

Package ‘jmvcore’

September 24, 2026

Type Package

Title Dependencies for the 'jamovi' Framework

Version 28.3

Date 2026-09-24

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/jamovi/issues>

License GPL (>= 2)

ByteCompile yes

Depends R (>= 3.2)

Imports R6 (>= 1.0.1), rlang (>= 0.3.0.1), jsonlite, base64enc

Suggests testthat (>= 1.0.2), RProtoBuf, knitr, commonmark, ggplot2, RColorBrewer, ragg, fastmap, export, jmvReadWrite

Encoding UTF-8

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Jonathon Love [aut, cre, cph]

Repository CRAN

Date/Publication 2026-09-24 07:30:02 UTC

Contents

jmvcore-package	2
.	4
.. . . .	4
Action	4
asFormula	5

canBeNumeric	5
Cell.BEGIN_GROUP	6
colorPalette	6
composeFormula	7
composeTerm	8
constructFormula	9
create	9
createError	10
decomposeFormula	10
extractErrorMessage	11
format	11
htmlEscape	12
isError	13
marshalData	13
marshalFormula	14
matchSet	14
naOmit	15
NoticeType	15
Options	15
resolveQuo	16
select	16
sourcify	17
startsWith	18
stringifyTerm	18
theme_default	19
theme_hadley	20
theme_min	20
theme_spss	21
toB64	21
toNumeric	22
tryNaN	22
validateSafeFormula	23
Index	24

jmvcore-package

jmvcore: Dependencies for the 'jamovi' Framework

Description

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

Number formatting

Numbers in results tables are formatted as jamovi formats them: each column is given enough decimal places for its smallest value to have the required number of significant figures, p-values are given to a fixed number of decimal places (and smaller ones are shown as $< .001$), and integers are shown as they are.

This is set with the `jmvcore.format` option, a list of:

- `style`: 'sf' (significant figures) or 'dp' (decimal places), for values. Default 'sf'
- `digits`: how many, for values. Default 4
- `pStyle`: 'sf' or 'dp', for p-values. Default 'dp'
- `pDigits`: how many, for p-values. Default 3

Any of these may be given, and the rest take their defaults. Setting the option replaces any earlier setting, rather than adding to it. The defaults are jamovi's, but with a significant figure more for values (`digits=3` gives jamovi's own formatting).

The decimal symbol is taken from R's `OutDec` option.

Quarto and R Markdown

When a document is knitted to html, results are rendered as html, as jamovi exports them: tables, headings, notes and plots. Other output formats get the same text as the console.

A chunk may set its own number format, over `jmvcore.format`, with the `jmv-format` chunk option:

```
#| jmv-format: {digits: 5}
jmv::ttestIS(ToothGrowth, vars='len', group='supp')
```

Author(s)

Maintainer: Jonathon Love <jon@thon.cc> [copyright holder]

Authors:

- Jonathon Love <jon@thon.cc> [copyright holder]

See Also

Useful links:

- <https://www.jamovi.org>
- Report bugs at <https://github.com/jamovi/jamovi/issues>

Examples

```
## Not run:
# 5 significant figures, and p-values to 4 decimal places
options(jmvcore.format=list(digits=5, pDigits=4))

# jamovi's own formatting
```

```
options(jmvcore.format=list(digits=3))

# back to the defaults
options(jmvcore.format=NULL)

## End(Not run)
```

.	<i>Designate string as translated</i>
---	---------------------------------------

Description

Designate string as translated

Usage

```
.(text, n = 1)
```

Arguments

text	the string to translate
n	the number (if applicable) for pluralisation

..	<i>Designate string as client-side translated</i>
----	---

Description

Designate string as client-side translated

Usage

```
..(format, values = NULL)
```

Arguments

format	a format string
values	a string or named list of strings to substitute

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

Description

the jmvcore Object classes

asFormula	<i>Convert and validate an object as a formula</i>
-----------	--

Description

Accepts formula-like input, converts it with [stats::as.formula()], validates it using [validateSafeFormula()], and returns the formula.

Usage

```
asFormula(object, env = parent.frame(), additional_allowed_functions = NULL)
```

Arguments

object	A formula-like object (typically a character string or formula).
env	The environment to use when parsing character formulas.
additional_allowed_functions	Optional character vector of extra function names to permit for this conversion and validation call.

Value

A validated formula object.

Note

Available from jamovi 2.7.27.

