

# Package ‘jmvcore’

May 28, 2019

**Type** Package

**Title** Dependencies for the 'jamovi' Framework

**Version** 1.0.0

**Date** 2019-05-28

**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** 2019-05-28 04:50:08 UTC

## R topics documented:

Analysis . . . . .	2
canBeNumeric . . . . .	3
Cell.BEGIN_GROUP . . . . .	3
colorPalette . . . . .	4
composeFormula . . . . .	5
composeTerm . . . . .	5
constructFormula . . . . .	6

create	7
createError	8
decomposeFormula	8
enquo	9
extractErrorMessage	9
format	10
isError	11
marshalData	11
marshalFormula	12
naOmit	12
Options	13
resolveQuo	14
select	14
sourcify	15
startsWith	16
stringifyTerm	16
theme_default	17
theme_hadley	18
theme_min	18
theme_spss	19
toB64	19
toNumeric	20
tryNaN	20

<b>Index</b>	<b>21</b>
--------------	-----------

---

Analysis	<i>the jmvcore Object classes</i>
----------	-----------------------------------

---

## 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	<i>Determines whether an object is or can be converted to numeric</i>
--------------	---

---

**Description**

Determines whether an object is or can be converted to numeric

**Usage**

canBeNumeric(object)

**Arguments**

object            the object

---

Cell.BEGIN_GROUP	<i>Constants to specify formatting of Table cells</i>
------------------	---

---

**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	<i>A function that creates a color palette</i>
--------------	--

---

**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	<i>Compose a formula string</i>
----------------	---------------------------------

---

**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	<i>Compose and decompose interaction terms to and from their components</i>
-------------	---

---

**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	<i>Construct a formula string</i>
------------------	-----------------------------------

---

**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	<i>Create and throw errors</i>
-------------	--------------------------------

---

**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 <a href="#">format</a>
code	an error code
...	additional arguments passed to <a href="#">format</a>

---

decomposeFormula	<i>Decompose a formula</i>
------------------	----------------------------

---

**Description**

Decompose a formula

**Usage**

```
decomposeFormula(formula)
```

**Arguments**

formula	the formula to decompose
---------	--------------------------

**Value**

a list of lists of the formulas components



---

enquo	<i>rlang::enquo</i> Simplifies things so packages overriding Analysis don't need to have rlang in their imports. This is intended for use by classes overriding Analysis
-------	--

---

**Description**

`rlang::enquo` Simplifies things so packages overriding Analysis don't need to have rlang in their imports. This is intended for use by classes overriding Analysis

**Usage**

```
enquo(arg)
```

**Arguments**

arg	the argument to enquote
-----	-------------------------

**Value**

the quosure

---

extractErrorMessage	<i>Extracts the error message from an error object</i>
---------------------	--

---

**Description**

Extracts the error message from an error object

**Usage**

```
extractErrorMessage(error)
```

**Arguments**

error	an error object
-------	-----------------

---

format	<i>Format a string with arguments</i>
--------	---------------------------------------

---

**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	<i>Determine if an object is an error</i>
---------	---

---

**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

---

marshalData	<i>Marshal the data from an environment into a data frame</i>
-------------	---

---

**Description**

Marshal the data from an environment into a data frame

**Usage**

```
marshalData(env, ...)
```

**Arguments**

env	the environment to marshal from
...	the variables to marshal

**Value**

a data frame

---

marshalFormula	<i>Marshal a formula into options</i>
----------------	---------------------------------------

---

**Description**

Marshal a formula into options

**Usage**

```
marshalFormula(formula, data, from = "rhs", type = "vars",
  permitted = c("numeric", "factor"), subset = ":", required = FALSE)
```

**Arguments**

formula	the formula
data	a data frame to marshal the data from
from	'rhs' or 'lhs', which side of the formula should be marshalled
type	'vars' or 'terms', the type of the option be marshalled to
permitted	the types of data the option permits
subset	a subset of the formula to marshal
required	whether this marshal is required or not
...	extra args for forwards compatibility

---

naOmit	<i>remove missing values from a data frame listwise</i>
--------	---

---

**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

OptionTerm

OptionVariable

OptionTerms

OptionInteger

OptionNumber

OptionString

OptionLevel

OptionGroup

OptionSort

OptionArray

OptionPairs

**Format**

An object of class R6ClassGenerator of length 24.

