Package: DTwrappers2 (via r-universe)

September 15, 2024

Title Extensions of 'DTwrappers'

Version 0.0.3

Depends R (>= 3.1.0)

Description Offers functionality which provides methods for data analyses and cleaning that can be flexibly applied across multiple variables and in groups. These include cleaning accidental text, contingent calculations, counting missing data, and building summarizations of the data.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.1

Suggests knitr, rmarkdown, covr, devtools, markdown, testthat (>= 2.1.0)

VignetteBuilder knitr

Imports data.table, DTwrappers

NeedsCompilation no

Author David Shilane [aut], Mayur Bansal [ctb], Srivastav Budugutta [ctb, cre]

Maintainer Srivastav Budugutta <sb4788@columbia.edu>

Repository https://srivastavbudugutta.r-universe.dev

RemoteUrl https://github.com/srivastavbudugutta/dtwrappers2

RemoteRef HEAD

RemoteSha 4f73a5e34253367b94fc9c9c766aea3e41d2d58c

Contents

character.coercion.culprits 2
dt.character.coercion.culprits 3
dt.format.numerics 4
dt.max.numerics 6

dt.mean.measured . . . 7
dt.mean.missing . . . 9
dt.mean.numerics . . . 10
dt.median.numerics . . . 12
dt.min.numerics . . . 13
dt.quantile.numerics . . . 15
dt.remove.erroneous.characters . . . 16
dt.round.exactly . . . 17
dt.round.numerics . . . 19
dt.sd.numerics . . . 20
dt.summarize . . . 22
dt.total.measured . . . 23
dt.total.missing . . . 24
dt.trimws.character . . . 26
dt.var.numerics . . . 27
format.numerics . . . 29
lower.quartile . . . 30
max.numerics . . . 31
mean.measured . . . 31
mean.missing . . . 32
mean.numerics . . . 32
median.numerics . . . 33
min.numerics . . . 33
remove.erroneous.characters . . . 34
round.numerics . . . 35
round_exactly . . . 35
round_exactly_one_value . . . 36
sd.numerics . . . 37
total.measured . . . 37
total.missing . . . 38
var.numerics . . . 38

Index **39**

`character.coercion.culprits`

character.coercion.culprits This function identifies which character values in a vector are preventing it from being converted to a numeric vector.

Description

character.coercion.culprits This function identifies which character values in a vector are preventing it from being converted to a numeric vector.

Usage

```
character.coercion.culprits(x, threshold.for.numeric = 0.5, ...)
```

Arguments

x	A character vector of values that should be a numeric vector but was coerced to a character due to a small number of non-numeric entries.
threshold.for.numeric	A value between 0 and 1 specifying the maximum proportion of x that does not "look" numeric. If threshold.for.numeric = 0.1, then no more than 10 percentage of the values in x can be values that do not "look" numeric.
...	Additional arguments.

Value

A character vector of values that are preventing x from being converted to numeric. Returns NA if the proportion of non-numeric values exceeds the threshold.

---

dt.character.coercion.culprits

dt.character.coercion.culprits

Description

a wrapper function to determine if a character variable that might reasonably be reformatted as numeric

Usage

```
dt.character.coercion.culprits(
  dt.name,
  threshold.for.numeric = 0.5,
  the.variables = ".",
  the.filter = NULL,
  grouping.variables = NULL,
  grouping.type = "keyby",
  add.function.name = FALSE,
  ...
)
```

Arguments

dt.name	a character value specifying the name of a data.frame or data.table object.
threshold.for.numeric	a value between 0 and 1 specifying the maximum proportion of x that does not "look" numeric, e.g. "2.154" is a character value that can be converted to a numeric value.. If threshold.for.numeric = 0.1, then no more than 10 percent of the values in x can be values that do not "look" numeric.

`the.variables` a character vector specifying the variables that we want to apply a function to. Only values that exist in names(dat) will be used; other values in the.variables will be excluded from the calculation. When the.variables includes ".", then all values in names(dat) will be selected. Values of the.variables that also exist in grouping.variables will be excluded from the.variables (but grouped by these values).

`the.filter` a character value, logical value, or expression stating the logical operations to be performed in filtering the data prior to calculating the.function.

`grouping.variables` a character vector specifying variables to group by in performing the computation. Only values that exist in names(dat) will be used.

`grouping.type` a character value specifying whether the grouping should be sorted (keyby) or as is (by). Defaults to keyby unless "by" is specified.

`add.function.name` a logical value specifying whether the name of the function applied should be appended to the column names in the resulting table.

`...` additional arguments to be passed

Value

Returns a data table object resulting from the application of the 'character coercion culprits' analysis on the specified data frame or data table (dt). This function identifies character variables within the specified columns (the.variables) of the data table 'dt.name' that could potentially be converted to numeric based on the specified 'threshold.for.numeric'. It applies the given logical filter (if any) before the analysis and groups the results based on 'grouping.variables' and 'grouping.type' parameters. If 'add.function.name' is TRUE, the name of the function is appended to the column names in the resultant table. The output will contain columns corresponding to the analyzed variables, indicating the proportion of values in each that can potentially be converted to numeric, respecting the specified threshold.

`dt.format.numerics` *dt.format.numerics*

Description

wrapper of the format function that is only applied to numeric inputs

Usage

```
dt.format.numerics(
  dt.name,
  digits,
  the.variables = ".",
  the.filter = NULL,
  grouping.variables = NULL,
  add.function.name = FALSE,
```

```
  return.as = "result",
  big.mark = "",
  big.interval = 3L,
  small.mark = "",
  small.interval = 5L,
  decimal.mark = getOption("OutDec"),
  input.d.mark = decimal.mark,
  preserve.width = c("common", "individual", "none"),
  envir = parent.frame(),
  ...
)
```

Arguments

dt.name	a character value specifying the name of a data.frame or data.table object.
digits	the number of digits to round to. This number will be exact, in that there will be exactly k decimal places listed even if this includes lagging zeros. For instance, setting k = 5 for x = 2.54 would result in 2.54000
the.variables	a character vector specifying the variables that we want to apply a function to. Only values that exist in names(dat) will be used; other values in the.variables will be excluded from the calculation. When the.variables includes ".", then all values in names(dat) will be selected. Values of the.variables that also exist in grouping.variables will be excluded from the.variables (but grouped by these values).
the.filter	a character value, logical value, or expression stating the logical operations to be performed in filtering the data prior to calculating the.function.
grouping.variables	a character vector specifying variables to group by in performing the computation. Only values that exist in names(dat) will be used.
add.function.name	a logical value specifying whether the name of the function applied should be appended to the column names in the resulting table.
return.as	describes whether return should be result, code or mixture of both
big.mark	big mark
big.interval	big.interval
small.mark	small mark
small.interval	small interval
decimal.mark	decimal mark
input.d.mark	input d mark
preserve.width	preserve width
envir	the environment in which the code would be evaluated; parent.frame() by default.
...	additional arguments to be passed

Value

Depending on the value of 'return.as', this function returns different outputs: - If 'return.as' is "result", it returns a data frame or data table with the specified numeric columns formatted according to the provided parameters. This includes adjustments to decimal places, digit grouping, and the inclusion of specified marks for readability. - If 'return.as' is "code", it might return the R code or expressions that would result in the formatted data, allowing users to review or execute the formatting commands separately. - If 'return.as' is a mixture of both or another specified return type, the output may combine both the formatted data and the corresponding R code or expressions.

