Package 'doudpackage'

June 29, 2023

Title Create Elegant Table 1 in HTML for Bio-Statistics

Version 2.1.0

Description
Creates the ``table one'' of bio-medical papers. Fill it with your data and the name of the variable which you'll make the group(s) out of and it will make univariate, bivariate analysis and parse it into HTML.
It also allows you to visualize all your data with graphic representation.

License GPL (>= 3)

Encoding UTF-8

Language en-US

RoxygenNote 7.2.3

Imports dplyr, kableExtra, methods, parallel, purrr, stats, stringi, tibble, tidyr

NeedsCompilation no

Author Edouard Baudouin [aut, cre] (<https://orcid.org/0000-0002-8324-6908>)

Maintainer Edouard Baudouin <edouardpierre.baudouin@aphp.fr>

Repository CRAN

Date/Publication 2023-06-29 17:20:06 UTC

R topics documented:

`anaBiv` *anaBiv generic function*

Description

Generic function of anaBiv which gives bivariate analysis according to group

Usage

```
anaBiv(var, group, parallel, ...)
```

Arguments

`var`	listVar object or data.frame
`group`	Variable to make subgroups with
`parallel`	Logical. Make analysis using parallel from `parallel::mclapply()`.
`...`	digits.p can be specified as descTab

Value

A list of VarGroup object or data.frame

`anaBiv,data.frame,character-method`

anaBiv data.frame function

Description

Generic function of anaBiv which gives bivariate analysis according to group

Usage

```
## S4 method for signature 'data.frame,character'
anaBiv(var, group, parallel, ...)
```

Arguments

`var`	listVar object or data.frame
`group`	Variable to make subgroups with
`parallel`	Logical. Make analysis using parallel from `parallel::mclapply()`.
`...`	digits.p can be specified as descTab

Value

A list of VarGroup object or data.frame

`anaBiv,listVar,character-method`

anaBiv data.frame function

Description

Generic function of anaBiv which gives bivariate analysis according to group

Usage

```
## S4 method for signature 'listVar,character'
anaBiv(var, group, parallel, ...)
```

Arguments

`var`	listVar object or data.frame
`group`	Variable to make subgroups with
`parallel`	Logical. Make analysis using parallel from `parallel::mclapply()`.
`...`	digits.p can be specified as descTab

Value

A list of VarGroup object or data.frame

`descTab` *Generic function to create a table of descriptive analysis of a dataset*

Description

This function allows you to display all together all univariate analysis (median/mean; IQR/SD; proportions) and bivariates analysis (Wilcoxon, Chi² or Fisher). The univariate analysis can be sub-grouped by a variable of interest of n levels. Appropriate statistics test will be applied

Usage

```
descTab(
  data,
  group = NULL,
  quanti = TRUE,
  quali = TRUE,
  na.print = FALSE,
  pvalue = TRUE,
  digits.p = 3L,
  digits.qt = 1L,
  digits.ql = 1L,
  normality = "normal",
  parallel = FALSE,
  mc.cores = 0
)
```

Arguments

`data`	A datasaset. Needs to be a data.frame/tibble object
`group`	Optional. The name of the variable to make sub-groups comparisons.
`quanti, quali, na.print, pvalue`	Logical. If false, won't display quantitative/qualitative/Missing values/pvalues variable results
`digits.p`	Integer. Significant digits for p value
`digits.qt`	Integer. Significant digits for mean/median, SD/IQR
`digits.ql`	Integer. Significant digits for proportions
`normality`	One of "assess", "normal", "manual", "non normal". See details
`parallel`	Logical. Make analysis using parallel from `parallel::mclapply()`.
`mc.cores`	If parallel is TRUE, how many Cores to used.

Value

A S4 objects parseClass() containing the main table accessible by ["table"] subscript.

Examples

```
data(iris)
library(stringi)
iris$fact_1<-as.factor(as.character(sample(1:5, 150, replace = TRUE)))
n_na<-sample(1:150, 30)
iris[n_na, "fact_1"]<-NA
iris$fact_2<-as.factor(as.character(sample(1:2, 150, replace = TRUE)))
n_na<-sample(1:150, 10)
iris[n_na, "fact_2"]<-NA
iris$fact_3<-as.factor(as.character(stri_rand_strings(150, 1, '[A-B]')))
iris$num<-runif(150, min = 0, max = 100)
n_na<-sample(1:150, 5)
iris[n_na, "num"]<-NA
iris_test<-descTab(iris, group = "Species", na.print = TRUE)
```

---

ft_ana_biv *This function is depreciated, please use anaBiv().* anaBiv()

Description

This function is depreciated, please use anaBiv(). anaBiv()

