

# Package ‘clickR’

October 12, 2022

**Type** Package

**Title** Semi-Automatic Preprocessing of Messy Data with Change Tracking  
for Dataset Cleaning

**Version** 0.8.3

**Author** David Hervas Marin

**Maintainer** David Hervas Marin <ddhervas@yahoo.es>

**Imports** beeswarm, methods, stringdist

**Description** Tools for assessing data quality, performing exploratory analysis, and  
semi-automatic preprocessing of messy data with change tracking for integral dataset cleaning.

**License** GPL (>= 2)

**Encoding** UTF-8

**RoxygenNote** 7.2.0

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2022-09-24 21:30:02 UTC

## R topics documented:

antimoda . . . . .	2
bivariate_outliers . . . . .	3
check_quality . . . . .	3
cluster_var . . . . .	4
descriptive . . . . .	5
extreme_values . . . . .	6
fix_all . . . . .	6
fix_concat . . . . .	7
fix_dates . . . . .	7
fix_factors . . . . .	8
fix_levels . . . . .	9
fix_NA . . . . .	10
fix_numerics . . . . .	10
forge . . . . .	11

fxd . . . . .	12
f_replace . . . . .	12
GK_assoc . . . . .	13
good2go . . . . .	14
ipboxplot . . . . .	14
kill.factors . . . . .	15
kurtosis . . . . .	15
manual_fix . . . . .	16
may.numeric . . . . .	16
mine.plot . . . . .	17
moda . . . . .	17
moda_cont . . . . .	18
mtapply . . . . .	18
nearest . . . . .	19
nice_names . . . . .	19
numeros . . . . .	20
outliers . . . . .	20
peek . . . . .	21
prop_may . . . . .	21
prop_min . . . . .	22
remove_empty . . . . .	22
restore_changes . . . . .	23
scale_01 . . . . .	23
search_scripts . . . . .	24
skewness . . . . .	24
text_date . . . . .	25
track_changes . . . . .	25
ttrue . . . . .	26
unforge . . . . .	26
workspace . . . . .	27
workspace_sapply . . . . .	28
%>NA% . . . . .	28
%>=NA% . . . . .	29
%<NA% . . . . .	29
%<=NA% . . . . .	30

**Index** **31**

---

antimoda	<i>Get anti-mode</i>
----------	----------------------

---

**Description**

Returns the least repeated value

**Usage**

antimoda(x)

**Arguments**

x                    A categorical variable

**Value**

The anti-mode (least repeated value)

---

bivariate\_outliers      *Check for bivariate outliers*

---

**Description**

Checks for bivariate outliers in a data.frame

**Usage**

```
bivariate_outliers(x, threshold_r = 10, threshold_b = 1.5)
```

**Arguments**

x                    A data.frame object  
threshold\_r        Threshold for the case of two continuous variables  
threshold\_b        Threshold for the case of one continuous and one categorical variable

**Value**

A data frame with all the observations considered as bivariate outliers

**Examples**

```
bivariate_outliers(iris)
```

---

check\_quality            *Checks data quality of a variable*

---

**Description**

Returns different data quality details of a numeric or categorical variable

**Usage**

```

check_quality(
  x,
  id = 1:length(x),
  plot = TRUE,
  numeric = NULL,
  n = ifelse(is.numeric(x) | ttrue(numeric) | class(x) %in% "Date", 5, 2),
  output = FALSE,
  ...
)

```

**Arguments**

x	A variable from a data.frame
id	ID column to reference the found extreme values
plot	If the variable is numeric, should a boxplot be drawn?
numeric	If set to TRUE, forces the variable to be considered numeric
n	Number of extreme values to extract
output	Format of the output. If TRUE, optimize for exporting as csv
...	further arguments passed to boxplot()

**Value**

A list of a data.frame with information about data quality of the variable

**Examples**

```

check_quality(airquality$Ozone) #For one variable
lapply(airquality, check_quality) #For a data.frame
lapply(airquality, check_quality, output=TRUE) #For a data.frame, one row per variable

```

---

cluster\_var

*Clustering of variables*


---

**Description**

Displays associations between variables in a data.frame in a heatmap with clustering