The function is designed to apply numeric formatting like rounding to a specified number of digits, adding thousand separators, and adjusting decimal marks, according to the parameters provided by the user. The exact nature of the returned object is determined by the function's settings and the input data.

`dt.max.numerics` *dt.max.numerics*

Description

wrapper function that computes the maximal value for each selected quantitative variable in each group after applying a filter

Usage

```
dt.max.numerics(
  dt.name,
  the.variables = ".",
  the.filter = NULL,
  grouping.variables = NULL,
  sortby.group = TRUE,
  table.format = "wide",
  add.function.name = FALSE,
  return.as = "result",
  envir = parent.frame(),
  na.rm = TRUE,
  non.numeric.value = "missing",
  ...
)
```

Arguments

`dt.name`	a character value specifying the name of a data.frame or data.table object.
`the.variables`	a character vector specifying the variables that we want to apply a function to. Only values that exist in names(dat) will be used; other values in the.variables will be excluded from the calculation. When the.variables includes ".", then all values in names(dat) will be selected. Values of the.variables that also exist in grouping.variables will be excluded from the.variables (but grouped by these values).

`the.filter`	a character value, logical value, or expression stating the logical operations to be performed in filtering the data prior to calculating the.function.
`grouping.variables`	a character vector specifying variables to group by in performing the computation. Only values that exist in names(dat) will be used.
`sortby.group`	a logical value to specify if the sorting functionality needs to be applied or not
`table.format`	a character vector soecifying if table should be in a wide format or a tall format
`add.function.name`	a logical value specifying whether the name of the function applied should be appended to the column names in the resulting table.
`return.as`	describes whether return should be result, code or mixture of both
`envir`	the environment in which the code would be evaluated; parent.frame() by default.
`na.rm`	a logical value specifying whether missing values should be removed from the calculations specified by the.functions.
`non.numeric.value`	if "missing", returns NA for variables that are not numeric, integer, logical, or complex. Otherwise returns first entry of the vector.
`...`	additional arguments to be passed

Value

The function returns a data frame or data table, depending on the 'return.as' parameter: - If 'return.as' is "result", it returns a modified version of the input data frame or data table with the maximum values computed for the specified numeric variables, after applying any filters and grouping as specified. The function respects the 'na.rm' parameter to handle missing values and the 'non.numeric.value' setting for non-numeric columns. - If 'return.as' is "code", it provides the R code or expressions intended to generate the result, allowing the user to evaluate or inspect the logic separately. - If 'return.as' specifies a mixture or an alternative option, the output may include both the calculated maximum values and the corresponding R code or expressions.

The function adapts to the 'table.format' parameter, offering the results in either a wide or tall format, and can also sort the results by group if 'sortby.group' is set to TRUE. The inclusion of the function name in the output column names is controlled by 'add.function.name'.

`dt.mean.measured` *dt.mean.measured*

Description

Calculates the proportion of measured values for each specified variable in each group after applying a filter.

Usage

```
dt.mean.measured(
  dt.name,
  the.variables = ".",
  the.filter = NULL,
  grouping.variables = NULL,
  sortby.group = TRUE,
  table.format = "wide",
  add.function.name = FALSE,
  return.as = "result",
  envir = parent.frame(),
  ...
)
```

Arguments

dt.name	a character value specifying the name of a data.frame or data.table object.
the.variables	a character vector specifying the variables that we want to apply a function to. Only values that exist in names(dat) will be used; other values in the.variables will be excluded from the calculation. When the.variables includes ".", then all values in names(dat) will be selected. Values of the.variables that also exist in grouping.variables will be excluded from the.variables (but grouped by these values).
the.filter	a character value, logical value, or expression stating the logical operations to be performed in filtering the data prior to calculating the.function.
grouping.variables	a character vector specifying variables to group by in performing the computation. Only values that exist in names(dat) will be used.
sortby.group	a logical value to specify if the sorting functionality needs to be applied or not
table.format	a character vector specifying if table should be in a wide format or a tall format
add.function.name	a logical value specifying whether the name of the function applied should be appended to the column names in the resulting table.
return.as	describes whether return should be result, code or mixture of both
envir	the environment in which the code would be evaluated; parent.frame() by default.
...	additional arguments to be passed

Value

The function returns an object based on the 'return.as' parameter: - If 'return.as' is "result", it outputs a modified version of the input data frame or data table, showing the proportion of measured (non-missing) values for each specified variable, potentially grouped and sorted as defined by the user. The results are presented in the format (wide or tall) specified by the 'table.format' parameter. - If 'return.as' is "code", the function provides the R code or expressions that would generate the aforementioned results, allowing users to evaluate or review the logic independently. - For other

values of 'return.as', the output may include both the computed proportions and the corresponding R code, depending on the function's implementation.

The function is designed to facilitate an understanding of data completeness, providing insights into the proportion of actual measurements available within the dataset, after any specified subgrouping and filtering.

`dt.mean.missing` *dt.mean.missing*

Description

Calculates the proportion of measured values for each specified variable in each group after applying a filter.

Usage

```
dt.mean.missing(
  dt.name,
  the.variables = ".",
  the.filter = NULL,
  grouping.variables = NULL,
  sortby.group = TRUE,
  table.format = "wide",
  add.function.name = FALSE,
  return.as = "result",
  envir = parent.frame(),
  ...
)
```

Arguments

`dt.name`	a character value specifying the name of a data.frame or data.table object.
`the.variables`	a character vector specifying the variables that we want to apply a function to. Only values that exist in names(dat) will be used; other values in the.variables will be excluded from the calculation. When the.variables includes ".", then all values in names(dat) will be selected. Values of the.variables that also exist in grouping.variables will be excluded from the.variables (but grouped by these values).
`the.filter`	a character value, logical value, or expression stating the logical operations to be performed in filtering the data prior to calculating the.function.
`grouping.variables`	a character vector specifying variables to group by in performing the computation. Only values that exist in names(dat) will be used.
`sortby.group`	a logical value to specify if the sorting functionality needs to be applied or not
`table.format`	a character vector specifying if table should be in a wide format or a tall format

`add.function.name`
: a logical value specifying whether the name of the function applied should be appended to the column names in the resulting table.

`return.as`
: describes whether return should be result, code or mixture of both

`envir`
: the environment in which the code would be evaluated; parent.frame() by default.