Usage

```
ft_ana_biv(...)
```

Arguments

...	None

Value

No return value, depreciated

ft_desc_tab *This function is depreciated, please use anaBiv(). descTab()*

Description

This function is depreciated, please use anaBiv(). descTab()

Usage

```
ft_desc_tab(...)
```

Arguments

...	None

Value

No return value, depreciated

ft_parse *This function is depreciated, please use parseClassFun()*

Description

This function is depreciated, please use parseClassFun()

Usage

```
ft_parse(...)
```

Arguments

...	None

Value

No return value, depreciated

initialize,parseClass-method
S4 class initialization function

Description

Initialization function for parseClass object `initialize,parseClass-method()`

Usage

```
## S4 method for signature 'parseClass'
initialize(
  .Object,
  table,
  group,
  pvalue,
  na.print,
  quanti,
  quali,
  var_list,
  data,
  digits.qt,
  digits.ql
)
```

Arguments

`.Object`	The object to create
`table`	The result of descTab
`group`	The variable from which to make subgroups
`pvalue, na.print, quanti, quali`	Values from descTab `descTab()`
`var_list`	An object of listVar `listVar-class()`
`data`	The dataset provided in descTab
`digits.qt, digits.ql`	As provided in descTab

Value

parseClass object

`initialize,Var-method` *S4 class initialization function*

Description

Initialization function for Var `initialize,Var-method()`

Usage

```
## S4 method for signature 'Var'
initialize(.Object, name, type, normal)
```

Arguments

`.Object`	Object to be initialized
`name`	A character taking name of the variable
`type`	A character taking name of the variable type
`normal`	Logical, if variable, is numeric; is it normal

Value

Var Object

`initialize,VarGroup-method`
S4 class initialization function

Description

Initialization function for VarGroup `initialize,VarGroup-method()`

Usage

```
## S4 method for signature 'VarGroup'
initialize(
  .Object,
  x,
  group_var,
  pvalue,
  parsed_name,
  value,
  missing.value,
  missing.value.name
)
```

Arguments

.Object	Object to be initialized
x	A Var object
group_var	The subgroup for which proportions, mean/sd were calculated and missing values
pvalue	The calculated pvalue
parsed_name	The name of the variable parsed with the n (%), mean (SD)
value	The values calculated parsed
missing.value	Missing values numbers and proportions n (%)
missing.value.name	Missing values concatenate with the level of the variable if it factor

Value

VarGroup object

---

listVar-class *S4 class*

Description

A class of list of Var object

Slots

List a list of Var

---

parseClass *S4 class initialization function*

Description

Initialization function for parseClass object initialize,parseClass-method()

Usage

```
parseClass(
  table,
  group,
  pvalue,
  na.print,
  quanti,
  quali,
  var_list,
  data,
  digits.qt,
  digits.ql
)
```

Arguments

`table`	The result of descTab
`group`	The variable from which to make subgroups
`pvalue, na.print, quanti, quali`	Values from descTab `descTab()`
`var_list`	An object of listVar `listVar-class()`
`data`	The dataset provided in descTab
`digits.qt, digits.ql`	As provided in descTab

Value

parseClass object

`parseClass-class` *S4 class*

Description

A S4 class containing all the information needed for parsClassFun the missing values and the group for which it was calculated

Slots

`table` The result of descTab

`group` The variable from which to make subgroups

`pvalue,na.print,quanti,quali` Values from descTab `descTab()`

`var_list` An object of listVar `listVar-class()`

`data` The dataset provided in descTab

`digits.qt,digits.ql` As provided in descTab

parseClassFun *Make the LaTeX/HTML table. Generic function*

Description

Make the LaTeX/HTML table. Generic function

Usage

```
parseClassFun(
  table,
  col.order = NULL,
  levels_to_keep = NULL,
  group_rows_labels = NULL
)
```

Arguments

table	The output of descTab() or anaBiv(), an S4 object.
col.order	Optional. A vector containing the column order. If set, must contains at least all levels of group. Three columns created are "var", "Total", and "pvalue" which can be present in the vector
levels_to_keep	Optional, named list. If the variable is binary, which level to keep. Default is the last level of levels(variable). Must be as: list("variable name" = "level to keep").
group_rows_labels	Optional, named list. Create row labels in order to regroup them. Must be as list("label" = c("var1", "var2), "label2" = c("var3", "var4")).