**Usage**

```
cluster_var(x, margins = c(8, 1))
```

**Arguments**

x	A data.frame
margins	Margins for the plot

**Value**

A heatmap with the variable associations

**Examples**

```
cluster_var(iris)
cluster_var(mtcars)
```

---

descriptive

*Detailed summary of the data*

---

**Description**

Creates a detailed summary of the data

**Usage**

```
descriptive(x, z = 3, ignore.na = TRUE, by = NULL, print = TRUE)
```

**Arguments**

x	A data.frame
z	Number of decimal places
ignore.na	If TRUE NA values will not count for relative frequencies calculations
by	Factor variable defining groups for the summary
print	Should results be printed?

**Value**

Summary of the data

**Examples**

```
descriptive(iris)
descriptive(iris, by="Species")
```

---

extreme_values	<i>Extreme values from a numeric vector</i>
----------------	---------------------------------------------

---

**Description**

Returns the nth lowest and highest values from a vector

**Usage**

```
extreme_values(x, n = 5, id = NULL)
```

**Arguments**

x	A vector
n	Number of extreme values to return
id	ID column to reference the found extreme values

**Value**

A matrix with the lowest and highest values from a vector

---

fix_all	<i>fix_all</i>
---------	----------------

---

**Description**

Tries to automatically fix all problems in the data.frame

**Usage**

```
fix_all(x, select = 1:ncol(x), track = TRUE)
```

**Arguments**

x	A data.frame
select	Numeric vector with the positions (all by default) to be affected by the function
track	Track changes?

---

fix_concat	<i>fix_concat</i>
------------	-------------------

---

### Description

Fixes concatenated values in a variable

### Usage

```
fix_concat(x, varname, sep = ", |; | ", track = TRUE)
```

### Arguments

x	A data.frame
varname	Variable name
sep	Separator for the different values
track	Track changes?

### Examples

```
mydata <- data.frame(concat=c("a", "b", "a b" , "a b, c", "a; c"),
  numeric = c(1, 2, 3, 4, 5))
fix_concat(mydata, "concat")
```

---

fix_dates	<i>Fix_dates</i>
-----------	------------------

---

### Description

Fixes dates. Dates can be recorded in numerous formats depending on the country, the traditions and the field of knowledge. `fix_dates` tries to detect all possible date formats and transforms all of them in the ISO standard favored by R (yyyy-mm-dd).

### Usage

```
fix_dates(
  x,
  max.NA = 0.8,
  min.obs = nrow(x) * 0.05,
  use.probs = TRUE,
  select = 1:ncol(x),
  track = TRUE
)
```

**Arguments**

x	A data.frame
max.NA	Maximum allowed proportion of NA values created by coercion. If the coercion to date creates more NA values than those specified in max.NA, then all changes will be reverted and the variable will remain unchanged.
min.obs	Minimum number of non-NA observations allowed per variable. If the variable has fewer non-NA observations, then it will be ignored by fix.dates.
use.probs	When there are multiple date formats in the same column, there can be ambiguities. For example, 04-06-2015 can be interpreted as 2015-06-04 or as 2015-04-06. If use.probs=TRUE, ambiguities will be solved by assigning to the most frequent date format in the column.
select	Numeric vector with the positions (all by default) to be affected by the function
track	Track changes?

**Examples**

```
mydata<-data.frame(Dates1=c("25/06/1983", "25-08/2014", "2001/11/01", "2008-10-01"),
                  Dates2=c("01/01/85", "04/04/1982", "07/12-2016", "September 24, 2020"),
                  Numeric1=rnorm(4))
fix_dates(mydata)
```

---

fix\_factors

*Fix factors imported as numerics*


---

**Description**

Fixes factors imported as numerics. It is usual in some fields to encode factor variables as integers. This function detects such variables and transforms them into factors. When drop=TRUE (by default) it detects multiple versions of the same levels due to different capitalization, whitespaces or non-ASCII characters.

**Usage**

```
fix_factors(x, k = 5, select = 1:ncol(x), drop = TRUE, track = TRUE)
```

**Arguments**

x	A data.frame
k	Maximum number of different numeric values to be converted to factor
select	Numeric vector with the positions (all by default) to be affected by the function
drop	Drop similar levels?
track	Keep track of changes?



