11){tl25\_20201108}else{TRUE},**

**L12 = if(i0 == 12){tl25\_20201108}else{FALSE},**

**L13 = if(i0 == 13){tl25\_20201108}else{"test.function"},**

**L14 = if(i0 == 14){tl25\_20201108}else{list(NULL)}**

**),**

**thread.nb = NULL,**

**plot.fun = FALSE,**

**export = TRUE,**

**res.path = "C:\\Users\\Gael\\Desktop\\"**

**)**

**}**

######## fun\_info() #### recover object information

**# 20201108 checking each argument separately, with default values for the others**

**load("C:/Users/gael/Documents/Git\_projects/cute\_little\_R\_functions/other/test\_dataset.20201108.RData")** *# recover the test list tl file*

*# performed with test\_dataset.RData commited 20201107 (25 different objects)*

*# in green the values that are not the default values*

**for(i0 in 1:3){**

**res <- fun\_test(**

**fun = "fun\_info",**

**arg = c(**

*L1 = "***data***",*

*L2 = "***n***",*

*L3 = "***warn.print***"*

**),**

**val = list(**

**L1 = if(i0 == 1){tl25\_20201108}else{"a"},**

**L2 = if(i0 == 2){tl25\_20201108}else{list(NULL)},**

**L3 = if(i0 == 3){tl25\_20201108}else{TRUE}**

**),**

**expect.error = list(**

**L1 = if(i0 == 1){c(TRUE, TRUE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE)}else{FALSE},**

**L2 = if(i0 == 2){c(FALSE, TRUE, FALSE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE)}else{FALSE},**

**L3 = if(i0 == 3){c(TRUE, TRUE, TRUE, FALSE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE)}else{FALSE}**

**),**

**thread.nb = NULL,**

**plot.fun = FALSE,**

**export = TRUE,**

**res.path = "C:\\Users\\Gael\\Desktop\\"**

**)**

**}**

**# 20201108 checking each argument separately, with all the other authorized / challenging values for the others**

**load("C:/Users/gael/Documents/Git\_projects/cute\_little\_R\_functions/other/test\_dataset.20201108.RData")** *# recover the test list tl file*

*# performed with test\_dataset.RData commited 20201107 (25 different objects)*

*# in green the values that are not the default values*

**for(i0 in 1:3){**

**res <- fun\_test(**

**fun = "fun\_info",**

**arg = c(**

*L1 = "***data***",*

*L2 = "***n***",*

*L3 = "***warn.print***"*

**),**

**val = list(**

**L1 = tl25\_20201108,**

**L2 = if(i0 == 2){tl25\_20201108}else{list(NULL, -1, 0, 1, Inf)},**

**L3 = if(i0 == 3){tl25\_20201108}else{list(TRUE, FALSE, NA)}**

**),**

**expect.error = list(**

**L1 = c(TRUE, TRUE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE),**

**L2 = if(i0 == 2){c(FALSE, TRUE, FALSE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE)}else{c(FALSE, TRUE, TRUE, FALSE, FALSE)},**

**L3 = if(i0 == 3){c(TRUE, TRUE, TRUE, FALSE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE)}else{c(FALSE, FALSE, TRUE)}**

**),**

**thread.nb = 16,**

**plot.fun = FALSE,**

**export = TRUE,**

**res.path = "C:\\Users\\Gael\\Desktop\\"**

**)**

**}**

######## fun\_head() #### head of the left or right of big 2D objects

**# Check OK: clear to go Apollo**

######## fun\_tail() #### tail of the left or right of big 2D objects

**# Check OK: clear to go Apollo**

######## fun\_comp\_1d() #### comparison of two 1D datasets (vectors, factors, 1D tables)

**# Check OK: clear to go Apollo**

######## fun\_comp\_2d() #### comparison of two 2D datasets (row & col names, dimensions, etc.)

**# Check OK: clear to go Apollo**

######## fun\_comp\_list() #### comparison of two lists

**# Check OK: clear to go Apollo**

######## fun\_test() #### test combinations of argument values of a function

**# Check OK: clear to go Apollo**

################ Object modification

######## fun\_name\_change() #### check a vector of character strings and modify any string if present in another vector

**# Check OK: clear to go Apollo**

######## fun\_df\_remod() #### remodeling a data frame to have column name as a qualitative values and vice-versa

**# Check OK: clear to go Apollo**

######## fun\_merge() #### merge the columns of two 2D objects, by common rows

######## fun\_round() #### rounding number if decimal present

**# Check OK: clear to go Apollo**

######## fun\_mat\_rotate() #### 90° clockwise matrix rotation

**# Check OK: clear to go Apollo**

######## fun\_mat\_num2color() #### convert a numeric matrix into hexadecimal color matrix

**# Check OK: clear to go Apollo**

######## fun\_mat\_op() #### assemble several matrices with operation

**# Check OK: clear to go Apollo**

######## fun\_mat\_inv() #### return the inverse of a square matrix

**# Check OK: clear to go Apollo**

######## fun\_mat\_fill() #### fill the empty half part of a symmetric square matrix

