Package 'jmvcore'

December 10, 2018

Type Package

Title Dependencies for the 'jamovi' Framework

Version 0.9.5.2

Date 2018-11-04

Author Jonathon Love

Maintainer Jonathon Love <jon@thon.cc>

Description A framework for creating rich interactive analyses for the jamovi platform (see <https://www.jamovi.org> for more information).

URL https://www.jamovi.org

BugReports https://github.com/jamovi/jmvcore/issues

License GPL (>= 2)

ByteCompile yes

Depends R (>= 3.2)

Imports R6 (>= 1.0.1), rlang (>= 0.3.0.1), rjson, base64enc, stringi

Suggests testthat (>= 1.0.2), RProtoBuf, knitr, ggplot2, RColorBrewer

RoxygenNote 6.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2018-12-10 09:00:03 UTC

R topics documented:

`Analysis` *the jmvcore Object classes*

Description

the jmvcore Object classes

Usage

```
Analysis

Array

Column

Group

Html

Image

Preformatted

State

Table
```

Format

An object of class `R6ClassGenerator` of length 24.

`canBeNumeric` *Determines whether an object is or can be converted to numeric*

Description

Determines whether an object is or can be converted to numeric

Usage

```
canBeNumeric(object)
```

Arguments

`object`	the object

`Cell.BEGIN_GROUP` *Constants to specify formatting of Table cells*

Description

Cell.BEGIN_GROUP adds spacing above a cell

Usage

```
Cell.BEGIN_GROUP

Cell.END_GROUP

Cell.BEGIN_END_GROUP

Cell.NEGATIVE

Cell.INDENTED
```

Format

An object of class `numeric` of length 1.

Details

Cell.END_GROUP add spacing below a cell

Cell.BEGIN_END_GROUP add spacing above and below a cell

Cell.NEGATIVE specifies that the cells contents is negative

Examples

```
## Not run:

table$addFormat(rowNo=1, col=1, Cell.BEGIN_END_GROUP)

## End(Not run)
```

`colorPalette` *A function that creates a color palette*

Description

A function that creates a color palette

Usage

```
colorPalette(n = 5, pal = "jmv", type = "fill")
```

Arguments

`n`	Number of colors needed
`pal`	Color palette name
`type`	'fill' or 'color'

Value

a vector of hex color codes

`composeFormula` *Compose a formula string*

Description

Compose a formula string

Usage

```
composeFormula(lht, rht)
```

Arguments

`lht`	list of character vectors making up the left
`rht`	list of character vectors making up the right

Value

a string representation of the formula

Examples

```
composeFormula(list('a', 'b', c('a', 'b')))
# ~a+b+a:b

composeFormula('f', list('a', 'b', c('a', 'b')))
# "f~a+b+a:b"

composeFormula('with spaces', list('a', 'b', c('a', 'b')))
'`with spaces`~a+b+a:b'
```

`composeTerm` *Compose and decompose interaction terms to and from their components*

Description

Compose and decompose interaction terms to and from their components

Usage

```
composeTerm(components)

composeTerms(listOfComponents)

decomposeTerm(term)

decomposeTerms(terms)
```

Arguments

`components`	a character vectors of components
`listOfComponents`	a list of character vectors of components
`term`	a string with components separated with colons
`terms`	a character vector of components separated with colons

Examples

```
composeTerm(c('a', 'b', 'c'))
# 'a:b:c'

composeTerm(c('a', 'b', 'with space'))
# 'a:b:`with space`'

decomposeTerm('a:b:c')
# c('a', 'b', 'c')

decomposeTerm('a:b:`with space`')
# c('a', 'b', 'with space')
```

`constructFormula` *Construct a formula string*

Description

Construct a formula string

Usage

```
constructFormula(dep = NULL, terms)
```

Arguments

`dep`	the name of the dependent variable
`terms`	list of character vectors making up the terms

Value

a string representation of the formula

Examples

```
constructFormula(terms=list('a', 'b', c('a', 'b')))
# a+b+a:b

constructFormula('f', list('a', 'b', c('a', 'b')))
# "f~a+b+a:b"

constructFormula('with spaces', list('a', 'b', c('a', 'b')))
'`with spaces`~a+b+a:b'
```

create *Create an analysis*

Description

Used internally by jamovi

Usage

```
create(ns, name, optionsPB, datasetId, analysisId, revision)
```

Arguments

ns	package name
name	analysis name
optionsPB	options protobuf object
datasetId	dataset id
analysisId	analysis id
revision	revision

createError *Create and throw errors*

Description

These functions are convenience functions for creating and throwing errors.

Usage

```
createError(formats, code = NULL, ...)

reject(formats, code = NULL, ...)
```

Arguments

formats	a format string which is passed to format
code	an error code
...	additional arguments passed to format

extractErrorMessage *Extracts the error message from an error object*

Description

Extracts the error message from an error object

Usage

```
extractErrorMessage(error)
```

Arguments

error	an error object

format *Format a string with arguments*

Description

Substitutes the arguments into the argument str. See the examples below.