**Examples**

```
# mtcars data has all variables encoded as numeric, even the factor variables.
descriptive(mtcars)
# After using fix_factors, factor variables are recognized as such.
descriptive(fix_factors(mtcars))
```

---

fix_levels	<i>Fix levels</i>
------------	-------------------

---

**Description**

Fixes levels of a factor

**Usage**

```
fix_levels(
  data,
  factor_name,
  method = "dl",
  levels = NULL,
  plot = FALSE,
  k = ifelse(!is.null(levels), length(levels), 2),
  track = TRUE,
  ...
)
```

**Arguments**

data	data.frame with the factor to fix
factor_name	Name of the factor to fix (as character)
method	Method from stringdist package to estimate distances
levels	Optional vector with the levels names. If "auto", levels are assigned based on frequency
plot	Optional: Plot cluster dendrogram?
k	Number of levels for clustering
track	Keep track of changes?
...	Further parameters passed to stringdist::stringdistmatrix function

**Examples**

```
mydata <- data.frame(factor1=factor(c("Control", "Treatment", "Tretament", "Tratment", "treatment",
  "teatment", "contr1", "cntrol", "CONTol", "not available", "na")))
fix_levels(mydata, "factor1", k=4, plot=TRUE) #Chose k to select matching levels
fix_levels(mydata, "factor1", levels=c("Control", "Treatment"), k=4)
```

---

fix_NA	<i>fix_NA</i>
--------	---------------

---

### Description

Fixes miscoded missing values

### Usage

```
fix_NA(
  x,
  na.strings = c("^$", "^ $", "\\?$", "-$", "\\.$", "NaN$", "NULL$", "N/A$"),
  track = TRUE
)
```

### Arguments

x	A data.frame
na.strings	Strings to be considered NA
track	Track changes?

### Examples

```
mydata <- data.frame(prueba = c("", NA, "A", 4, " ", "?", "-", "+"),
  casa = c("", 1, 2, 3, 4, " ", 6, 7))
fix_NA(mydata)
```

---

fix_numerics	<i>Fix numeric data</i>
--------------	-------------------------

---

### Description

Fixes numeric data. In many cases, numeric data are not recognized by R because there are data inconsistencies (wrong decimal separator, whitespaces, typos, thousand separator, etc.). `fix_numerics` detects and corrects these variables, making them numeric again.

### Usage

```
fix_numerics(x, k = 8, max.NA = 0.2, select = 1:ncol(x), track = TRUE)
```

**Arguments**

x	A data.frame
k	Minimum number of different values a variable has to have to be considered numerical
max.NA	Maximum allowed proportion of NA values created by coercion. If the coercion to numeric creates more NA values than those specified in max.NA, then all changes will be reverted and the variable will remain unchanged.
select	Numeric vector with the positions (all by default) to be affected by the function
track	Keep track of changes?

**Examples**

```
mydata<-data.frame(Numeric1=c(7.8, 9.2, "5.4e+2", 3.3, "6,8", "3..3"),
                  Numeric2=c(3.1, 1.2, "3.4s", "48,500.04 $", 7, "$ 6.4"))
descriptive(mydata)
descriptive(fix_numerics(mydata, k=5))
```

forge

*Forge***Description**

Reshapes a data frame from wide to long format

**Usage**

```
forge(data, affixes, force.fixed = NULL, var.name = "time")
```

**Arguments**

data	data.frame
affixes	Affixes for repeated measures
force.fixed	Variables with matching affix to be excluded
var.name	Name for the new created variable (repetitions)

**Examples**

```
#Data frame in wide format
df1 <- data.frame(id = 1:4, age = c(20, 30, 30, 35), score1 = c(2,2,3,4),
                 score2 = c(2,1,3,1), score3 = c(1,1,0,1))

df1
#Data frame in long format
forge(df1, affixes= c("1", "2", "3"))

#Data frame in wide format with two repeated measured variables
df2 <- data.frame(df1, var1 = c(15, 20, 16, 19), var3 = c(12, 15, 15, 17))
```

```
df2
#Missing times are filled with NAs
forge(df2, affixes = c("1", "2", "3"))

#Use of parameter force.fixed
df3 <- df2[, -7]
df3
forge(df3, affixes=c("1", "2", "3"))
forge(df3, affixes=c("1", "2", "3"), force.fixed = c("var1"))
```

