Package ‘formula.tools’

September 21, 2017

Type Package

Title Programmatic Utilities for Manipulating Formulas, Expressions, Calls, Assignments and Other R Objects

Version 1.6.8

Date 2017-09-21

Description These utilities facilitate the programmatic manipulations of formulas, expressions, calls, assignments and other R language objects. These objects all share the same structure: a left-hand side, operator and right-hand side. This package provides methods for accessing and modifying this structures as well as extracting and replacing names and symbols from these objects.

Depends R (>= 3.0.0)

Imports operator.tools(>= 1.4.0), utils, methods

Suggests magrittr, testthat

License GPL-2 | file LICENSE

URL https://github.com/decisionpatterns/formula.tools

BugReports https://github.com/decisionpatterns/formula.tools/issues

LazyLoad yes

LazyData yes

Collate 'as.character.formula.R' 'catcont.R' 'env.R' 'get.vars.R'
'invert.R' 'is.one.sided.R' 'is.two.sided.R' 'lhs.vars.R'
'op.type.R' 'parts.R' 'parts.lhs.get.R' 'parts.lhs.set.R'
'parts.op.get.R' 'parts.op.set.R' 'parts.rhs.get.R'
'parts.rhs.set.R' 'rhs.vars.R' 'terms.R' 'zzz.R'

RoxygenNote 6.0.1.9000

Repository CRAN

NeedsCompilation no

Author Christopher Brown [aut, cre],
Decision Patterns [cph]

Maintainer Christopher Brown <chris.brown@decisionpatterns.com>

Date/Publication 2017-09-21 19:29:49 UTC
R topics documented:

.invert.single ......................................................... 2
as.character.formula .................................................. 2
env ................................................................. 3
formula.parts .......................................................... 4
get.vars ............................................................... 8
invert ................................................................. 10
is.cat ................................................................. 11
is.one.sided ........................................................... 13
op.type ............................................................... 15
terms.call ............................................................ 16

Index

.invert.single  Invert multiple elements of a multiple element object

Description
Invert multiple elements of a multiple element object

Usage
.invert.single(x)
.invert.plural(x)

Arguments
x object to invert from

See Also
.invert.single

as.character.formula  Converts a formula to character

Description
Convers a formula to character representaion

Usage
## S3 method for class 'formula'
as.character(x, ...)

as.character.formula
Arguments

x  formula object

Details

Coerces formula to a character by deparsing.

Value

A character vector

Author(s)

Christopher Brown

See Also

deparse

Examples

as.character( y ~ mx + b )

## The function is currently defined as
function(x)
  Reduce( paste, deparse(x) )
Details

S3 returns the environment associated with an object.
For a formula object, 'env' returns the environment in the '.Environment' attribute.

Value

Environment

Examples

eval( env( lhs ~ rhs ) )

formula.parts
Manipulate the component parts of formulas, expressions, calls, name/symbols and list and vectors of such objects.

Description

lhs, rhs, op, and op.type retrieve the various parts of R formulas, calls, expressions, names/symbols. These functions were designed to greatly facilitate symbolic manipulation using native R objects. Also provided are methods to handle list of these objects.

Usage

lhs(x, ...)

## S4 method for signature 'call'
lhs(x)

## S4 method for signature 'formula'
lhs(x)

## S4 method for signature 'expression'
lhs(x, ...)

## S4 method for signature 'list'
lhs(x, ...)

lhs(x) <- value

## S4 replacement method for signature 'call'
lhs(x) <- value
## S4 replacement method for signature 'formula'

```r
lhs(x) <- value
```

```r
.replace.lhs.plural(x, value)
```

## S4 replacement method for signature 'expression'

```r
lhs(x) <- value
```

## S4 replacement method for signature 'list'

```r
lhs(x) <- value
```

```r
op(x)
```

## S4 method for signature 'formula'

```r
op(x)
```

## S4 method for signature 'call'

```r
op(x)
```

## S4 method for signature 'name'

```r
op(x)
```

## S4 method for signature 'expression'

```r
op(x)
```

## S4 method for signature 'list'

```r
op(x)
```

```r
op(x) <- value
```

## S4 replacement method for signature 'call'

```r
op(x) <- value
```

## S4 replacement method for signature 'formula'

```r
op(x) <- value
```

```r
.replace.op.plural(x, value)
```

## S4 replacement method for signature 'expression'

```r
op(x) <- value
```

## S4 replacement method for signature 'list'

```r
op(x) <- value
```
op(x) <- value

rhs(x, ...)

.rhs.singular(x)

## S4 method for signature 'call'

rhs(x)

## S4 method for signature 'formula'

rhs(x)

## S4 method for signature 'expression'

rhs(x, ...)