`...`
: additional arguments to be passed

Value

The function outputs an object based on the specified 'return.as' parameter: - If 'return.as' is "result", it returns a data frame or data table modified to include the proportion of missing values for each specified variable, after the data has been filtered and potentially grouped according to the parameters provided. The results are formatted according to the 'table.format' parameter, in either wide or tall form, and can be sorted by groups if 'sortby.group' is enabled. - If 'return.as' is "code", the function will return the R code or expressions designed to calculate these proportions, allowing the user to review or execute the code independently. - If 'return.as' encompasses other values, the output may combine the computed results and the R code, varying with the function's implementation and user specifications.

This function is tailored for analyzing data completeness, specifically by quantifying the missingness in the dataset, facilitating detailed examination and understanding of the data's integrity, especially after applying any specified filters and groupings.

`dt.mean.numerics` *dt.mean.numerics*

Description

wrapper function that computes the mean value for each selected quantitative variable in each group after applying a filter.

Usage

```
dt.mean.numerics(
  dt.name,
  the.variables = ".",
  the.filter = NULL,
  grouping.variables = NULL,
  sortby.group = TRUE,
  table.format = "wide",
  add.function.name = FALSE,
  return.as = "result",
  envir = parent.frame(),
  na.rm = TRUE,
  non.numeric.value = "missing",
  ...
)
```

Arguments

dt.name	a character value specifying the name of a data.frame or data.table object.
the.variables	a character vector specifying the variables that we want to apply a function to. Only values that exist in names(dat) will be used; other values in the.variables will be excluded from the calculation. When the.variables includes ".", then all values in names(dat) will be selected. Values of the.variables that also exist in grouping.variables will be excluded from the.variables (but grouped by these values).
the.filter	a character value, logical value, or expression stating the logical operations to be performed in filtering the data prior to calculating the.function.
grouping.variables	a character vector specifying variables to group by in performing the computation. Only values that exist in names(dat) will be used.
sortby.group	a logical value to specify if the sorting functionality needs to be applied or not
table.format	a character vector specifying if table should be in a wide format or a tall format
add.function.name	a logical value specifying whether the name of the function applied should be appended to the column names in the resulting table.
return.as	describes whether return should be result, code or mixture of both
envir	the environment in which the code would be evaluated; parent.frame() by default.
na.rm	a logical value specifying whether missing values should be removed from the calculations specified by the.functions.
non.numeric.value	if "missing", returns NA for variables that are not numeric, integer, logical, or complex. Otherwise returns first entry of the vector.
...	additional arguments to be passed

Value

The function returns an object determined by the 'return.as' parameter: - If 'return.as' is "result", it outputs a modified version of the input data frame or data table, presenting the mean values for each selected numeric variable, adjusted for any applied filters and groupings. The results are formatted according to the user's preference (wide or tall format) and can incorporate sorting by groups if specified. - If 'return.as' is "code", it provides the R code or expressions that would result in the calculation of these means, which allows the user to review or manually execute the code. - If 'return.as' includes other values, the output might combine both the calculated means and the R code, depending on the function's implementation.

The function efficiently aggregates the mean values, considering the handling of missing values as specified by 'na.rm' and adjusting for non-numeric values based on 'non.numeric.value'. This enables a detailed analysis of the dataset's quantitative aspects, especially after subgrouping and applying specific filters.

dt.median.numerics *dt.median.numerics*

Description

wrapper function that computes the median value for each selected quantitative variable in each group after applying a filter.

Usage

```
dt.median.numerics(
  dt.name,
  the.variables = ".",
  the.filter = NULL,
  grouping.variables = NULL,
  sortby.group = TRUE,
  table.format = "wide",
  add.function.name = FALSE,
  return.as = "result",
  envir = parent.frame(),
  na.rm = TRUE,
  non.numeric.value = "missing",
  ...
)
```

Arguments

dt.name	a character value specifying the name of a data.frame or data.table object.
the.variables	a character vector specifying the variables that we want to apply a function to. Only values that exist in names(dat) will be used; other values in the.variables will be excluded from the calculation. When the.variables includes ".", then all values in names(dat) will be selected. Values of the.variables that also exist in grouping.variables will be excluded from the.variables (but grouped by these values).
the.filter	a character value, logical value, or expression stating the logical operations to be performed in filtering the data prior to calculating the.function.
grouping.variables	a character vector specifying variables to group by in performing the computation. Only values that exist in names(dat) will be used.
sortby.group	a logical value to specify if the sorting functionality needs to be applied or not
table.format	a character vector specifying if table should be in a wide format or a tall format
add.function.name	a logical value specifying whether the name of the function applied should be appended to the column names in the resulting table.
return.as	describes whether return should be result, code or mixture of both

envir	the environment in which the code would be evaluated; parent.frame() by default.
na.rm	a logical value specifying whether missing values should be removed from the calculations specified by the.functions.
non.numeric.value	if "missing", returns NA for variables that are not numeric, integer, logical, or complex. Otherwise returns first entry of the vector.
...	additional arguments to be passed

Value

The function returns an output based on the 'return.as' parameter: - If 'return.as' is "result", it provides a modified version of the input data frame or data table, showing the median values for the specified numeric variables, after applying the set filters and groupings. The data is presented in the format specified by 'table.format', either wide or tall, and it reflects any sorting by group as dictated by 'sortby.group'. - If 'return.as' is "code", the function will return the R code or expressions that would generate the calculated medians, offering users the opportunity to inspect or execute the code separately. - If 'return.as' incorporates other options, the output may consist of both the calculated medians and the R code, varying according to the function's implementation.

The function aims to aggregate median values, taking into account the handling of missing values as specified by 'na.rm' and adapting for non-numeric values as determined by 'non.numeric.value'. This functionality allows for a nuanced analysis of the central tendency within the dataset, particularly after subgrouping and applying specified filters.

`dt.min.numerics` *dt.min.numerics*

Description

wrapper function that computes the minimal value for each selected quantitative variable in each group after applying a filter.

Usage

```
dt.min.numerics(
  dt.name,
  the.variables = ".",
  the.filter = NULL,
  grouping.variables = NULL,
  sortby.group = TRUE,
  table.format = "wide",
  add.function.name = FALSE,
  return.as = "result",
  envir = parent.frame(),
  na.rm = TRUE,
  non.numeric.value = "missing",
  ...
)
```

Arguments

dt.name	a character value specifying the name of a data.frame or data.table object.
the.variables	a character vector specifying the variables that we want to apply a function to. Only values that exist in names(dat) will be used; other values in the.variables will be excluded from the calculation. When the.variables includes ".", then all values in names(dat) will be selected. Values of the.variables that also exist in grouping.variables will be excluded from the.variables (but grouped by these values).
the.filter	a character value, logical value, or expression stating the logical operations to be performed in filtering the data prior to calculating the.function.
grouping.variables	a character vector specifying variables to group by in performing the computation. Only values that exist in names(dat) will be used.
sortby.group	a logical value to specify if the sorting functionality needs to be applied or not
table.format	a character vector specifying if table should be in a wide format or a tall format
add.function.name	a logical value specifying whether the name of the function applied should be appended to the column names in the resulting table.
return.as	describes whether return should be result, code or mixture of both
envir	the environment in which the code would be evaluated; parent.frame() by default.
na.rm	a logical value specifying whether missing values should be removed from the calculations specified by the.functions.
non.numeric.value	if "missing", returns NA for variables that are not numeric, integer, logical, or complex. Otherwise returns first entry of the vector.
...	additional arguments to be passed

Value

The function's output depends on the 'return.as' parameter: - If 'return.as' is "result", it generates a modified version of the input data frame or data table, detailing the minimum values for each specified numeric variable, adjusted for any applied filters and groupings. The data is structured according to the specified 'table.format' (wide or tall) and reflects sorting by group if 'sortby.group' is enabled. - If 'return.as' is "code", the function returns the R code or expressions that would compute these minimum values, giving the user the ability to inspect or execute the code separately. - For other values of 'return.as', the function may produce both the calculated minimum values and the R code, depending on the function's specifics.

