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

[################ Object analysis 1](#_Toc33526677)

[######## fun\_check() #### check class, type, length, etc., of objects 1](#_Toc33526678)

[######## fun\_info() #### recover object information 2](#_Toc33526679)

[######## fun\_head() #### head of the left or right of big 2D objects 2](#_Toc33526680)

[######## fun\_tail() #### tail of the left or right of big 2D objects 2](#_Toc33526681)

[######## fun\_comp\_1d() #### comparison of two 1D datasets (vectors, factors, 1D tables) 2](#_Toc33526682)

[######## fun\_comp\_2d() #### comparison of two 2D datasets (row & col names, dimensions, etc.) 2](#_Toc33526683)

[######## fun\_comp\_list() #### comparison of two lists 2](#_Toc33526684)

[######## fun\_test() #### test combinations of argument values of a function 2](#_Toc33526685)

[################ Object modification 2](#_Toc33526686)

[######## fun\_name\_change() #### check a vector of character strings and modify any string if present in another vector 2](#_Toc33526687)

[######## fun\_df\_remod() #### remodeling a data frame to have column name as a qualitative values and vice-versa 2](#_Toc33526688)

[######## fun\_merge() #### merge the columns of two 2D objects, by common rows 2](#_Toc33526689)

[######## fun\_round() #### rounding number if decimal present 2](#_Toc33526690)

[######## fun\_mat\_rotate() #### 90° clockwise matrix rotation 3](#_Toc33526691)

[######## fun\_mat\_num2color() #### convert a numeric matrix into hexadecimal color matrix 3](#_Toc33526692)

[######## fun\_mat\_op() #### assemble several matrices with operation 3](#_Toc33526693)

[######## fun\_mat\_inv() #### return the inverse of a square matrix 3](#_Toc33526694)

[######## fun\_mat\_fill() #### fill the empty half part of a symmetric square matrix 3](#_Toc33526695)

[######## fun\_permut() #### progressively breaks a vector order 3](#_Toc33526696)

[################ Graphics management 3](#_Toc33526697)

[######## fun\_width() #### window width depending on classes to plot 3](#_Toc33526698)

[######## fun\_open() #### open a GUI or pdf graphic window 3](#_Toc33526699)

[######## fun\_prior\_plot() #### set graph param before plotting (erase axes for instance) 3](#_Toc33526700)

[######## fun\_scale() #### select nice label numbers when setting number of ticks on an axis 4](#_Toc33526701)

[######## fun\_post\_plot() #### set graph param after plotting (axes redesign for instance) 4](#_Toc33526702)

[######## fun\_close() #### close specific graphic windows 4](#_Toc33526703)

[################ Standard graphics 4](#_Toc33526704)

[######## fun\_empty\_graph() #### text to display for empty graphs 4](#_Toc33526705)

[################ gg graphics 5](#_Toc33526706)

[######## fun\_gg\_palette() #### ggplot2 default color palette 5](#_Toc33526707)

[######## fun\_gg\_just() #### ggplot2 justification of the axis labeling, depending on angle 5](#_Toc33526708)

[######## fun\_gg\_point\_rast() #### ggplot2 raster scatterplot layer 6](#_Toc33526709)

[######## fun\_gg\_scatter() #### ggplot2 scatterplot + lines (up to 6 overlays totally) 6](#_Toc33526710)

[######## fun\_gg\_bar() #### ggplot2 mean barplot + overlaid dots if required 6](#_Toc33526711)

[######## fun\_gg\_boxplot() #### ggplot2 boxplot + background dots if required 6](#_Toc33526712)

[######## fun\_gg\_prop() #### ggplot2 proportion barplot 9](#_Toc33526713)

[######## fun\_gg\_dot() #### ggplot2 categorial dotplot + mean/median 9](#_Toc33526714)

[######## fun\_gg\_violin() #### ggplot2 violins 9](#_Toc33526715)

[######## fun\_gg\_line() #### ggplot2 lines + background dots and error bars 9](#_Toc33526716)

[######## fun\_gg\_heatmap() #### ggplot2 heatmap + overlaid mask if required 9](#_Toc33526717)

[######## fun\_gg\_empty\_graph() #### text to display for empty graphs 9](#_Toc33526718)

[################ Graphic extraction 10](#_Toc33526719)

[######## fun\_trim() #### display values from a quantitative variable and trim according to defined cut-offs 10](#_Toc33526720)

[######## fun\_segmentation() #### segment a dot cloud on a scatterplot and define the dots from another cloud outside the segmentation 10](#_Toc33526721)

[################ Import 10](#_Toc33526722)

[######## fun\_pack() #### check if R packages are present and import into the working environment 10](#_Toc33526723)

[######## fun\_python\_pack() #### check if python packages are present 10](#_Toc33526724)

[################ Print / Exporting results (text & tables) 10](#_Toc33526725)

[######## fun\_report() #### print string or data object into output file 10](#_Toc33526726)

[######## fun\_get\_message() #### return messages of an expression (that can be exported) 10](#_Toc33526727)

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

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

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

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

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

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

######## 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 = "show.legend",*

*L33 = "classic",*

*L34 = "grid",*

*L35 = "raster",*

*L36 = "vectorial.limit",*

*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, 0.5), L7.2 = list(0.5))*

*),*

*thread.nb = NULL,*

*plot.fun = TRUE,*

*export = TRUE,*

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

*)*

*# 576 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"*

*),*

*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),*

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

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

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

*),*

*thread.nb = NULL,*

*plot.fun = TRUE,*

*export = TRUE,*

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

*)*

*# 576 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\\"*

*)*

######## 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 = "classic",*

*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 = TRUE,*

*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(20), rnorm(20) + 2), Group1 = rep(c("G", "H"), each = 10), Group2 = rep(c("A", "B"), time = 10)) ; set.seed(NULL) ; obs1$Time[1:21] <- 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 tidy dots & WITHOUT NA*

*set.seed(1) ; obs1 <- data.frame(Time = c(rnorm(20), rnorm(20) + 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"*

*),*

*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*

*)*

######## 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**