**# Check OK: clear to go Apollo**

######## fun\_permut() #### progressively breaks a vector order

################ Graphics management

######## fun\_width() #### window width depending on classes to plot

**# Check OK: clear to go Apollo**

######## fun\_open() #### open a GUI or pdf graphic window

**# Check OK: clear to go Apollo**

######## fun\_prior\_plot() #### set graph param before plotting (erase axes for instance)

**# Check OK: clear to go Apollo**

######## fun\_scale() #### select nice label numbers when setting number of ticks on an axis



**# Check OK: clear to go Apollo**

######## fun\_post\_plot() #### set graph param after plotting (axes redesign for instance)

****

**# Check OK: clear to go Apollo**

######## fun\_close() #### close specific graphic windows

**# Check OK: clear to go Apollo**

################ Standard graphics

######## fun\_empty\_graph() #### text to display for empty graphs

****

**# Check OK: clear to go Apollo**

################ gg graphics

######## fun\_gg\_palette() #### ggplot2 default color palette



**# Check OK: clear to go Apollo**

######## fun\_gg\_just() #### ggplot2 justification of the axis labeling, depending on angle



**# Check OK: clear to go Apollo**

######## fun\_gg\_point\_rast() #### ggplot2 raster scatterplot layer

**# Check OK: clear to go Apollo**

######## fun\_gg\_scatter() #### ggplot2 scatterplot + lines (up to 6 overlays totally)

*# TESTS*

*# 1 tests checking the all the arguments, using a single value*

*# b <- list(NULL)*

*# b <- list("a")*

*# b <- list(NA)*

*a <- fun\_test(*

*fun = "fun\_gg\_scatter",*

*arg = c(*

*L1 = "data1",*

*L2 = "x",*

*L3 = "y",*

*L4 = "categ",*

*L5 = "legend.name",*

*L6 = "color",*

*L7 = "geom",*

*L8 = "alpha",*

*L9 = "dot.size",*

*L10 = "line.size",*

*L11 = "x.lim",*

*L12 = "x.lab",*

*L13 = "x.log",*

*L14 = "x.tick.nb",*

*L15 = "x.inter.tick.nb",*

*L16 = "x.include.zero",*

*L17 = "x.left.extra.margin",*

*L18 = "x.right.extra.margin",*

*L19 = "x.text.angle",*

*L20 = "y.lim",*

*L21 = "y.lab",*

*L22 = "y.log",*

*L23 = "y.tick.nb",*

*L24 = "y.inter.tick.nb",*

*L25 = "y.include.zero",*

*L26 = "y.top.extra.margin",*

*L27 = "y.bottom.extra.margin",*

*L28 = "y.text.angle",*

*L29 = "text.size",*

*L30 = "title",*

*L31 = "title.text.size",*

*L32 = "legend.show",*

*L33 = "article",*

*L34 = "grid",*

*L35 = "raster",*

*L36 = "raster.threshold",*

*L37 = "return",*

*L38 = "plot",*

*L39 = "add",*

*L40 = "warn.print",*

*L41 = "lib.path"*

*),*

*val = list(*

*L1 = b,*

*L2 = b,*

*L3 = b,*

*L4 = b,*

*L5 = b,*

*L6 = b,*

*L7 = b,*

*L8 = b,*

*L9 = b,*

*L10 = b,*

*L11 = b,*

*L12 = b,*

*L13 = b,*

*L14 = b,*

*L15 = b,*

*L16 = b,*

*L17 = b,*

*L18 = b,*

*L19 = b,*

*L20 = b,*

*L21 = b,*

*L22 = b,*

*L23 = b,*

*L24 = b,*

*L25 = b,*

*L26 = b,*

*L27 = b,*

*L28 = b,*

*L29 = b,*

*L30 = b,*

*L31 = b,*

*L32 = b,*

*L33 = b,*

*L34 = b,*

*L35 = b,*

*L36 = b,*

*L37 = b,*

*L38 = b,*

*L39 = b,*

*L40 = b,*

*L41 = b*

*),*

*thread.nb = NULL,*

*plot.fun = TRUE,*

*export = TRUE,*

*res.path = "C:\\Users\\Gael\\Desktop\\"*

*)*

*# 576 tests checking the critical arguments*

*set.seed(1) ; obs1 <- data.frame(km = rnorm(10, 10, 3), time = rnorm(10, 10, 3), Group1 = rep(c("A1", "A2"), 5)) ; obs1$km[2:3] <- NA*

*obs2 <-data.frame(km = rnorm(10, 15, 3), time = rnorm(10, 15, 3), Group2 = rep(c("G1", "G2"), 5)) ; set.seed(NULL)*

*a <- fun\_test(*

*fun = "fun\_gg\_scatter",*

*arg = c(*

*L1 = "data1",*

*L2 = "x",*

*L3 = "y",*

*L4 = "categ",*

*L5 = "legend.name",*

*L6 = "color",*

*L7 = "geom",*

*L8 = "alpha"*

*),*

*val = list(*

*L1 = list(L1.1 = list(L1 = obs1, L2 = obs2), L1.2 = list(obs1), L1.3 = list("a")),*