This function is designed to compute and aggregate minimum values, accounting for the handling of missing data as dictated by 'na.rm' and addressing non-numeric values as per 'non.numeric.value'. It provides insightful analysis into the lower bounds of the dataset's quantitative variables, particularly after implementing specified subgroupings and filters.

dt.quantile.numerics *dt.quantile.numerics*

Description

wrapper function that computes the quantiles for each selected quantitative variable in each group after applying a filter.

Usage

```
dt.quantile.numerics(
  dt.name,
  the.variables = ".",
  probs = c(0.25, 0.75),
  the.filter = NULL,
  grouping.variables = NULL,
  sortby.group = TRUE,
  table.format = "long",
  return.as = "result",
  envir = parent.frame(),
  ...
)
```

Arguments

dt.name	a character value specifying the name of a data.frame or data.table object.
the.variables	a character vector specifying the variables that we want to apply a function to. Only values that exist in names(dat) will be used; other values in the.variables will be excluded from the calculation. When the.variables includes ".", then all values in names(dat) will be selected. Values of the.variables that also exist in grouping.variables will be excluded from the.variables (but grouped by these values).
probs	the range specifying the upper and lower quartiles
the.filter	a character value, logical value, or expression stating the logical operations to be performed in filtering the data prior to calculating the.function.
grouping.variables	a character vector specifying variables to group by in performing the computation. Only values that exist in names(dat) will be used.
sortby.group	a logical value to specify if the sorting functionality needs to be applied or not
table.format	a character vector specifying if table should be in a wide format or a tall format
return.as	describes whether return should be result, code or mixture of both
envir	the environment in which the code would be evaluated; parent.frame() by default.
...	additional arguments to be passed

Value

The function outputs depend on the specified 'return.as' parameter: - If 'return.as' is "result", it returns a modified version of the input data frame or data table, including the calculated quantiles for each specified numeric variable. These quantiles are computed based on the provided 'probs' array, after applying any specified filters and subgroupings. The results are structured according to the 'table.format', which can be wide or long, reflecting any group-based sorting if 'sortby.group' is enabled. - If 'return.as' is "code", the function will return the R code or expressions capable of computing the quantiles, allowing users to review or execute the calculations separately. - For other specified values of 'return.as', the function may provide both the quantile calculations and the corresponding R code, depending on how the function is implemented.

This function facilitates a comprehensive analysis of the data's distribution by calculating specific quantiles, aiding in the statistical examination of the dataset's quantitative attributes, particularly after implementing specific filters and groupings.

`dt.remove.erroneous.characters`

dt.remove.erroneous.characters

Description

removes erroneous characters

Usage

```
dt.remove.erroneous.characters(
  dt.name,
  threshold.for.numeric = 0.8,
  the.variables = ".",
  the.filter = NULL,
  variable.should.be = "numeric",
  value.for.missing = NULL,
  return.as = "result",
  envir = parent.frame(),
  ...
)
```

Arguments

`dt.name`	a character value specifying the name of a data.frame or data.table object.
`threshold.for.numeric`	a value between 0 and 1 specifying the maximum proportion of x that does not "look" numeric, e.g. "2.154" is a character value that can be converted to a numeric value.. If threshold.for.numeric = 0.1, then no more than 10 percent of the values in x can be values that do not "look" numeric.

`the.variables` a character vector specifying the variables that we want to apply a function to. Only values that exist in names(dat) will be used; other values in the.variables will be excluded from the calculation. When the.variables includes ".", then all values in names(dat) will be selected. Values of the.variables that also exist in grouping.variables will be excluded from the.variables (but grouped by these values).

`the.filter` a character value, logical value, or expression stating the logical operations to be performed in filtering the data prior to calculating the.function.

`variable.should.be` a character vector specifying whether variable should be numeric or text or something else

`value.for.missing` a character value, logical value, or expression stating the logical operations to be performed in stating the missing value

`return.as` describes whether return should be result, code or mixture of both

`envir` the environment in which the code would be evaluated; parent.frame() by default.

`...` additional arguments to be passed

Value

The output of the function is contingent upon the 'return.as' parameter: - If 'return.as' is "result", it returns a modified version of the input data frame or data table where erroneous characters in the specified variables are removed based on the given threshold. The function ensures that only values fitting the specified 'variable.should.be' criteria are retained, converting or imputing others as necessary, guided by the 'threshold.for.numeric' and 'value.for.missing' parameters. - If 'return.as' is "code", it provides the R code or expressions that would perform this character removal and data cleaning, allowing users to review or manually execute the modifications. - If another option is specified for 'return.as', the output may include both the cleaned data and the corresponding R code, depending on the specifics of the function's implementation.

The function aims to sanitize data by eliminating or correcting values in specified variables that do not conform to the expected numeric or text formats, thereby enhancing data quality and consistency.

`dt.round.exactly` *dt.round.exactly*

Description

rounds the number to exactly desired decimals

Usage

```
dt.round.exactly(
  dt.name,
  the.variables = ".",
  digits = 0,
  the.filter = NULL,
  decimal = ".",
  return.as = "result",
  envir = parent.frame(),
  ...
)
```

Arguments

dt.name	a character value specifying the name of a data.frame or data.table object.
the.variables	a character vector specifying the variables that we want to apply a function to. Only values that exist in names(dat) will be used; other values in the.variables will be excluded from the calculation. When the.variables includes ".", then all values in names(dat) will be selected. Values of the.variables that also exist in grouping.variables will be excluded from the.variables (but grouped by these values).
digits	the number of digits to round to. This number will be exact, in that there will be exactly k decimal places listed even if this includes lagging zeros. For instance, setting k = 5 for x = 2.54 would result in 2.54000
the.filter	a character value, logical value, or expression stating the logical operations to be performed in filtering the data prior to calculating the.function.
decimal	The character specifying the decimal, which splits between whole numbers (greater than 1) and the fractional component (less than 1).
return.as	describes whether return should be result, code or mixture of both
envir	the environment in which the code would be evaluated; parent.frame() by default.
...	additional arguments to be passed

Value

The function's output depends on the 'return.as' parameter: - If 'return.as' is "result", it returns a modified version of the input data frame or data table with the specified variables rounded to the exact number of decimal places as defined by 'digits'. The rounding is performed even if this results in trailing zeros, ensuring that each value has precisely the specified number of decimal places. - If 'return.as' is "code", the function provides the R code or expressions that would perform this precise rounding, allowing users to review or manually execute the code as needed. - If 'return.as' specifies another option, the output might include both the rounded data and the corresponding R code, depending on the function's implementation.