Usage

```
format(str, ..., context = "normal")
```

Arguments

str	the format string
...	the arguments to substitute into the string
context	'normal' or 'R'

Value

the resultant string

Examples

```
jmvcore::format('the {} was delish', 'fish')

# 'the fish was delish'

jmvcore::format('the {} was more delish than the {}', 'fish', 'cow')

# 'the fish was more delish than the cow'
```

```
jmvcore::format('the {1} was more delish than the {0}', 'fish', 'cow')

# 'the cow was more delish than the fish'

jmvcore::format('the {what} and the {which}', which='fish', what='cow')

# 'the cow and the fish'

jmvcore::format('that is simply not {}', TRUE)

# 'that is simply not true'

jmvcore::format('that is simply not {}', TRUE, context='R')

# 'that is simply not TRUE'
```

---

isError *Determine if an object is an error*

Description

Determine if an object is an error

Usage

```
isError(object)
```

Arguments

object	the object to test

Value

TRUE if the object is an error

---

naOmit *remove missing values from a data frame listwise*

Description

removes all rows from the data frame which contain missing values (NA)

Usage

```
naOmit(object)
```

Arguments

object	the object to remove missing values from

Details

this function is equivalent to na.omit from the stats package, however it preserves attributes on columns in data frames

---

Options *The jmv Options classes*

Description

The jmv Options classes

Usage

```
Options

OptionBool

OptionList

OptionNMXList

OptionVariables

OptionVariable

OptionTerms

OptionInteger

OptionNumber

OptionString

OptionLevel

OptionGroup

OptionSort

OptionArray

OptionPairs
```

Format

An object of class `R6ClassGenerator` of length 24.

`select` *Create a new data frame with only the selected columns*

Description

Shorthand equivalent to `subset(df, select=columnNames)`, however it additionally preserves attributes on the columns

Usage

```
select(df, columnNames)
```

Arguments

`df`	the data frame
`columnNames`	the names of the columns to make up the new data frame

Value

the new data frame

`sourcify` *Converts basic R object into their source representation*

Description

Converts basic R object into their source representation

Usage

```
sourcify(object, indent = "")
```

Arguments

`object`	the object to convert to source
`indent`	the level of indentation to use

Value

a string of the equivalent source code

Examples

```
sourcify(NULL)

# 'NULL'

sourcify(c(1,2,3))

# 'c(1,2,3)'

l <- list(a=7)
l[['b']] <- 3
l[['c']] <- list(d=3, e=4)
sourcify(l)

# 'list(
#      a=7,
#      b=3,
#      c=list(
#          d=3,
#          e=4))'
```

---

startsWith *Test whether strings start or end with a particular string*

Description

Same as `base::startsWith()` and `base::endsWith()` except available for R < 3.3

Usage

```
startsWith(x, prefix)

endsWith(x, suffix)
```

Arguments

`x`	a string to test
`prefix`	a string to test the presence of
`suffix`	a string to test the presence of

`stringifyTerm` *Converts a term into a string*

Description

Converts a term (a vector of components) into a string for display purposes

Usage

```
stringifyTerm(components, sep = getOption("jmvTermSep", ":"))
```

Arguments

`components`	a character vector of components
`sep`	a separator to go between the components

Value

the components joined together into a string for disply

Examples

```
stringifyTerm(c('a', 'b', 'c'))

# "a:b:c"

stringifyTerm(c('a', 'b', 'c'), sep=' * ')

# "a * b * c"

options('jmvTermSep', ' * ')
stringifyTerm(c('a', 'b', 'c'))

# "a * b * c"

#' stringifyTerm(c('`quoted`', 'b', 'c'))

# "quoted * b * c"
```

theme_default *Creates the default jmv ggplot2 theme*

Description

Creates the default jmv ggplot2 theme

Usage

```
theme_default(base_size = 16, scale = "none", palette = "jmv")
```

Arguments

base_size	Font size
scale	'none' or 'discrete'
palette	Color palette name

Value

the default jmv ggplot2 theme

theme_hadley *Creates the hadley jmv ggplot2 theme*

Description

Creates the hadley jmv ggplot2 theme

Usage

```
theme_hadley(base_size = 16, scale = "none", palette = "jmv")
```

Arguments

base_size	Font size
scale	'none' or 'discrete'
palette	Color palette name

Value

the hadley jmv ggplot2 theme

theme_min *Creates the minimal jmv ggplot2 theme*

Description

Creates the minimal jmv ggplot2 theme

Usage

```
theme_min(base_size = 16, scale = "none", palette = "jmv")
```

Arguments

base_size	Font size
scale	'none' or 'discrete'
palette	Color palette name

Value

the minimal jmv ggplot2 theme

theme_spss *Creates the spss jmv ggplot2 theme*

Description

Creates the spss jmv ggplot2 theme

Usage

```
theme_spss(base_size = 16, scale = "none", palette = "jmv")
```

Arguments

base_size	Font size
scale	'none' or 'discrete'
palette	Color palette name

Value

the spss jmv ggplot2 theme

toB64 *Convert names to and from Base64 encoding*

Description

Note: uses the . and _ characters rather than + and / allowing these to be used as variable names

Usage

```
toB64(names)

fromB64(names)
```

Arguments

names	the names to be converted base64

toNumeric *Converts a vector of values to numeric*

Description

Similar to as.numeric, however if the object has a values attribute attached, these are used as the numeric values

Usage

```
toNumeric(object)
```

Arguments

object	the vector to convert

`tryNaN` *try an expression, and return NaN on failure*

Description

if the expression fails, NaN is returned silently

Usage

```
tryNaN(expr)
```

Arguments

`expr`	an expression to evaluate

Value

the result, or NaN on failure

Index