*L2 = list(L2.1 = list("km", "km"), L2.2 = list("km")),*

*L3 = list(L3.1 = list("time", "time"), L3.2 = list("time")),*

*L4 = list(L4.1 = list("Group1", "Group2"), L4.2 = list("Group1")),*

*L5 = list(L5.1 = list(NULL, NULL), L5.2 = list(NULL)),*

*L6 = list(L6.1 = list("green", "blue"), L6.2 = list(1:2), L6.3 = list(1:2, 3:4)),*

*L7 = list(L7.1 = list("geom\_point", "geom\_point"), L7.2 = list("geom\_point")),*

*L8 = list(L8.1 = list(1, 0.1), L7.2 = list(0.1))*

*),*

*thread.nb = 7,*

*plot.fun = TRUE,*

*export = TRUE,*

*res.path = "C:\\Users\\Gael\\Desktop\\"*

*)*

*# 1 test checking the data frame aspects*

*set.seed(1) ; obs1 <- data.frame(km = rnorm(10, 10, 3), time = rnorm(10, 10, 3), Group1 = rep(c("A1", "A2"), 5)) ; obs1$km[2:3] <- NA*

*obs2 <-data.frame(km = rnorm(10, 15, 3), time = rnorm(10, 15, 3), Group2 = rep(c("G1", "G2"), 5)) ; set.seed(NULL)*

*a <- fun\_test(*

*fun = "fun\_gg\_scatter",*

*arg = c(*

*L1 = "data1",*

*L2 = "x",*

*L3 = "y",*

*L4 = "categ",*

*L5 = "legend.name",*

*L6 = "color",*

*L7 = "geom",*

*L8 = "alpha"*

*),*

*val = list(*

*L1 = list(L1.1 = obs1),*

*L2 = list(L2.1 = "km"),*

*L3 = list(L3.1 = "time"),*

*L4 = list(L4.1 = "Group1"),*

*L5 = list(L5.1 = NULL, "LEGEND"),*

*L6 = list(L6.1 = c("green", "blue"), L6.2 = c("green")),*

*L7 = list(L7.1 = "geom\_point", L7.2 = "geom\_line" , L7.3 = "geom\_path"),*

*L8 = list(L8.1 = 0.25, L8.2 = 0.5, L8.3 = 1)*

*),*

*thread.nb = NULL,*

*plot.fun = TRUE,*

*export = TRUE,*

*res.path = "C:\\Users\\Gael\\Desktop\\"*

*)*

*# 1 tests checking the list aspects*

*set.seed(1) ; obs1 <- data.frame(km = rnorm(10, 10, 3), time = rnorm(10, 10, 3), Group1 = rep(c("A1", "A2"), 5)) ; obs1$km[2:3] <- NA*

*obs2 <-data.frame(km = rnorm(10, 15, 3), time = rnorm(10, 15, 3), Group2 = rep(c("G1", "G2"), 5)) ; set.seed(NULL)*

*a <- fun\_test(*

*fun = "fun\_gg\_scatter",*

*arg = c(*

*L1 = "data1",*

*L2 = "x",*

*L3 = "y",*

*L4 = "categ",*

*L5 = "legend.name",*

*L6 = "color",*

*L7 = "geom",*

*L8 = "alpha"*

*),*

*val = list(*

*L1 = list(L1.1 = list(L1 = obs1, L2 = obs2)),*

*L2 = list(L2.1 = list("km", "km")),*

*L3 = list(L3.1 = list("time", "time")),*

*L4 = list(L4.1 = list("Group1", "Group2")),*

*L5 = list(L5.1 = list("LEG1", "LEG2")),*

*L6 = list(L6.1 = list(1:2, 3:4)),*

*L7 = list(L7.1 = list("geom\_point", "geom\_point")),*

*L8 = list(L8.1 = list(1, 0.5))*

*),*

*thread.nb = NULL,*

*plot.fun = TRUE,*

*export = TRUE,*

*res.path = "C:\\Users\\Gael\\Desktop\\"*

*)*

*# 16384 tests checking the data frame aspects*

*set.seed(1) ; obs1 <- data.frame(km = rnorm(10, 10, 3), time = rnorm(10, 10, 3), Group1 = rep(c("A1", "A2"), 5)) ; obs1$km[2:3] <- NA*

*obs2 <-data.frame(km = rnorm(10, 15, 3), time = rnorm(10, 15, 3), Group2 = rep(c("G1", "G2"), 5)) ; set.seed(NULL)*