The function facilitates precise numerical formatting, enhancing data readability and consistency, especially useful in scenarios where a specific decimal precision is required for subsequent analyses or reporting.

dt.round.numerics *dt.round.numerics*

Description

rounds the number to desired decimal digits

Usage

```
dt.round.numerics(
  dt.name,
  digits,
  the.variables = ".",
  the.filter = NULL,
  grouping.variables = NULL,
  sortby.group = TRUE,
  add.function.name = FALSE,
  return.as = "result",
  envir = parent.frame(),
  ...
)
```

Arguments

dt.name	a character value specifying the name of a data.frame or data.table object.
digits	the number of digits to round to. This number will be exact, in that there will be exactly k decimal places listed even if this includes lagging zeros. For instance, setting k = 5 for x = 2.54 would result in 2.54000
the.variables	a character vector specifying the variables that we want to apply a function to. Only values that exist in names(dat) will be used; other values in the.variables will be excluded from the calculation. When the.variables includes ".", then all values in names(dat) will be selected. Values of the.variables that also exist in grouping.variables will be excluded from the.variables (but grouped by these values).
the.filter	a character value, logical value, or expression stating the logical operations to be performed in filtering the data prior to calculating the.function.
grouping.variables	a character vector specifying variables to group by in performing the computation. Only values that exist in names(dat) will be used.
sortby.group	a logical value to specify if the sorting functionality needs to be applied or not
add.function.name	a logical value specifying whether the name of the function applied should be appended to the column names in the resulting table.
return.as	describes whether return should be result, code or mixture of both

envir	the environment in which the code would be evaluated; parent.frame() by default.
...	additional arguments to be passed

Value

The function's output varies based on the 'return.as' parameter: - If 'return.as' is "result", it returns the input data frame or data table with specified numeric variables rounded to the defined number of decimal places. The rounding is applied even if it results in trailing zeros, ensuring a consistent number of decimal places across the data. - If 'return.as' is "code", the function provides the R code or expressions designed to perform this rounding, allowing users to examine or execute the rounding logic independently. - If another option is specified for 'return.as', the output may include both the rounded data and the corresponding R code, varying with the function's setup.

The function is crafted to modify numerical data by rounding it to a specified number of decimal places, aiding in data standardization and precision control, especially beneficial for detailed numerical analysis or reporting.

`dt.sd.numerics` *dt.sd.numerics*

Description

wrapper function that computes the standard deviation for each selected quantitative variable in each group after applying a filter.

Usage

```
dt.sd.numerics(
  dt.name,
  the.variables = ".",
  the.filter = NULL,
  grouping.variables = NULL,
  sortby.group = TRUE,
  table.format = "wide",
  add.function.name = FALSE,
  return.as = "result",
  envir = parent.frame(),
  na.rm = TRUE,
  non.numeric.value = "missing",
  ...
)
```

Arguments

dt.name	a character value specifying the name of a data.frame or data.table object.

`the.variables` a character vector specifying the variables that we want to apply a function to. Only values that exist in names(dat) will be used; other values in the.variables will be excluded from the calculation. When the.variables includes ".", then all values in names(dat) will be selected. Values of the.variables that also exist in grouping.variables will be excluded from the.variables (but grouped by these values).

`the.filter` a character value, logical value, or expression stating the logical operations to be performed in filtering the data prior to calculating the.function.

`grouping.variables` a character vector specifying variables to group by in performing the computation. Only values that exist in names(dat) will be used.

`sortby.group` a logical value to specify if the sorting functionality needs to be applied or not

`table.format` a character vector specifying if table should be in a wide format or a tall format

`add.function.name` a logical value specifying whether the name of the function applied should be appended to the column names in the resulting table.

`return.as` describes whether return should be result, code or mixture of both

`envir` the environment in which the code would be evaluated; parent.frame() by default.

`na.rm` a logical value specifying whether missing values should be removed from the calculations specified by the.functions.

`non.numeric.value` if "missing", returns NA for variables that are not numeric, integer, logical, or complex. Otherwise returns first entry of the vector.

`...` additional arguments to be passed

Value

The output of the function varies based on the 'return.as' parameter: - If 'return.as' is "result", it provides a modified version of the input data frame or data table, incorporating the standard deviation values for the specified numeric variables, computed post any filtering and grouping operations. These standard deviations are calculated considering the 'na.rm' parameter, which dictates the handling of missing values. - If 'return.as' is "code", the function returns the R code or expressions that would compute the standard deviations, allowing the user to inspect or run the calculations separately. - If 'return.as' is set to another value, the output might include both the computed standard deviations and the R code, depending on the function's design.

This function is designed to provide a statistical summary, particularly the standard deviation, which is a measure of the amount of variation or dispersion of a set of values. It's particularly useful for data analysis and understanding the variability of the dataset.

dt.summarize *dt.summarize*

Description

summarizes the dataset

Usage

```
dt.summarize(
  dt.name,
  the.functions = c("min", "lower.quartile", "median", "mean", "upper.quartile", "max",
    "sd", "num.records", "total.missing"),
  the.variables = ".",
  the.filter = NULL,
  grouping.variables = NULL,
  sortby.group = TRUE,
  other.params = "",
  table.format = "long",
  add.function.name = TRUE,
  return.as = "result",
  envir = parent.frame(),
  ...
)
```

Arguments

dt.name	a character value specifying the name of a data.frame or data.table object.
the.functions	a character vector or list specifying the name of the function to apply to the variables. This may either be specified by the name of the function as a character (e.g. "mean") or by defining a function;
the.variables	a character vector specifying the variables that we want to apply a function to. Only values that exist in names(dat) will be used; other values in the.variables will be excluded from the calculation. When the.variables includes ".", then all values in names(dat) will be selected. Values of the.variables that also exist in grouping.variables will be excluded from the.variables (but grouped by these values).
the.filter	a character value, logical value, or expression stating the logical operations to be performed in filtering the data prior to calculating the.function.
grouping.variables	a character vector specifying variables to group by in performing the computation. Only values that exist in names(dat) will be used.
sortby.group	a logical value to specify if the sorting functionality needs to be applied or not
other.params	additional parameters to be passed
table.format	a character vector specifying if table should be in a wide format or a tall format

add.function.name
: a logical value specifying whether the name of the function applied should be appended to the column names in the resulting table.

return.as
: describes whether return should be result, code or mixture of both

envir
: the environment in which the code would be evaluated; parent.frame() by default.

...
: additional arguments to be passed

Value

The output of the function is determined by the 'return.as' parameter: - If 'return.as' is "result", it returns a data frame or data table that summarizes the specified variables using the functions listed in 'the.functions'. The summary might include statistics like minimum, maximum, mean, median, standard deviation, and other specified measures, applied after any set filtering and grouping. - If 'return.as' is "code", the function will return the R code or expressions that generate the summary, allowing users to inspect or execute the code independently. - If 'return.as' specifies a different option, the output may include both the summary statistics and the corresponding R code, varying with the function's implementation.