---

fxd	<i>Internal function to fix_dates</i>
-----	---------------------------------------

---

### Description

Function to format dates

### Usage

```
fxd(d, use.probs = TRUE)
```

### Arguments

d	A character vector
use.probs	Solve ambiguities by similarity to the most frequent formats

---

f_replace	<i>Find and replace</i>
-----------	-------------------------

---

### Description

Searches a data.frame for a specific character string and replaces it with another one

### Usage

```
f_replace(
  x,
  string,
  replacement,
  complete = TRUE,
  select = 1:ncol(x),
  track = TRUE
)
```

**Arguments**

x	A data.frame
string	A character string to search in the data.frame
replacement	A character string to replace the old string (can be NA)
complete	If TRUE, search for complete strings only. If FALSE, search also for partial strings.
select	Numeric vector with the positions (all by default) to be affected by the function
track	Track changes?

**Examples**

```
iris2 <- f_replace(iris, "setosa", "ensata")
track_changes(iris2)
```

---

GK\_assoc

*Computes Goodman and Kruskal's tau*


---

**Description**

Returns Goodman and Kruskal's tau measure of association between two categorical variables

**Usage**

```
GK_assoc(x, y)
```

**Arguments**

x	A categorical variable
y	A categorical variable

**Value**

Goodman and Kruskal's tau

**Examples**

```
data(Infert)
GK_assoc(Infert$education, Infert$case)
GK_assoc(Infert$case, Infert$education) #Not the same
```

---

`good2go`*Good to go*

---

**Description**

Loads all libraries used in scripts inside the selected path

**Usage**

```
good2go(path = getwd(), info = TRUE, load = TRUE)
```

**Arguments**

<code>path</code>	Path where the scripts are located
<code>info</code>	List the libraries found?
<code>load</code>	Should the libraries found be loaded?

---

`ipboxplot`*Improved boxplot*

---

**Description**

Creates an improved boxplot with individual data points

**Usage**

```
ipboxplot(formula, boxwex = 0.6, ...)
```

**Arguments**

<code>formula</code>	Formula for the boxplot
<code>boxwex</code>	Width of the boxes
<code>...</code>	further arguments passed to beeswarm()

**Examples**

```
ipboxplot(Sepal.Length ~ Species, data=iris)  
ipboxplot(mpg ~ gear, data=mtcars)
```

---

kill.factors	<i>Kill factors</i>
--------------	---------------------

---

**Description**

Changes factor variables to character

**Usage**

```
kill.factors(dat, k = 10)
```

**Arguments**

dat	A data.frame
k	Maximum number of levels for factors

**Examples**

```
d <- data.frame(Letters=letters[1:20], Nums=1:20)
d$Letters
d <- kill.factors(d)
d$Letters
```

---

kurtosis	<i>Computes kurtosis</i>
----------	--------------------------

---

**Description**

Calculates kurtosis of a numeric variable

**Usage**

```
kurtosis(x)
```

**Arguments**

x	A numeric variable
---	--------------------

**Value**

kurtosis value

manual\_fix                      *Tracked manual fixes to data*

---

**Description**

Tracks manual fixes performed on a variable in a data.frame

**Usage**

```
manual_fix(data, variable, subset, newvalues = NULL)
```

**Arguments**

data	A data.frame
variable	A character string with the name of the variable to be fixed
subset	A logical expression for selecting the cases to be fixed
newvalues	New value or values that will take the cases selected by subset parameter.