*a <- fun\_test(*

*fun = "fun\_gg\_scatter",*

*arg = c(*

*L1 = "data1",*

*L2 = "x",*

*L3 = "y",*

*L4 = "categ",*

*L5 = "legend.name",*

*L6 = "color",*

*L7 = "geom",*

*L8 = "alpha",*

*L9 = "dot.size",*

*L10 = "line.size",*

*L11 = "x.lim",*

*L12 = "x.lab",*

*L13 = "x.log",*

*L14 = "x.tick.nb",*

*L15 = "x.inter.tick.nb",*

*L16 = "x.include.zero",*

*L17 = "x.left.extra.margin",*

*L18 = "x.right.extra.margin",*

*L19 = "x.text.angle"*

*),*

*val = list(*

*L1 = list(L1.1 = obs1),*

*L2 = list(L2.1 = "km"),*

*L3 = list(L3.1 = "time"),*

*L4 = list(L4.1 = "Group1"),*

*L5 = list(L5.1 = NULL, "LEGEND"),*

*L6 = list(L6.1 = c("green", "blue"), L6.2 = 1),*

*L7 = list(L7.1 = "geom\_point"),*

*L8 = list(L8.1 = 0.25, L8.2 = 1),*

*L9 = list(L9.1 = 2, L9.2 = 0),*

*L10 = list(L10.1 = 2, L10.2 = 0),*

*L11 = list(L11.1 = NULL, L11.2 = c(1, 20)),*

*L12 = list(L12.1 = NULL, L12.2 = "XLAB"),*

*L13 = list(L13.1 = "no", L13.2 = "log10"),*

*L14 = list(L14.1 = NULL, L14.2 = 6),*

*L15 = list(L15.1 = NULL, L15.2 = 1),*

*L16 = list(L16.1 = TRUE, L16.2 = FALSE),*

*L17 = list(L17.1 = 0, L17.2 = 0.2),*

*L18 = list(L18.1 = 0, L18.2 = 0.3),*

*L19 = list(L19.1 = 0, L19.2 = 120)*

*),*

*thread.nb = NULL,*

*plot.fun = TRUE,*

*export = TRUE,*

*res.path = "C:\\Users\\Gael\\Desktop\\"*

*)*

*# Matrix tests checking the data frame aspects*

*set.seed(1) ; obs1 <- data.frame(km = rnorm(10, 10, 3), time = rnorm(10, 10, 3), Group1 = rep(c("A1", "A2"), 5)) ; obs1$km[2:3] <- NA*

*obs2 <-data.frame(km = rnorm(10, 15, 3), time = rnorm(10, 15, 3), Group2 = rep(c("G1", "G2"), 5)) ; set.seed(NULL)*