This function is intended to provide a comprehensive summary of the dataset, offering insights into each selected variable's distribution and central tendencies, facilitating a thorough understanding of the dataset's characteristics.

---

dt.total.measured *dt.total.measured*

Description

Calculates the proportion of measured values for each specified variable in each group after applying a filter.

Usage

```
dt.total.measured(
  dt.name,
  the.variables = ".",
  the.filter = NULL,
  grouping.variables = NULL,
  sortby.group = TRUE,
  table.format = "wide",
  add.function.name = FALSE,
  return.as = "result",
  envir = parent.frame(),
  ...
)
```

Arguments

dt.name	a character value specifying the name of a data.frame or data.table object.
the.variables	a character vector specifying the variables that we want to apply a function to. Only values that exist in names(dat) will be used; other values in the.variables will be excluded from the calculation. When the.variables includes ".", then all values in names(dat) will be selected. Values of the.variables that also exist in grouping.variables will be excluded from the.variables (but grouped by these values).
the.filter	a character value, logical value, or expression stating the logical operations to be performed in filtering the data prior to calculating the.function.
grouping.variables	a character vector specifying variables to group by in performing the computation. Only values that exist in names(dat) will be used.
sortby.group	a logical value to specify if the sorting functionality needs to be applied or not
table.format	a character vector soecifying if table should be in a wide format or a tall format
add.function.name	a logical value specifying whether the name of the function applied should be appended to the column names in the resulting table.
return.as	describes whether return should be result, code or mixture of both
envir	the environment in which the code would be evaluated; parent.frame() by default.
...	additional arguments to be passed

Value

The function's output varies based on the 'return.as' parameter: - If 'return.as' is "result", it returns a data frame or data table with the calculated proportions of measured (non-missing) values for each specified variable, adjusted for any applied filtering and subgrouping. The data is formatted as specified, in either a wide or tall format, depending on the 'table.format' setting. - If 'return.as' is "code", it provides the R code or expressions intended to generate these calculations, allowing the user to review or execute the code independently. - If another option is specified for 'return.as', the output may include both the calculated proportions and the corresponding R code, contingent on the function's design.

This function is particularly useful for assessing data completeness, providing insights into the proportion of actual measurements available within the dataset, especially after subgrouping and applying specific filters.

---

dt.total.missing *dt.total.missing*

Description

Calculates the proportion of missing values for each specified variable in each group after applying a filter.

Usage

```
dt.total.missing(
  dt.name,
  the.variables = ".",
  the.filter = NULL,
  grouping.variables = NULL,
  sortby.group = TRUE,
  table.format = "wide",
  add.function.name = FALSE,
  return.as = "result",
  envir = parent.frame(),
  ...
)
```

Arguments

dt.name	a character value specifying the name of a data.frame or data.table object.
the.variables	a character vector specifying the variables that we want to apply a function to. Only values that exist in names(dat) will be used; other values in the.variables will be excluded from the calculation. When the.variables includes ".", then all values in names(dat) will be selected. Values of the.variables that also exist in grouping.variables will be excluded from the.variables (but grouped by these values).
the.filter	a character value, logical value, or expression stating the logical operations to be performed in filtering the data prior to calculating the.function.
grouping.variables	a character vector specifying variables to group by in performing the computation. Only values that exist in names(dat) will be used.
sortby.group	a logical value to specify if the sorting functionality needs to be applied or not
table.format	a character vector soecifying if table should be in a wide format or a tall format
add.function.name	a logical value specifying whether the name of the function applied should be appended to the column names in the resulting table.
return.as	describes whether return should be result, code or mixture of both
envir	the environment in which the code would be evaluated; parent.frame() by default.
...	additional arguments to be passed

Value

The function's output is determined by the 'return.as' parameter: - If 'return.as' is "result", it provides a modified data frame or data table displaying the proportions of missing values for each specified variable. These calculations are performed after considering any filters applied and subgrouping defined, presented in the format specified by 'table.format'. - If 'return.as' is "code", the function outputs the R code or expressions designed to calculate these proportions, allowing users to inspect or execute the code independently. - If a different value is specified for 'return.as', the

output might include both the calculated proportions of missing values and the corresponding R code, depending on the specifics of the function's implementation.

This function is essential for data quality assessment, offering insights into the extent of missing data within each variable of the dataset, especially useful after applying subgrouping and specific filters.

`dt.trimws.character` *dt.trimws.character*

Description

wrapper function

Usage

```
dt.trimws.character(
  dt.name,
  the.variables = ".",
  the.filter = NULL,
  grouping.variables = NULL,
  sortby.group = TRUE,
  table.format = "wide",
  add.function.name = FALSE,
  return.as = "result",
  envir = parent.frame(),
  which = c("both", "left", "right"),
  whitespace = "[ \t\r\n]",
  convert.factor = FALSE,
  ...
)
```

Arguments

`dt.name`	a character value specifying the name of a data.frame or data.table object.
`the.variables`	a character vector specifying the variables that we want to apply a function to. Only values that exist in names(dat) will be used; other values in the.variables will be excluded from the calculation. When the.variables includes ".", then all values in names(dat) will be selected. Values of the.variables that also exist in grouping.variables will be excluded from the.variables (but grouped by these values).
`the.filter`	a character value, logical value, or expression stating the logical operations to be performed in filtering the data prior to calculating the.function.
`grouping.variables`	a character vector specifying variables to group by in performing the computation. Only values that exist in names(dat) will be used.
`sortby.group`	a logical value to specify if the sorting functionality needs to be applied or not

table.format	a character vector specifying if table should be in a wide format or a tall format
add.function.name	a logical value specifying whether the name of the function applied should be appended to the column names in the resulting table.
return.as	describes whether return should be result, code or mixture of both
envir	the environment in which the code would be evaluated; parent.frame() by default.
which	both, left or right
whitespace	encoded whitespace
convert.factor	logical value specifying if variable needs to be converted to a factor before calculations
...	additional arguments to be passed

Value

The function's output varies based on the 'return.as' parameter: - If 'return.as' is "result", it returns a modified version of the input data frame or data table with specified variables having their leading, trailing, or both types of whitespace trimmed according to the 'which' parameter. This is performed after any set filtering and subgrouping, with options for trimming specified by 'whitespace' and 'convert.factor'. - If 'return.as' is "code", it provides the R code or expressions designed to execute this whitespace trimming, allowing users to review or implement the modifications independently. - If another value is specified for 'return.as', the output might include both the adjusted data and the corresponding R code, depending on the function's configuration.

This function is crucial for cleaning character data, ensuring consistency, and preparing data for analysis by removing unwanted whitespace from the specified variables.