## S4 method for signature 'list'

rhs(x, ...)

rhs(x) <- value

.replace.rhs.singular(x, value)

## S4 replacement method for signature 'call'

rhs(x) <- value

## S4 replacement method for signature 'formula'

rhs(x) <- value

.replace.rhs.plural(x, value)

## S4 replacement method for signature 'expression'

rhs(x) <- value

## S4 replacement method for signature 'list'

rhs(x) <- value

Arguments

x object from where to get/set the lhs/rhs

... arguments passed to additional methods

value the value to set for the lhs/rhs

Details

lhs retrieves the left-hand side

rhs retrieves the right-hand side

op retrieves the operation

op.type
returns the type operator
There are also functions \texttt{lhs.vars} and \texttt{rhs.vars}. Like \texttt{all.vars}, these functions interpret the variables on the left-hand and right-hand sides respectively.

These are simple functions for extracting the left-hand side, right-hand side, operator and operator type from formulas, expressions, calls, names/symbols and list containing these objects. \texttt{lhs}, \texttt{rhs} are only defined for formulas and calls (and list and expressions) that are defined with either one of the relational or tilde ('\~') operators. If the object does not contain one of these operators, it will fail with a warning.

The defined operator types are defined by the \texttt{operator.tools} package: See \texttt{operators} and \texttt{setOperator}

The \texttt{lhs.vars} and \texttt{rhs.vars} methods, return the variables used on the lhs and rhs, respectively. If special formula variables are used, such as '.', a data.frame or environment must also be provided such that the variable list may be properly infered.

\textbf{Value}

Value depends on the argument.

\textbf{Note}

Methods for the non-standard "<-" class exist and are not included in the usage documentation because CRAN does not support S4 documentation for this class.

\textbf{Author(s)}

Christopher Brown

\textbf{See Also}

terms, \texttt{all.vars}, \texttt{all.names}, \texttt{operators}

\textbf{Examples}

```r
# FORMULA
f <- A + B ~ C + D
lhs(f)
lhs(f) <- quote(E / F )

rhs(f)
rhs(f) <- quote(G + H )
op(f)
op(rhs(f))
op(quote(A)) # NULL:
op.type(f)

# ONE-SIDED FORMULA
f <- ~ A  #
lhs(f)    # NULL
rhs(f)    # A
```
get.vars

Get variable (names) from various R objects

Description

get.vars extracts variable names from various R objects such as formulas, expressions, calls, symbols, etc. It is very similar to all.vars except that all symbols, etc. are interpolated to the names of variables.

Usage

get.vars(x, data = NULL, ...)

## S4 method for signature 'formula,ANY'
get.vars(x, data = NULL, ...)

## S4 method for signature 'call,ANY'
get.vars(x, data = NULL, ...)

## S4 method for signature 'expression,missing'
get.vars(x, data = NULL, ...)

## S4 method for signature 'name,ANY'
get.vars(x, data = NULL, ...)

## S4 method for signature 'ANY,ANY'
get.vars(x, data = NULL, ...)

## S4 method for signature '\texttt{\textquoteinoisu}',ANY'
get.vars(x, data = NULL, ...)

lhs.vars(x, ...)

.lhs.vars(x, ..., data = NULL)

## S4 method for signature 'formula'
lhs.vars(x, ..., data = NULL)

## S4 method for signature 'call'
lhs.vars(x, ..., data = NULL)

## S4 method for signature 'expression'
lhs.vars(x, ...)

rhs.vars(x, ...)

.rhs.vars(x, ..., data = NULL)

## S4 method for signature 'formula'
rhs.vars(x, ..., data = NULL)

## S4 method for signature 'call'
rhs.vars(x, ..., data = NULL)

## S4 method for signature 'expression'
rhs.vars(x, ...)

Arguments

\texttt{x} \hspace{1cm} \text{object to extract vars from.}

\texttt{data} \hspace{1cm} \text{data set/list or environment on which the names are defined}

\texttt{...} \hspace{1cm} \text{arguments passed to subsequent functions}

get.vars and variant get the variables from objects optionally interpreting on . on the data. This is useful, for example, when you wish to know what data is used based on a given formula.

Methods/functions beginning with . are not exported

Value

character vector of variables names in order that they appear in \texttt{x}. 
See Also

all.vars

Examples

get.vars( Species ~ ., iris)
get.vars( quote( Sepal.Length * Sepal.Width ), iris)

Description

Invert the operators in an object, usually a formula or expression

Usage

invert(x, ...)

## S4 method for signature 'call'
invert(x)

## S4 method for signature 'expression'
invert(x)

Arguments

x function for invert
... additional arguments passed other functions

invert is a S4 generic method for inverting relational operators, i.e. functions prefixed with a . are not exported and should probably not be called directly

Value

The operand is returned with the relational operators inverted.

Author(s)

Christopher Brown

See Also

op, op.type
Examples

invert( quote( A > 5 ) )
invert( quote( A >= 5 ) )
invert( quote( A < 5 ) )
invert( quote( A <= 5 ) )
invert( quote( A == 5 ) )
invert( quote( A != 5 ) )
invert( quote( A %in% letters[1:5] ) )
invert( quote( A %!in% letters[1:5] ) )

Description

These functions are used to identify which/if a variable or variables are categorical or continous. is.cat and is.cont take single variable arguments. which.cat and which.cont take a list or data.frame or any structures whose elements have a class method.

Usage

is.cat(x, ...)