*a <- fun\_test(*

*fun = "fun\_gg\_scatter",*

*arg = c(*

*L1 = "data1",*

*L2 = "x",*

*L3 = "y",*

*L4 = "categ",*

*L5 = "legend.name",*

*L6 = "color",*

*L7 = "geom",*

*L8 = "alpha",*

*L9 = "dot.size",*

*L10 = "line.size",*

*L11 = "x.lim",*

*L12 = "x.lab",*

*L13 = "x.log",*

*L14 = "x.tick.nb",*

*L15 = "x.inter.tick.nb",*

*L16 = "x.include.zero",*

*L17 = "x.left.extra.margin",*

*L18 = "x.right.extra.margin",*

*L19 = "x.text.angle",*

*L20 = "y.lim",*

*L21 = "y.lab",*

*L22 = "y.log",*

*L23 = "y.tick.nb",*

*L24 = "y.inter.tick.nb",*

*L25 = "y.include.zero",*

*L26 = "y.top.extra.margin",*

*L27 = "y.bottom.extra.margin",*

*L28 = "y.text.angle",*

*L29 = "text.size",*

*L30 = "title",*

*L31 = "title.text.size",*

*L32 = "show.legend",*

*L33 = "article",*

*L34 = "grid",*

*L35 = "raster",*

*L36 = "vectorial.limit",*

*L37 = "return",*

*L38 = "plot",*

*L39 = "add",*

*L40 = "warn.print"*

*),*

*val = list(*

*L1 = list(L1.1 = obs1),*

*L2 = list(L2.1 = "km"),*

*L3 = list(L3.1 = "time"),*

*L4 = list(L4.1 = "Group1"),*

*L5 = list(L5.1 = NULL, "LEGEND"),*

*L6 = list(L6.1 = c("green", "blue"), L6.2 = 1),*

*L7 = list(L7.1 = "geom\_point"),*

*L8 = list(L8.1 = 0.25, L8.2 = 1),*

*L9 = list(L9.1 = 2, L9.2 = 0),*

*L10 = list(L10.1 = 2, L10.2 = 0),*

*L11 = list(L11.1 = NULL, L11.2 = c(1, 20)),*

*L12 = list(L12.1 = NULL, L12.2 = "XLAB"),*

*L13 = list(L13.1 = "no", L13.2 = "log10"),*

*L14 = list(L14.1 = NULL, L14.2 = 6),*

*L15 = list(L15.1 = NULL, L15.2 = 1),*

*L16 = list(L16.1 = TRUE, L16.2 = FALSE),*

*L17 = list(L17.1 = 0, L17.2 = 0.2),*

*L18 = list(L18.1 = 0, L18.2 = 0.3),*

*L19 = list(L19.1 = 0, L19.2 = 120),*

*L20 = list(L20.1 = NULL, L20.2 = c(1, 20)),*

*L21 = list(L21.1 = NULL, L21.2 = "YLAB"),*

*L22 = list(L22.1 = "no", L22.2 = "log10"),*

*L23 = list(L23.1 = NULL, L23.2 = 7),*

*L24 = list(L24.1 = NULL, L24.2 = 3),*

*L25 = list(L25.1 = TRUE, L25.2 = FALSE),*

*L26 = list(L26.1 = 0, L26.2 = 0.2),*

*L27 = list(L27.1 = 0, L27.2 = 0.3),*

*L28 = list(L28.1 = 0, L28.2 = 270),*

*L29 = list(L29.1 = 0, L29.2 = 15),*

*L30 = list(L30.1 = NULL, L30.2 = "TITLE"),*

*L31 = list(L31.1 = 0, L31.2 = 20),*

*L32 = list(L32.1 = TRUE, L32.2 = FALSE),*

*L33 = list(L33.1 = TRUE, L33.2 = FALSE),*

*L34 = list(L34.1 = TRUE, L34.2 = FALSE),*

*L35 = list(L35.1 = TRUE, L35.2 = FALSE),*

*L36 = list(L36.1 = NULL, L36.2 = 5),*

*L37 = list(L37.1 = TRUE, L37.2 = FALSE),*

*L38 = list(L38.1 = TRUE, L38.2 = FALSE),*

*L39 = list(L39.1 = NULL, L39.2 = "+ggplot2::theme\_dark()"),*

*L40 = list(L40.1 = TRUE, L40.2 = FALSE)*

*),*

*thread.nb = 8,*

*plot.fun = TRUE,*

*export = TRUE,*

*res.path = "C:\\Users\\Gael\\Desktop\\"*

*)*

######## fun\_gg\_bar() #### ggplot2 mean barplot + overlaid dots if required

######## fun\_gg\_boxplot() #### ggplot2 boxplot + background dots if required

*# TESTS*

*# 1 tests checking the all the arguments, using a single value*

*# b <- list(NULL)*

*# b <- list("a")*

*# b <- list(NA, NA)*

*a <- fun\_test(*

*fun = "fun\_gg\_boxplot",*

*arg = c(*

*# L1 = "data1",*

*# L2 = "y",*

*# L3 = "categ",*

*# L4 = "categ.class.order",*

*# L5 = "categ.legend.name",*

*# L6 = "categ.color",*

*# L7 = "box.fill",*

*L8 = "box.width",*

*L9 = "box.space",*

*L10 = "box.line.size",*

*L11 = "box.notch",*

*L12 = "box.alpha",*

*L13 = "box.mean",*

*L14 = "box.whisker.kind",*

*L15 = "box.whisker.width",*

*L16 = "dot.color",*

*L17 = "dot.categ",*

*L18 = "dot.categ.class.order",*

*L19 = "dot.categ.legend.name",*

*L20 = "dot.tidy",*

*L21 = "dot.tidy.bin.nb",*

*L22 = "dot.jitter",*

*L23 = "dot.size",*

*L24 = "dot.alpha",*

*L25 = "dot.border.size",*

*L26 = "dot.border.color",*

*L27 = "x.lab",*

*L28 = "y.lab",*

*L29 = "y.lim",*

*L30 = "y.log",*

*L31 = "y.tick.nb",*

*L32 = "y.inter.tick.nb",*

*L33 = "y.include.zero",*

*L34 = "y.top.extra.margin",*

*L35 = "y.bottom.extra.margin",*

*L36 = "stat.disp",*

*L37 = "stat.disp.mean",*

*L38 = "stat.size",*

*L39 = "stat.dist",*

*L40 = "vertical",*

*L41 = "text.size",*

*L42 = "text.angle",*

*L43 = "title",*

*L44 = "title.text.size",*

*L45 = "article",*

*L46 = "grid",*

*L47 = "return",*

*L48 = "plot",*

*L49 = "add",*

*L50 = "warn.print",*

*L51 = "lib.path"*

*),*

*val = list(*

*# L1 = b,*

*# L2 = b,*

*# L3 = b,*

*# L4 = b,*

*# L5 = b,*

*# L6 = b,*

*# L7 = b,*

*L8 = b,*

*L9 = b,*

*L10 = b,*

*L11 = b,*

*L12 = b,*

*L13 = b,*

*L14 = b,*

*L15 = b,*

*L16 = b,*

*L17 = b,*

*L18 = b,*

*L19 = b,*

*L20 = b,*

*L21 = b,*

*L22 = b,*

*L23 = b,*

*L24 = b,*

*L25 = b,*

*L26 = b,*

*L27 = b,*

*L28 = b,*

*L29 = b,*

*L30 = b,*

*L31 = b,*

*L32 = b,*

*L33 = b,*

*L34 = b,*

*L35 = b,*

*L36 = b,*

*L37 = b,*

*L38 = b,*

*L39 = b,*

*L40 = b,*

*L41 = b,*

*L42 = b,*

*L43 = b,*

*L44 = b,*

*L45 = b,*

*L46 = b,*

*L47 = b,*

*L48 = b,*

*L49 = b,*

*L50 = b,*

*L51 = b*

*),*

*thread.nb = NULL,*

*plot.fun = TRUE,*

*export = TRUE,*

*res.path = "C:\\Users\\Gael\\Desktop\\"*

*)*

*# 36 tests checking the critical arguments*

*set.seed(1) ; obs1 <- data.frame(Time = c(rnorm(10), rnorm(10) + 2), Group1 = rep(c("G", "H"), each = 10)) ; set.seed(NULL)*