---

dt.var.numerics *dt.var.numerics*

Description

wrapper function that computes the variance for each selected quantitative variable in each group after applying a filter

Usage

```
dt.var.numerics(
  dt.name,
  the.variables = ".",
  the.filter = NULL,
  grouping.variables = NULL,
  sortby.group = TRUE,
  table.format = "wide",
  add.function.name = FALSE,
  return.as = "result",
```

```
  envir = parent.frame(),
  na.rm = TRUE,
  non.numeric.value = "missing",
  ...
)
```

Arguments

dt.name	a character value specifying the name of a data.frame or data.table object.
the.variables	a character vector specifying the variables that we want to apply a function to. Only values that exist in names(dat) will be used; other values in the.variables will be excluded from the calculation. When the.variables includes ".", then all values in names(dat) will be selected. Values of the.variables that also exist in grouping.variables will be excluded from the.variables (but grouped by these values).
the.filter	a character value, logical value, or expression stating the logical operations to be performed in filtering the data prior to calculating the.function.
grouping.variables	a character vector specifying variables to group by in performing the computation. Only values that exist in names(dat) will be used.
sortby.group	a logical value to specify if the sorting functionality needs to be applied or not
table.format	a character vector specifying if table should be in a wide format or a tall format
add.function.name	a logical value specifying whether the name of the function applied should be appended to the column names in the resulting table.
return.as	describes whether return should be result, code or mixture of both
envir	the environment in which the code would be evaluated; parent.frame() by default.
na.rm	a logical value specifying whether missing values should be removed from the calculations specified by the.functions.
non.numeric.value	if "missing", returns NA for variables that are not numeric, integer, logical, or complex. Otherwise returns first entry of the vector.
...	additional arguments to be passed

Value

The function's output varies based on the 'return.as' parameter: - If 'return.as' is "result", it returns a modified data frame or data table that includes the variance calculations for each specified quantitative variable, adjusted post any filtering and subgrouping. The variance is computed considering the 'na.rm' parameter to handle missing values and is presented in the specified 'table.format'. - If 'return.as' is "code", it provides the R code or expressions designed to perform these variance computations, allowing users to inspect or execute the code independently. - If 'return.as' specifies another option, the output may include both the calculated variance values and the corresponding R code, depending on the function's setup.

This function is particularly valuable for statistical analysis, providing insights into the variability of each variable within the dataset, which is crucial for understanding the data distribution and variability after subgrouping and applying specific filters.

`format.numerics` *This function formats numeric values with specified rounding and marking options.*

Description

This function formats numeric values with specified rounding and marking options.

Usage

```
## S3 method for class 'numerics'
format(
  x,
  digits = 0,
  big.mark = "",
  big.interval = 3L,
  small.mark = "",
  small.interval = 5L,
  decimal.mark = getOption("OutDec"),
  input.d.mark = decimal.mark,
  preserve.width = c("common", "individual", "none"),
  zero.print = NULL,
  replace.zero = FALSE,
  drop0trailing = FALSE,
  is.cmplx = NA,
  ...
)
```

Arguments

`x`	A numeric, integer, or logical vector to be formatted.
`digits`	The number of digits to round to. Defaults to 0.
`big.mark`	A character string used as a mark for thousands. Defaults to "".
`big.interval`	An integer specifying the interval for the big mark. Defaults to 3.
`small.mark`	A character string used as a mark for small intervals. Defaults to "".
`small.interval`	An integer specifying the interval for the small mark. Defaults to 5.
`decimal.mark`	A character string used as a decimal mark. Defaults to the value of getOption("OutDec").
`input.d.mark`	The decimal mark to be used for input. Defaults to the value of `decimal.mark`.
`preserve.width`	A character string specifying how to preserve the width of the output. Can be "common", "individual", or "none". Defaults to "common".

zero.print	A character string to be used for printing zero. Defaults to NULL.
replace.zero	A logical value indicating whether to replace zeros with the zero.print value. Defaults to FALSE.
drop0trailing	A logical value indicating whether to drop trailing zeros. Defaults to FALSE.
is.cmplx	A logical value indicating whether the input is complex. Defaults to NA.
...	Additional arguments passed to prettyNum.

Value

A character vector with formatted numeric values.

`lower.quartile` *lower.quartile*

Description

value that cuts off the first 25

Usage

```
lower.quartile(x, na.rm = TRUE, ...)
```

Arguments

x	a vector
na.rm	a logical value specifying whether missing values should be removed from the calculations specified by the.functions.
...	additional arguments to be passed

Value

Returns a numeric value representing the lower quartile (25th percentile) of the given vector 'x'. The lower quartile is the value below which 25 If 'na.rm' is TRUE, any missing values (NA) are removed before the calculation. The function utilizes the 'quantile' function internally to compute this statistic, and any additional arguments provided (...) are passed along to this underlying function.

max.numerics *max.numerics*

Description

This function computes the maximal value of a numeric, integer, logical, or complex vector. If the vector is not one of these types, the function returns either NA or the first entry of the vector.

Usage

```
## S3 method for class 'numerics'
max(x, na.rm = TRUE, non.numeric.value = "missing", ...)
```

Arguments

x	A vector.
na.rm	A logical value specifying whether missing values should be removed from the calculations specified by the function. Defaults to TRUE.
non.numeric.value	If "missing", returns NA for variables that are not numeric, integer, logical, or complex. Otherwise, returns the first entry of the vector. Defaults to "missing".
...	Additional arguments .

Value

If x is numeric, integer, logical, or complex, the maximal value will be computed. Otherwise, the first value of x will be returned untouched or NA based on non.numeric.value.

mean.measured *mean.measured*

Description

This function calculates the proportion of non-NA values in a vector.

Usage

```
## S3 method for class 'measured'
mean(x, ...)
```

Arguments

x	A vector.
...	Additional arguments.

Value

A numeric value representing the proportion of non-NA values in the vector.

`mean.missing` *Calculate the Proportion of NA Values*

Description

This function calculates the proportion of NA values in a vector.

Usage

```
## S3 method for class 'missing'
mean(x, ...)
```

Arguments

`x`	A vector.
`...`	Additional arguments.

Value

A numeric value representing the proportion of NA values in the vector x.

`mean.numerics` *mean.numerics*

Description

This function computes the mean value of a numeric, integer, logical, or complex vector. If the vector is not one of these types, the function returns either NA or the first entry of the vector.

Usage

```
## S3 method for class 'numerics'
mean(x, na.rm = TRUE, non.numeric.value = "missing", ...)
```

Arguments

`x`	A vector.
`na.rm`	A logical value specifying whether missing values should be removed from the calculations specified by the function. Defaults to TRUE.
`non.numeric.value`	If "missing", returns NA for variables that are not numeric, integer, logical, or complex. Otherwise, returns the first entry of the vector. Defaults to "missing".
`...`	Additional arguments.

Value

If x is numeric, integer, logical, or complex, the mean value will be computed. Otherwise, the first value of x will be returned untouched or NA based on non.numeric.value.

`median.numerics` *median.numerics*

Description

This function computes the median value of a numeric, integer, logical, or complex vector. If the vector is not one of these types, the function returns either NA or the first entry of the vector.