**Examples**

```
iris2 <- manual_fix(iris, "Petal.Length", Petal.Length < 1.2, 0)  
track_changes(iris2)
```

---

may.numeric                      *Checks if each value might be numeric*

---

**Description**

Checks if each value from a vector might be numeric

**Usage**

```
may.numeric(x)
```

**Arguments**

x	A vector
---	----------

**Value**

A logical vector



---

mine.plot	<i>Mine plot</i>
-----------	------------------

---

**Description**

Creates a heatmap-like plot for exploring the data

**Usage**

```
mine.plot(  
  x,  
  fun = is.na,  
  spacing = 5,  
  sort = F,  
  show.x = TRUE,  
  show.y = TRUE,  
  ...  
)
```

**Arguments**

x	A data.frame
fun	A function that evaluates a vector and returns a logical vector
spacing	Numerical separation between lines at the y-axis
sort	If TRUE, variables are sorted according to their results
show.x	Should the x-axis be plotted?
show.y	Should the y-axis be plotted?
...	further arguments passed to order()

**Examples**

```
mine.plot(airquality) #Displays missing data  
mine.plot(airquality, fun=outliers) #Shows extreme values
```

---

moda	<i>Get mode</i>
------	-----------------

---

**Description**

Returns the most repeated value

**Usage**

```
moda(x)
```

**Arguments**

x                    A categorical variable

**Value**

The mode

---

moda_cont	<i>Estimates number of modes</i>
-----------	----------------------------------

---

**Description**

Estimates the number of modes

**Usage**

```
moda_cont(x)
```

**Arguments**

x                    A numeric variable

**Value**

Estimated number of modes.

---

mtapply	<i>Multiple tapply</i>
---------	------------------------

---

**Description**

Modification of the tapply function to use with data.frames. Consider using aggregate()

**Usage**

```
mtapply(x, group, fun)
```

**Arguments**

x                    A data.frame

group                Grouping variable

fun                  Function to apply by group

**Examples**

```
mtapply(mtcars, mtcars$gear, mean)
```

---

nearest	<i>Internal function for descriptive()</i>
---------	--------------------------------------------

---

**Description**

Finds positions for substitution of characters in Distribution column

**Usage**

```
nearest(x, to = seq(0, 1, length.out = 30))
```

**Arguments**

x	A numeric value between 0-1
to	Range of reference values

**Value**

The nearest position to the input value

---

nice_names	<i>Nice names</i>
------------	-------------------

---

**Description**

Changes names of a data frame to ease work with them

**Usage**

```
nice_names(x, select = 1:ncol(x), tolower = TRUE, track = TRUE)
```

**Arguments**

x	A data.frame
select	Numeric vector with the positions (all by default) to be affected by the function
tolower	Set all names to lower case?
track	Track changes?

**Value**

The input data.frame x with the fixed names

**Examples**

```
d <- data.frame('Variable 1'=NA, '% Response'=NA, ' Variable 3'=NA, check.names=FALSE)
names(d)
names(nice_names(d))
```

---

numeros	<i>Brute numeric coercion</i>
---------	-------------------------------

---

**Description**

If possible, coerces values from a vector to numeric

**Usage**

```
numeros(x)
```

**Arguments**

x	A vector
---	----------

**Value**

A numeric vector

---

outliers	<i>outliers</i>
----------	-----------------

---

**Description**

Function for detecting outliers based on the boxplot method

**Usage**

```
outliers(x, threshold = 1.5)
```

**Arguments**

x	A vector
threshold	Threshold (as multiple of the IQR) to consider an observation as outlier

**Examples**

```
outliers(iris$Petal.Length)
outliers(airquality$Ozone)
```

---

peek

*Peek*

---

### Description

Takes a peek into a data.frame returning a concise visualization about it

### Usage

```
peek(x, n = 10, which = 1:ncol(x))
```

### Arguments

x	A data.frame
n	Number of rows to include in output
which	Columns to include in output

### Examples

```
peek(iris)
```

---

prop\_may

*Gets proportion of most repeated value*

---

### Description

Returns the proportion for the most repeated value

### Usage

```
prop_may(x, ignore.na = TRUE)
```

### Arguments

x	A categorical variable
ignore.na	Should NA values be ignored for computing proportions?

### Value

A proportion

---

prop_min	<i>Gets proportion of least repeated value</i>
----------	------------------------------------------------

---

**Description**

Returns the proportion for the least repeated value

**Usage**

```
prop_min(x, ignore.na = TRUE)
```

**Arguments**

x	A categorical variable
ignore.na	Should NA values be ignored for computing proportions?