*a <- fun\_test(*

*fun = "fun\_gg\_boxplot",*

*arg = c(*

*L1 = "data1",*

*L2 = "y",*

*L3 = "categ"*

*),*

*val = list(*

*L1 = list(L1.1 = NULL, L1.2 = obs1, L1.3 = "a"),*

*L2 = list(L2.1 = NULL, L2.2 = "Time", L2.3 = list(data.frame())),*

*L3 = list(L3.1 = NULL, L3.2 = "Group1", L3.3 = c("Group1", "Group2"), L3.4 = list(data.frame()))*

*),*

*thread.nb = NULL,*

*plot.fun = TRUE,*

*export = FALSE,*

*res.path = "C:\\Users\\Gael\\Desktop\\"*

*)*

*# 960 tests checking the box and dot colors with tidy dots & NA removing classes*

*set.seed(1) ; obs1 <- data.frame(Time = c(rnorm(10), rnorm(10) + 2), Group1 = rep(c("G", "H"), each = 10), Group2 = rep(c("A", "B"), time = 10)) ; set.seed(NULL) ; obs1$Time[1:11] <- NA*

*a <- fun\_test(*

*fun = "fun\_gg\_boxplot",*

*arg = c(*

*L1 = "data1",*

*L2 = "y",*

*L3 = "categ",*

*L4 = "categ.class.order",*

*L5 = "categ.color",*

*L6 = "dot.color",*

*L7 = "dot.categ",*

*L8 = "dot.categ.class.order",*

*L9 = "dot.categ.legend.name"*

*),*

*val = list(*

*L1 = list(L1.1 = obs1),*

*L2 = list(L2.1 = "Time"),*

*L3 = list(L3.1 = "Group1", L3.2 = c("Group1", "Group2")),*

*L4 = list(L4.1 = list(c("G", "H")), L4.2 = list(c("H", "G")), L4.3 = list(c("G", "H"), c("A", "B")), L4.4 = list(c("H", "G"), c("A", "B")), L4.5 = list(c("H", "G"), c("B", "A"))),*

*L5 = list(L5.1 = NULL, L5.2 = "green", L5.3 = c("blue", "green"), L5.4 = c("green", "blue")),*

*L6 = list(L6.1 = "same", L6.2 = NULL, L6.3= "black", L6.4 = c("red", "brown")),*

*L7 = list(L7.1 = c("Group1")),*

*L8 = list(L7.1 = NULL, L7.2 = c("G", "H"), L7.3 = c("H", "G")),*

*L9 = list(L9.1 = NULL, L9.2 = "DOT1")*

*),*

*thread.nb = 7,*

*plot.fun = TRUE,*

*res.path = "C:\\Users\\Gael\\Desktop\\",*

*export = TRUE*

*)*

*# 960 tests checking the box and dot colors with jitter dots & WITHOUT NA*

*set.seed(1) ; obs1 <- data.frame(Time = c(rnorm(10), rnorm(10) + 2), Group1 = rep(c("G", "H"), each = 10), Group2 = rep(c("A", "B"), time = 10)) ; set.seed(NULL)*

*a <- fun\_test(*

*fun = "fun\_gg\_boxplot",*

*arg = c(*

*L1 = "data1",*

*L2 = "y",*

*L3 = "categ",*

*L4 = "categ.class.order",*

*L5 = "categ.color",*

*L6 = "dot.color",*

*L7 = "dot.categ",*

*L8 = "dot.categ.class.order",*

*L9 = "dot.categ.legend.name",*

*L10 = "dot.tidy"*

*),*

*val = list(*

*L1 = list(L1.1 = obs1),*

*L2 = list(L2.1 = "Time"),*

*L3 = list(L3.1 = "Group1", L3.2 = c("Group1", "Group2")),*

*L4 = list(L4.1 = list(c("G", "H")), L4.2 = list(c("H", "G")), L4.3 = list(c("G", "H"), c("A", "B")), L4.4 = list(c("H", "G"), c("A", "B")), L4.5 = list(c("H", "G"), c("B", "A"))),*

*L5 = list(L5.1 = NULL, L5.2 = "green", L5.3 = c("blue", "green"), L5.4 = c("green", "blue")),*

*L6 = list(L6.1 = "same", L6.2 = NULL, L6.3= "black", L6.4 = c("red", "brown")),*