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

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	<i>Create an analysis</i>
--------	---------------------------

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 format
code	an error code
...	additional arguments passed to format

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

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'
```

htmlEscape*Escape HTML special characters*

Description

Escape HTML special characters

Usage

```
htmlEscape(text, attribute = FALSE)
```

Arguments

text	a character vector to escape
attribute	logical; if TRUE, also escapes quotes and newlines for use in HTML attribute values

Value

the escaped character vector

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

matchSet	<i>Determines the index where an item appears</i>
----------	---

Description

Determines the index where an item appears

Usage

```
matchSet(x, table)
```

Arguments

x	the item to find
table	the object to search

Value

the index of where the item appears, or -1 if it isn't present

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

NoticeType	<i>Different notice levels</i>
------------	--------------------------------

Description

Different notice levels

Usage

```
NoticeType
```

Options	<i>The jmv Options classes</i>
---------	--------------------------------

Description

The jmv Options classes

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 and the data frame

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

<code>object</code>	the object to convert to source
<code>indent</code>	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

validateSafeFormula	<i>Secure formula validation using allowlist approach Prevents code injection while allowing legitimate statistical formulas</i>
---------------------	--

Description

Secure formula validation using allowlist approach Prevents code injection while allowing legitimate statistical formulas

Usage

```
validateSafeFormula(fmla, additional_allowed_functions = NULL)
```

Arguments

fmla	A formula object to validate.
additional_allowed_functions	Optional character vector of extra function names to permit for this validation call.

Value

Invisibly returns 'NULL'. Throws an error if the formula contains security violations.

Examples

```
# Safe usage:
validateSafeFormula(y ~ x + z)
validateSafeFormula(y ~ I(x^2) + log(z))

## Not run:
# Blocks dangerous code (throws an error):
validateSafeFormula(y ~ I(system('whoami'))))

## End(Not run)
```

Index

..[4](#)
...[4](#)

Action, [4](#)
Analysis (Action), [4](#)
Array (Action), [4](#)
as.numeric, [22](#)
asFormula, [5](#)

canBeNumeric, [5](#)
Cell.BEGIN_END_GROUP
 (Cell.BEGIN_GROUP), [6](#)
Cell.BEGIN_GROUP, [6](#)
Cell.END_GROUP (Cell.BEGIN_GROUP), [6](#)
Cell.INDENTED (Cell.BEGIN_GROUP), [6](#)
Cell.NEGATIVE (Cell.BEGIN_GROUP), [6](#)
colorPalette, [6](#)
Column (Action), [4](#)
composeFormula, [7](#)
composeTerm, [8](#)
composeTerms (composeTerm), [8](#)
constructFormula, [9](#)
create, [9](#)
createError, [10](#)

decomposeFormula, [10](#)
decomposeTerm (composeTerm), [8](#)
decomposeTerms (composeTerm), [8](#)

endsWith (startsWith), [18](#)
extractErrorMessage, [11](#)

format, [10](#), [11](#)
fromB64 (toB64), [21](#)

Group (Action), [4](#)

Html (Action), [4](#)
htmlEscape, [12](#)

Image (Action), [4](#)

isError, [13](#)

jmvcore (jmvcore-package), [2](#)
jmvcore-package, [2](#)

marshalData, [13](#)
marshalFormula, [14](#)
matchSet, [14](#)

na.omit, [15](#)
naOmit, [15](#)
Notice (Action), [4](#)
NoticeType, [15](#)

OptionAction (Options), [15](#)
OptionArray (Options), [15](#)
OptionBool (Options), [15](#)
OptionFile (Options), [15](#)
OptionGroup (Options), [15](#)
OptionInteger (Options), [15](#)
OptionLevel (Options), [15](#)
OptionList (Options), [15](#)
OptionNMXList (Options), [15](#)
OptionNumber (Options), [15](#)
OptionOutput (Options), [15](#)
OptionPair (Options), [15](#)
OptionPairs (Options), [15](#)
Options, [15](#)
OptionSort (Options), [15](#)
OptionString (Options), [15](#)
OptionTerm (Options), [15](#)
OptionTerms (Options), [15](#)
OptionVariable (Options), [15](#)
OptionVariables (Options), [15](#)
Output (Action), [4](#)

Preformatted (Action), [4](#)

reject (createError), [10](#)
resolveQuo, [16](#)

`select`, [16](#)
`sourcify`, [17](#)
`startsWith`, [18](#)
`State (Action)`, [4](#)
`stringifyTerm`, [18](#)
`subset`, [16](#)
`Svg (Action)`, [4](#)

`Table (Action)`, [4](#)
`Text (Action)`, [4](#)
`theme_default`, [19](#)
`theme_hadley`, [20](#)
`theme_min`, [20](#)
`theme_spss`, [21](#)
`toB64`, [21](#)
`toNumeric`, [22](#)
`tryNaN`, [22](#)

`validateSafeFormula`, [23](#)