Usage

```
## S3 method for class 'numerics'
median(x, na.rm = TRUE, non.numeric.value = "missing", ...)
```

Arguments

`x`	A vector.
`na.rm`	A logical value specifying whether missing values should be removed from the calculations specified by the function. Defaults to TRUE.
`non.numeric.value`	If "missing", returns NA for variables that are not numeric, integer, logical, or complex. Otherwise, returns the first entry of the vector. Defaults to "missing".
`...`	Additional arguments .

Value

If x is numeric, integer, logical, or complex, the median value will be computed. Otherwise, the first value of x will be returned untouched or NA based on non.numeric.value.

`min.numerics` *min.numerics*

Description

This function computes the minimal value of a numeric, integer, logical, or complex vector. If the vector is not one of these types, the function returns either NA or the first entry of the vector.

Usage

```
## S3 method for class 'numerics'
min(x, na.rm = TRUE, non.numeric.value = "missing", ...)
```

Arguments

x	A vector.
na.rm	A logical value specifying whether missing values should be removed from the calculations specified by the function. Defaults to TRUE.
non.numeric.value	If "missing", returns NA for variables that are not numeric, integer, logical, or complex. Otherwise, returns the first entry of the vector. Defaults to "missing".
...	Additional arguments .

Value

If x is numeric, integer, logical, or complex, the minimal value will be computed. Otherwise, the first value of x will be returned untouched or NA based on non.numeric.value.

---

remove.erroneous.characters

remove.erroneous.characters

Description

This function attempts to convert a character vector to a numeric or complex vector, replacing erroneous values based on a specified threshold.

Usage

```
remove.erroneous.characters(
  x,
  threshold.for.numeric = 0.8,
  variable.should.be = "numeric",
  value.for.missing = NULL,
  ...
)
```

Arguments

x	A character vector of values that should be a numeric vector but was coerced to a character due to a small number of entries.
threshold.for.numeric	A value between 0 and 1 specifying the maximum proportion of x that does not "look" numeric. If threshold.for.numeric = 0.1, then no more than 10 percentage of the values in x can be values that do not "look" numeric.
variable.should.be	A character string specifying the target variable type ("numeric" or "complex"). Defaults to "numeric".
value.for.missing	The value to replace missing or erroneous entries with. Defaults to NA_real_ for numeric and NA_complex_ for complex.
...	Additional arguments .

Value

A numeric or complex vector with erroneous entries replaced, or the original character vector if the proportion of erroneous values exceeds the threshold.

`round.numerics` *round.numerics*

Description

This function rounds numeric or complex values to the specified number of digits. If the input is not numeric or complex, the values are returned untouched.

Usage

```
## S3 method for class 'numerics'
round(x, digits = 0, ...)
```

Arguments

`x`	A vector.
`digits`	The number of digits to round to. Defaults to 0.
`...`	Additional arguments.

Value

If x is numeric or complex, the values will be rounded to the specified number of digits. Otherwise, the values of x will be returned untouched.

`round_exactly` *round_exactly*

Description

This function rounds numeric values to a specified number of decimal places. The rounding is exact, meaning there will be exactly the specified number of decimal places even if this includes trailing zeros.

Usage

```
round_exactly(x, digits = 0, decimal = ".", ...)
```

Arguments

x	A numeric vector.
digits	The number of digits to round to. This number will be exact, meaning there will be exactly this number of decimal places listed even if this includes trailing zeros. For instance, setting digits = 5 for x = 2.54 would result in 2.54000.
decimal	The character specifying the decimal, which splits between whole numbers and the fractional component. Defaults to ".".
...	Additional arguments .

Value

A character vector of rounded numeric values with exactly the specified number of decimal places.

---

round_exactly_one_value

round_exactly_one_value This internal function rounds exactly one value to the specified number of digits.

Description

round_exactly_one_value This internal function rounds exactly one value to the specified number of digits.

Usage

```
round_exactly_one_value(x, digits)
```

Arguments

x	A character vector where the first element is the integer part and the second element is the decimal part.
digits	The number of digits to round to.

Value

A character string representing the rounded value.

sd.numerics *sd.numerics*

Description

This function computes the standard deviation of a numeric, integer, logical, or complex vector. If the vector is not one of these types, the function returns either NA or the first entry of the vector.

Usage

```
sd.numerics(x, na.rm = TRUE, non.numeric.value = "missing", ...)
```

Arguments

x	A vector.
na.rm	A logical value specifying whether missing values should be removed from the calculations specified by the function. Defaults to TRUE.
non.numeric.value	If "missing", returns NA for variables that are not numeric, integer, logical, or complex. Otherwise, returns the first entry of the vector. Defaults to "missing".
...	Additional arguments.

Value

If x is numeric, integer, logical, or complex, the standard deviation will be computed. Otherwise, the first value of x will be returned untouched or NA based on non.numeric.value.

total.measured *total.measured*

Description

This function calculates the total number of non-NA values in a vector.

Usage

```
total.measured(x, ...)
```

Arguments

x	A vector.
...	Additional arguments.

Value

An integer representing the total number of non-NA values in the vector x.

total.missing *total.missing*

Description

This function calculates the total number of NA values in a vector.

Usage

```
total.missing(x, ...)
```

Arguments

x	A vector.
...	Additional arguments.

Value

An integer representing the total number of NA values in the vector x.

var.numerics *var.numerics*

Description

This function computes the variance of a numeric, integer, logical, or complex vector. If the vector is not one of these types, the function returns either NA or the first entry of the vector.

Usage

```
var.numerics(x, na.rm = TRUE, non.numeric.value = "missing", ...)
```

Arguments

x	A vector.
na.rm	A logical value specifying whether missing values should be removed from the calculations specified by the function. Defaults to TRUE.
non.numeric.value	If "missing", returns NA for variables that are not numeric, integer, logical, or complex. Otherwise, returns the first entry of the vector. Defaults to "missing".
...	Additional arguments.

Value

If x is numeric, integer, logical, or complex, the variance will be computed. Otherwise, the first value of x will be returned untouched or NA based on non.numeric.value.

Index

character.coercion.culprits, 2

dt.character.coercion.culprits, 3
dt.format.numerics, 4
dt.max.numerics, 6
dt.mean.measured, 7
dt.mean.missing, 9
dt.mean.numerics, 10
dt.median.numerics, 12
dt.min.numerics, 13
dt.quantile.numerics, 15
dt.remove.erroneous.characters, 16
dt.round.exactly, 17
dt.round.numerics, 19
dt.sd.numerics, 20
dt.summarize, 22
dt.total.measured, 23
dt.total.missing, 24
dt.trimws.character, 26
dt.var.numerics, 27

format.numerics, 29

lower.quartile, 30

max.numerics, 31
mean.measured, 31
mean.missing, 32
mean.numerics, 32
median.numerics, 33
min.numerics, 33

remove.erroneous.characters, 34
round.numerics, 35
round_exactly, 35
round_exactly_one_value, 36

sd.numerics, 37

total.measured, 37
total.missing, 38

var.numerics, 38