*L7 = list(L7.1 = c("Group1")),*

*L8 = list(L7.1 = NULL, L7.2 = c("G", "H"), L7.3 = c("H", "G")),*

*L9 = list(L9.1 = NULL, L9.2 = "DOT1"),*

*L10 = list(L10.1 = FALSE)*

*),*

*thread.nb = 7,*

*plot.fun = TRUE,*

*res.path = "C:\\Users\\Gael\\Desktop\\",*

*export = TRUE*

*)*

*# 512 tests checking the box and dot colors with tidy dots & NA removing classes*

*set.seed(1) ; obs1 <- data.frame(Time = c(rnorm(10), rnorm(10) + 2), Group1 = rep(c("G", "H"), each = 10), Group2 = rep(c("A", "B"), time = 10)) ; set.seed(NULL) ; obs1$Time[1:11] <- NA*

*b <- list(TRUE, FALSE)*

*z <- list(0, 0.5)*

*y <- list(NULL, "TEST\_LEG")*

*a <- fun\_test(*

*fun = "fun\_gg\_boxplot",*

*arg = c(*

*L1 = "data1",*

*L2 = "y",*

*L3 = "categ",*

*# L4 = "categ.class.order",*

*# L5 = "categ.legend.name",*

*# L6 = "categ.color",*

*L7 = "box.fill",*

*L8 = "box.width",*

*L9 = "box.space",*

*L10 = "box.line.size",*

*L11 = "box.notch",*

*L12 = "box.alpha",*

*L13 = "box.mean",*

*L14 = "box.whisker.kind",*

*L15 = "box.whisker.width"*

*# L16 = "dot.color",*

*# L17 = "dot.categ",*

*# L18 = "dot.categ.class.order",*

*# L19 = "dot.categ.legend.name",*

*# L20 = "dot.tidy",*

*# L21 = "dot.tidy.bin.nb",*

*# L22 = "dot.jitter",*

*# L23 = "dot.size",*

*# L24 = "dot.alpha",*

*# L25 = "dot.border.size",*

*# L26 = "dot.border.color",*

*# L27 = "x.lab",*

*# L28 = "y.lab",*

*# L29 = "y.lim",*

*# L30 = "y.log"*

*# L31 = "y.tick.nb",*

*# L32 = "y.inter.tick.nb",*

*# L33 = "y.include.zero",*

*# L34 = "y.top.extra.margin",*

*# L35 = "y.bottom.extra.margin",*

*# L36 = "stat.disp",*

*# L37 = "stat.disp.mean",*

*# L38 = "stat.size",*

*# L39 = "stat.dist",*

*# L40 = "vertical",*

*# L41 = "text.size",*

*# L42 = "text.angle",*

*# L43 = "title",*

*# L44 = "title.text.size",*

*# L45 = "article",*

*# L46 = "grid",*

*# L47 = "return",*

*# L48 = "plot",*

*# L49 = "add",*

*# L50 = "warn.print"*

*# L51 = "lib.path"*

*),*

*val = list(*

*L1 = list(obs1),*

*L2 = list("Time"),*

*L3 = list(c("Group1", "Group2")),*

*# L4 = b,*

*# L5 = b,*

*# L6 = b,*

*L7 = b,*

*L8 = z,*

*L9 = z,*

*L10 =z,*

*L11 = b,*

*L12 = z,*

*L13 = b,*

*L14 = list("std", "no"),*

*L15 = z*

*# L16 = b,*

*# L17 = b,*

*# L18 = b,*

*# L19 = b,*

*# L20 = b,*

*# L21 = c(5, 100),*

*# L22 = z,*

*# L23 = z,*

*# L24 = z,*

*# L25 = z,*

*# L26 = "blue",*

*# L27 = y,*

*# L28 = y,*

*# L29 = list(c(2, 10)),*

*# L30 = list("no", "log10")*

*# L31 = list(6),*

*# L32 = list(4),*

*# L33 = b,*

*# L34 = list(0, 0.1),*

*# L35 = list(0, 0.1),*

*# L36 = list(NULL, "above"),*

*# L37 = b,*

*# L38 = list(10),*

*# L39 = 1,*

*# L40 = b,*

*# L41 = list(10),*

*# L42 = list(120),*

*# L43 = "TITLE\_TEST",*

*# L44 = list(20),*

*# L45 = b,*

*# L46 = b,*

*# L47 = b,*

*# L48 = b,*

*# L49 = list(NULL),*

*# L50 = FALSE*

*# L51 = b*

*),*

*thread.nb = 8,*

*plot.fun = TRUE,*

*res.path = "C:\\Users\\Gael\\Desktop\\",*

*export = TRUE*

*)*

*# 512 tests checking the box and dot colors with tidy dots & NA removing classes*

*set.seed(1) ; obs1 <- data.frame(Time = c(rnorm(10), rnorm(10) + 2), Group1 = rep(c("G", "H"), each = 10), Group2 = rep(c("A", "B"), time = 10)) ; set.seed(NULL) ; obs1$Time[1:11] <- NA*