**Value**

A proportion

---

remove_empty	<i>remove_empty</i>
--------------	---------------------

---

**Description**

Removes empty rows or columns from data.frames

**Usage**

```
remove_empty(x, remove_rows = TRUE, remove_cols = TRUE, track = TRUE)
```

**Arguments**

x	A data.frame
remove_rows	Remove empty rows?
remove_cols	Remove empty columns?
track	Track changes?

**Examples**

```
mydata <- data.frame(a = c(NA, NA, NA, NA, NA), b = c(1, NA, 3, 4, 5),
  c=c(NA, NA, NA, NA, NA), d=c(4, NA, 5, 6, 3))
remove_empty(mydata)
```

---

restore_changes	<i>Restore changes</i>
-----------------	------------------------

---

**Description**

Restores original values after using a fix function

**Usage**

```
restore_changes(tracking)
```

**Arguments**

tracking            A data.frame generated by track\_changes() function

**Examples**

```
mydata<-data.frame(Dates1=c("25/06/1983", "25-08/2014", "2001/11/01", "2008-10-01"),
                   Dates2=c("01/01/85", "04/04/1982", "07/12-2016", NA),
                   Numeric1=rnorm(4))
mydata <- fix_dates(mydata)
mydata
tracking <- track_changes(mydata)
mydata_r <- restore_changes(tracking)
mydata_r
```

---

scale_01	<i>Scales data between 0 and 1</i>
----------	------------------------------------

---

**Description**

Escale data to 0-1

**Usage**

```
scale_01(x)
```

**Arguments**

x                    A numeric variable

**Value**

Scaled data

---

search_scripts	<i>Search scripts</i>
----------------	-----------------------

---

**Description**

Searches for strings in R script files

**Usage**

```
search_scripts(string, path = getwd(), recursive = TRUE)
```

**Arguments**

string	Character string to search
path	Character vector with the path name
recursive	Logical. Should the search be recursive into subdirectories?

**Value**

A list with each element being one of the files containing the search string

---

skewness	<i>Computes skewness</i>
----------	--------------------------

---

**Description**

Calculates skewness of a numeric variable

**Usage**

```
skewness(x)
```

**Arguments**

x	A numeric variable
---	--------------------

**Value**

skewness value



---

text_date	<i>Internal function for dates with text</i>
-----------	----------------------------------------------

---

**Description**

Function to transform text into dates

**Usage**

```
text_date(date, format = "%d/%Y %b")
```

**Arguments**

date	A date
format	Format of the date

---

track_changes	<i>track_changes</i>
---------------	----------------------

---

**Description**

Gets a data.frame with all the changes performed by the different fix functions

**Usage**

```
track_changes(x, subset)
```

**Arguments**

x	A data.frame
subset	Logical expression for subsetting the data.frame with the changes

**Examples**

```
mydata<-data.frame(Dates1=c("25/06/1983", "25-08/2014", "2001/11/01", "2008-10-01"),
  Dates2=c("01/01/85", "04/04/1982", "07/12-2016", NA),
  Numeric1=rnorm(4))
mydata <- fix_dates(mydata)
mydata
track_changes(mydata)
```

---

ttrue	<i>True TRUE</i>
-------	------------------

---

**Description**

Makes possible vectorized logical comparisons against NULL and NA values

**Usage**

```
ttrue(x)
```

**Arguments**

x	A logical vector
---	------------------

**Value**

A logical vector

---

unforge	<i>Un-Forge</i>
---------	-----------------

---

**Description**

Reshapes a data frame from long to wide format

**Usage**

```
unforge(data, origin, variables, prefix = origin)
```

**Arguments**

data	data.frame
origin	Character vector with variable names in data containing the values to be assigned to the different new variables
variables	Variable in data containing the variable names to be created
prefix	Vector with prefixes for the new variable names

**Examples**

```
#Data frame in wide format
df1 <- data.frame(id = 1:4, age = c(20, 30, 30, 35), score1 = c(2,2,3,4),
                  score2 = c(2,1,3,1), score3 = c(1,1,0,1))
df1
#Data frame in long format
df2 <- forge(df1, affixes= c("1", "2", "3"))
df2
#Data frame in wide format again
df3 <- unforge(df2, "score", "time", prefix="score")
```