---

resolveQuo	<i>Evaluates a quosure This is intended for use by classes overriding Analysis</i>
------------	--

---

**Description**

Evaluates a quosure This is intended for use by classes overriding Analysis

**Usage**

```
resolveQuo(quo)
```

**Arguments**

quo                    the quosure to evaluate

**Value**

the value of the quosure

---

select	<i>Create a new data frame with only the selected columns</i>
--------	---

---

**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	<i>Converts basic R object into their source representation</i>
----------	---

---

**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	<i>Test whether strings start or end with a particular string</i>
------------	---

---

**Description**

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

**Usage**

```
startsWith(x, prefix)
```

```
endsWith(x, suffix)
```

**Arguments**

<code>x</code>	a string to test
<code>prefix</code>	a string to test the presence of
<code>suffix</code>	a string to test the presence of

---

stringifyTerm	<i>Converts a term into a string</i>
---------------	--------------------------------------

---

**Description**

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

**Usage**

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

**Arguments**

<code>components</code>	a character vector of components
<code>sep</code>	a separator to go between the components
<code>raise</code>	whether duplicates should be raised to powers

**Value**

the components joined together into a string for display



**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	<i>Creates the default jmv ggplot2 theme</i>
---------------	--

---

**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	<i>Creates the hadley jmv ggplot2 theme</i>
--------------	---

---

**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	<i>Creates the minimal jmv ggplot2 theme</i>
-----------	--

---

**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	<i>Creates the spss jmv ggplot2 theme</i>
------------	---

---

**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	<i>Convert names to and from Base64 encoding</i>
-------	--

---

**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	<i>Converts a vector of values to numeric</i>
-----------	---

---

**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	<i>try an expression, and return NaN on failure</i>
--------	---

---

**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

## \*Topic **datasets**

- Analysis, [2](#)
  - Cell.BEGIN\_GROUP, [3](#)
  - Options, [13](#)
- Analysis, [2](#)  
Array (Analysis), [2](#)  
as.numeric, [20](#)
- canBeNumeric, [3](#)  
Cell.BEGIN\_END\_GROUP  
    (Cell.BEGIN\_GROUP), [3](#)  
Cell.BEGIN\_GROUP, [3](#)  
Cell.END\_GROUP (Cell.BEGIN\_GROUP), [3](#)  
Cell.INDENTED (Cell.BEGIN\_GROUP), [3](#)  
Cell.NEGATIVE (Cell.BEGIN\_GROUP), [3](#)  
colorPalette, [4](#)  
Column (Analysis), [2](#)  
composeFormula, [5](#)  
composeTerm, [5](#)  
composeTerms (composeTerm), [5](#)  
constructFormula, [6](#)  
create, [7](#)  
createError, [8](#)
- decomposeFormula, [8](#)  
decomposeTerm (composeTerm), [5](#)  
decomposeTerms (composeTerm), [5](#)
- endsWith (startsWith), [16](#)  
enquo, [9](#)  
extractErrorMessage, [9](#)
- format, [8](#), [10](#)  
fromB64 (toB64), [19](#)
- Group (Analysis), [2](#)
- Html (Analysis), [2](#)
- Image (Analysis), [2](#)
- isError, [11](#)
- marshalData, [11](#)  
marshalFormula, [12](#)
- na.omit, [12](#)  
naOmit, [12](#)
- OptionArray (Options), [13](#)  
OptionBool (Options), [13](#)  
OptionGroup (Options), [13](#)  
OptionInteger (Options), [13](#)  
OptionLevel (Options), [13](#)  
OptionList (Options), [13](#)  
OptionNMXList (Options), [13](#)  
OptionNumber (Options), [13](#)  
OptionPairs (Options), [13](#)  
Options, [13](#)  
OptionSort (Options), [13](#)  
OptionString (Options), [13](#)  
OptionTerm (Options), [13](#)  
OptionTerms (Options), [13](#)  
OptionVariable (Options), [13](#)  
OptionVariables (Options), [13](#)
- Preformatted (Analysis), [2](#)
- reject (createError), [8](#)  
resolveQuo, [14](#)
- select, [14](#)  
sourcify, [15](#)  
startsWith, [16](#)  
State (Analysis), [2](#)  
stringifyTerm, [16](#)  
subset, [14](#)
- Table (Analysis), [2](#)  
theme\_default, [17](#)  
theme\_hadley, [18](#)  
theme\_min, [18](#)

theme\_spss, 19  
toB64, 19  
toNumeric, 20  
tryNaN, 20