*b <- list(TRUE, FALSE)*

*z <- list(0, 0.5)*

*y <- list(NULL, "TEST\_LEG")*

*a <- fun\_test(*

*fun = "fun\_gg\_boxplot",*

*arg = c(*

*L1 = "data1",*

*L2 = "y",*

*L3 = "categ",*

*# L4 = "categ.class.order",*

*# L5 = "categ.legend.name",*

*# L6 = "categ.color",*

*# L7 = "box.fill",*

*# L8 = "box.width",*

*# L9 = "box.space",*

*# L10 = "box.line.size",*

*# L11 = "box.notch",*

*# L12 = "box.alpha",*

*# L13 = "box.mean",*

*# L14 = "box.whisker.kind",*

*# L15 = "box.whisker.width"*

*# L16 = "dot.color",*

*# L17 = "dot.categ",*

*# L18 = "dot.categ.class.order",*

*# L19 = "dot.categ.legend.name",*

*L20 = "dot.tidy",*

*# L21 = "dot.tidy.bin.nb",*

*L22 = "dot.jitter",*

*L23 = "dot.size",*

*L24 = "dot.alpha",*

*L25 = "dot.border.size",*

*L26 = "dot.border.color",*

*L27 = "x.lab",*

*L28 = "y.lab",*

*L29 = "y.lim",*

*L30 = "y.log",*

*L31 = "y.tick.nb",*

*L32 = "y.inter.tick.nb",*

*L33 = "y.include.zero"*

*# L34 = "y.top.extra.margin",*

*# L35 = "y.bottom.extra.margin",*

*# L36 = "stat.disp",*

*# L37 = "stat.disp.mean",*

*# L38 = "stat.size",*

*# L39 = "stat.dist",*

*# L40 = "vertical",*

*# L41 = "text.size",*

*# L42 = "text.angle",*

*# L43 = "title",*

*# L44 = "title.text.size",*

*# L45 = "article",*

*# L46 = "grid",*

*# L47 = "return",*

*# L48 = "plot",*

*# L49 = "add",*

*# L50 = "warn.print"*

*# L51 = "lib.path"*

*),*

*val = list(*

*L1 = list(obs1),*

*L2 = list("Time"),*

*L3 = list(c("Group1", "Group2")),*

*# L4 = b,*

*# L5 = b,*

*# L6 = b,*

*# L7 = b,*

*# L8 = z,*

*# L9 = z,*

*# L10 =z,*

*# L11 = b,*

*# L12 = z,*

*# L13 = b,*

*# L14 = list("std", "no"),*

*# L15 = z*

*# L16 = b,*

*# L17 = b,*

*# L18 = b,*

*# L19 = b,*

*L20 = b,*

*# L21 = c(5, 100),*

*L22 = z,*

*L23 = z,*

*L24 = z,*

*L25 = z,*

*L26 = "blue",*

*L27 = y,*

*L28 = y,*

*L29 = list(c(2, 10)),*

*L30 = list("no", "log10"),*

*L31 = list(6),*

*L32 = list(4),*

*L33 = b*

*# L34 = list(0, 0.1),*

*# L35 = list(0, 0.1),*

*# L36 = list(NULL, "above"),*

*# L37 = b,*

*# L38 = list(10),*

*# L39 = 1,*

*# L40 = b,*

*# L41 = list(10),*

*# L42 = list(120),*

*# L43 = "TITLE\_TEST",*

*# L44 = list(20),*

*# L45 = b,*

*# L46 = b,*

*# L47 = b,*

*# L48 = b,*

*# L49 = list(NULL),*

*# L50 = FALSE*

*# L51 = b*

*),*

*thread.nb = 8,*

*plot.fun = TRUE,*

*res.path = "C:\\Users\\Gael\\Desktop\\",*

*export = TRUE*

*)*

######## fun\_gg\_prop() #### ggplot2 proportion barplot

######## fun\_gg\_dot() #### ggplot2 categorial dotplot + mean/median

######## fun\_gg\_violin() #### ggplot2 violins

######## fun\_gg\_line() #### ggplot2 lines + background dots and error bars

######## fun\_gg\_heatmap() #### ggplot2 heatmap + overlaid mask if required

**# Check OK: clear to go Apollo**

######## fun\_gg\_empty\_graph() #### text to display for empty graphs



**# Check OK: clear to go Apollo**

################ Graphic extraction

######## fun\_trim() #### display values from a quantitative variable and trim according to defined cut-offs

**# Check OK: clear to go Apollo**

######## fun\_segmentation() #### segment a dot cloud on a scatterplot and define the dots from another cloud outside the segmentation

**# Check OK: clear to go Apollo**

################ Import

######## fun\_pack() #### check if R packages are present and import into the working environment

**# Check OK: clear to go Apollo**

######## fun\_python\_pack() #### check if python packages are present

**# Check OK: clear to go Apollo**

################ Print / Exporting results (text & tables)

######## fun\_report() #### print string or data object into output file

**# Check OK: clear to go Apollo**

######## fun\_get\_message() #### return messages of an expression (that can be exported)

**# Check OK: clear to go Apollo**