---

workspace

*Explores global environment workspace*

---

**Description**

Returns information regarding the different objects in global environment

**Usage**

```
workspace(table = FALSE)
```

**Arguments**

table            If TRUE a table with the frequencies of each type of object is given

**Value**

A list of object names by class or a table with frequencies if table = TRUE

**Examples**

```
df1 <- data.frame(x=rnorm(10), y=rnorm(10, 1, 2))
df2 <- data.frame(x=rnorm(20), y=rnorm(20, 1, 2))
workspace(table=TRUE) #Frequency table of the different object classes
workspace() #All objects in the global object separated by class
```

---

`workspace_sapply`      *Applies a function over objects of a specific class*

---

**Description**

Applies a function over all objects of a specific class in the global environment

**Usage**

```
workspace_sapply(object_class, action = "summary")
```

**Arguments**

`object_class`      Class of the objects where the function is to be applied  
`action`              Name of the function to apply

**Value**

Results of the function

**Examples**

```
df1 <- data.frame(x=rnorm(10), y=rnorm(10, 1, 2))
df2 <- data.frame(x=rnorm(20), y=rnorm(20, 1, 2))
workspace_sapply("data.frame", "summary") #Gives a summary of each data.frame
```

---

`%>NA%`                      *greater & NA*

---

**Description**

'>' operator where NA values return FALSE

**Usage**

```
x %>NA% y
```

**Arguments**

`x`                      Vector for the left side of the operator  
`y`                      A Scalar or vector of the same length as x for the right side of the operator

**Value**

A logical vector of the same length as x

---

%>=NA%

*geq & not NA*

---

**Description**

'>=' operator where NA values return FALSE

**Usage**

x %>=NA% y

**Arguments**

- x                    Vector for the left side of the operator
- y                    A Scalar or vector of the same length as x for the right side of the operator

**Value**

A logical vector of the same length as x

---

%<NA%

*less & NA*

---

**Description**

'<' operator where NA values return FALSE

**Usage**

x %<NA% y

**Arguments**

- x                    Vector for the left side of the operator
- y                    A Scalar or vector of the same length as x for the right side of the operator

**Value**

A logical vector of the same length as x

---

%<=NA%

*leq & not NA*

---

**Description**

'<=' operator where NA values return FALSE

**Usage**

x %<=NA% y

**Arguments**

x                    Vector for the left side of the operator

y                    A Scalar or vector of the same length as x for the right side of the operator

**Value**

A logical vector of the same length as x

# Index

[%<=NA%](#), 30  
[%<NA%](#), 29  
[%>=NA%](#), 29  
[%>NA%](#), 28

[antimoda](#), 2

[bivariate\\_outliers](#), 3

[check\\_quality](#), 3  
[cluster\\_var](#), 4

[descriptive](#), 5

[extreme\\_values](#), 6

[f\\_replace](#), 12  
[fix\\_all](#), 6  
[fix\\_concat](#), 7  
[fix\\_dates](#), 7  
[fix\\_factors](#), 8  
[fix\\_levels](#), 9  
[fix\\_NA](#), 10  
[fix\\_numerics](#), 10  
[forge](#), 11  
[fxd](#), 12

[GK\\_assoc](#), 13  
[good2go](#), 14

[ipboxplot](#), 14

[kill.factors](#), 15  
[kurtosis](#), 15

[manual\\_fix](#), 16  
[may.numeric](#), 16  
[mine.plot](#), 17  
[moda](#), 17  
[moda\\_cont](#), 18  
[mtapply](#), 18

[nearest](#), 19  
[nice\\_names](#), 19  
[numeros](#), 20

[outliers](#), 20

[peek](#), 21  
[prop\\_max](#), 21  
[prop\\_min](#), 22

[remove\\_empty](#), 22  
[restore\\_changes](#), 23

[scale\\_01](#), 23  
[search\\_scripts](#), 24  
[skewness](#), 24

[text\\_date](#), 25  
[track\\_changes](#), 25  
[ttrue](#), 26

[unforge](#), 26

[workspace](#), 27  
[workspace\\_sapply](#), 28