Value

An HTML/LaTex file which can be used directly in Rmarkdown and copy paste

Examples

```
data(iris)
library(stringi)
iris$fact_1<-as.factor(as.character(sample(1:5, 150, replace = TRUE)))
n_na<-sample(1:150, 30)
iris[n_na, "fact_1"]<-NA
iris$fact_2<-as.factor(as.character(stri_rand_strings(150, 1, '[A-B]')))
iris$num<-runif(150, min = 0, max = 100)
n_na<-sample(1:150, 5)
iris[n_na, "num"]<-NA
iris_test<-descTab(iris, group = "Species", na.print = TRUE)
testParse<-parseClassFun(iris_test, levels_to_keep = list("fact_2" =  "A"),
group_rows_labels = list("Size" = c("Petal.Length", "Petal.Width"),
"My_f" = c("num", "fact_2")))
```

parseClassFun,parseClass-method
Make the LaTeX/HTML table

Description

This functions takes the S4 output of descTab to create an HTML parsed table

Usage

```
## S4 method for signature 'parseClass'
parseClassFun(
  table,
  col.order = NULL,
  levels_to_keep = NULL,
  group_rows_labels = NULL
)
```

Arguments

table	The output of descTab() or anaBiv(), an S4 object.
col.order	Optional. A vector containing the column order. If set, must contains at least all levels of group. Three columns created are "var", "Total", and "pvalue" which can be present in the vector
levels_to_keep	Optional, named list. If the variable is binary, which level to keep. Default is the last level of levels(variable). Must be as: list("variable name" = "level to keep").
group_rows_labels	Optional, named list. Create row labels in order to regroup them. Must be as list("label" = c("var1", "var2), "label2" = c("var3", "var4")).

Value

An HTML/LaTex file which can be used directly in Rmarkdown and copy paste

Examples

```
data(iris)
library(stringi)
iris$fact_1<-as.factor(as.character(sample(1:5, 150, replace = TRUE)))
n_na<-sample(1:150, 30)
iris[n_na, "fact_1"]<-NA
iris$fact_2<-as.factor(as.character(stri_rand_strings(150, 1, '[A-B]')))
iris$num<-runif(150, min = 0, max = 100)
n_na<-sample(1:150, 5)
iris[n_na, "num"]<-NA
iris_test<-descTab(iris, group = "Species", na.print = TRUE)
testParse<-parseClassFun(iris_test, levels_to_keep = list("fact_2" =  "A"),
group_rows_labels = list("Size" = c("Petal.Length", "Petal.Width"),
"My_f" = c("num", "fact_2")))
```

Var *S4 class initialization function*

Description

Initialization function for Var `initialize,Var-method()`

Usage

```
Var(name, type = "", normal = TRUE)
```

Arguments

`name`	A character taking name of the variable
`type`	A character taking name of the variable type
`normal`	Logical, if variable, is numeric; is it normal

Value

Var Object

Var-class *S4 class*

Description

A S4 class containing name, type and normality assessment of variable

Slots

`name` A character taking name of the variable

`type` A character taking name of the variable type

`normal` Logical, if variable, is numeric; is it normal

VarGroup-class *S4 class*

Description

A S4 class containing Var `initialize,Var-method()` It also contains the pvalue, the parsed value the missing values and the group for which it was calculated

Slots

`group_var` The subgroup for which proportions, mean/sd were calculated and missing values

`pvalue` The calculated pvalue

`parsed_name` The name of the variable parsed with the n (%), mean (SD)

`value` The values calculated parsed

`missing.value` Missing values numbers and proportions n (%)

`missing.value.name` Missing values concatenate with the level of the variable if it factor

[,parseClass-method *Method to access S4 Var elements*

Description

Method to access parseClass `initialize,parseClass-method()` elements by name

Usage

```
## S4 method for signature 'parseClass'
x[i]
```

Arguments

`x`	: Object
`i`	: Element name

Value

object

[,Var-method *Method to access S4 Var elements*

Description

Method to access Var elements by name

Usage

```
## S4 method for signature 'Var'
x[i]
```

Arguments

x	: object
i	: value

Value

object of Var

[,VarGroup-method *Method to access S4 Var elements*

Description

Method to access VarGroup initialize,VarGroup-method() elements by name

Usage

```
## S4 method for signature 'VarGroup'
x[i]
```

Arguments

x	: object
i	: value

Value

object element

[<-,parseClass-method *Method to modify S4 Var elements*

Description

Method to modify parseClass `initialize,parseClass-method()` elements by name

Usage

```
## S4 replacement method for signature 'parseClass'
x[i] <- value
```

Arguments

`x`	: Object
`i`	: Element name
`value`	: Value to be added

Value

parseClass Object

[<-,Var-method *Method to access S4 Var elements*

Description

Method to modify Var elements by name

Usage

```
## S4 replacement method for signature 'Var'
x[i] <- value
```

Arguments

`x`	: object
`i`	: Element name
`value`	: Value to be added

Value

object

`[<-,VarGroup-method` *Method to access S4 Var elements*

Description

Method to modify VarGroup `initialize,VarGroup-method()` elements by name

Usage

```
## S4 replacement method for signature 'VarGroup'
x[i] <- value
```

Arguments

`x`	Object
`i`	Element name
`value`	Value to be added

Value

object

Index