## S4 method for signature 'character'
is.cat(x)

## S4 method for signature 'factor'
is.cat(x)

## S4 method for signature 'logical'
is.cat(x)

## S4 method for signature 'ANY'
is.cat(x)

is.cont(x, ...)

## S4 method for signature 'numeric'
is.cont(x)

## S4 method for signature 'integer'
is.cont(x)

## S4 method for signature 'complex'
is.cont(x)
Arguments

- **x**: object
- **...**: arguments passed to other functions
- **names**: logical; whether to return the names of the variables instead of their index

Details

These functions facilitate working with variables as categorical or continuous rather than logical, integer, numeric, factor, character, ..

Value

- `is.cat` returns `TRUE` if `x` is character, factor or logical. It is `FALSE` otherwise.
- `is.cont` returns `TRUE` if `x` is numeric, integer, complex, Date, POSIXct

which.cat and which.cont report which variables in an object are categorical and which are continuous. By default, integer indices are return. If `names=TRUE`, the names of the variables are returned instead.

See Also

- `which`, `is`

Examples

```r
## Not run:
is.cat(letters)       # TRUE
is.cat(factor(letters)) # TRUE
is.cat(TRUE)          # TRUE
is.cat(FALSE)         # TRUE
is.cat(1:10)          # FALSE
is.cat(rnorm(10))     # FALSE
is.cat( now() )       # FALSE
```
is.one.sided

```
is.cont(letters)       # FALSE
is.cont(factor(letters)) # FALSE
is.cont(TRUE)          # FALSE
is.cont(FALSE)         # FALSE
is.cont(1:10)          # TRUE
is.cont(rnorm(10))     # TRUE

which.cat(iris)
which.cat(iris, names=TRUE)

which.cont(iris)
which.cont(iris, names=TRUE)
```

## End(Not run)

---

**is.one.sided**  
Determine if an object is one- or two-sided. Test whether a object (typically formula, call or expression) is one- (e.g. ~x) or two-sided (e.g. x~y).

**Description**

Determine if an object is one- or two-sided.  
Test whether a object (typically formula, call or expression) is one- (e.g. ~x) or two-sided (e.g. x~y).

**Usage**

```
is.one.sided(x, ...)

## S4 method for signature 'formula'
is.one.sided(x, ...)

## S4 method for signature 'call'
is.one.sided(x, ...)

## S4 method for signature 'expression'
is.one.sided(x, ...)

## S4 method for signature 'list'
is.one.sided(x, ...)

## S4 method for signature 'ANY'
is.one.sided(x, ...)
```
is.two.sided(x, ...)

## S4 method for signature 'formula'
is.two.sided(x, ...)

## S4 method for signature 'call'
is.two.sided(x, ...)

## S4 method for signature 'expression'
is.two.sided(x, ...)

## S4 method for signature 'list'
is.two.sided(x, ...)

## S4 method for signature 'ANY'
is.two.sided(x, ...)

Arguments

x object to test for one-sidedness.
...
arguments passed to called functions

Details

These functions detect whether the formula is single- (unary) or double- sided. They work on formulas, expression, calls, assignments, etc.

is.single.sided and is.unary are alias for is.single.sided. is.double.sided and is.binary are aliases for is.two.sided.

Value

logical; whether x is an object is one-sided or two-sided formula.

Note

Methods for the "<-" class exist and are not included in the usage documentation because CRAN does not support S4 documentation for this class.

Examples

form <- y ~ x

is.one.sided(form)
# is.single.sided(form)
# is.unary(form)

is.two.sided(form)
op.type

# is.double.sided(form)
# is.binary(form)

---

**op.type**  
*Get the operator type used in an call, formula, expression, etc.*

### Description
Get the operator type used in an call, formula, expression, etc.

#### Usage
`op.type(x)`

```r
## S4 method for signature 'call'
op.type(x)
```

```r
## S4 method for signature 'formula'
op.type(x)
```

```r
## S4 method for signature 'ANY'
op.type(x)
```

```r
## S4 method for signature 'expression'
op.type(x)
```

```r
## S4 method for signature 'list'
op.type(x)
```

#### Arguments
- `x`: object from which to extract the operator type

#### Value
- a character vector of the operator type(s)

#### See Also
- `op`, `operator.type`
Description

terms method for call and expression objects

Usage

```r
## S3 method for class 'call'
terms(x, ...)

## S3 method for class 'expression'
terms(x, ...)
```

Arguments

- `x`: A call object
- `...`: Arguments passed to `terms.formula`

This S3 method returns a terms object for a call methods using a dispatch to `terms.formula`. The terms are generated by making a rhs only call to `terms.formula`. `data` is only needed and must be explicitly specified, i.e. `data` if there are special elements such as `.`. Otherwise the `data` argument is unused. Some edge cases may not be supported.

Value

A terms object. See `terms.object` for details.

Author(s)

Christopher Brown

See Also

`terms.object` and `terms.formula`

Examples

```r
terms(quote( A + B ) )
data(iris)
x <- terms(quote( . - Species ), data=iris)
```
# Index

*Topic **manip**
  - as.character.formula, 2
  - terms.call, 16

*Topic **symbolmath**
  - terms.call, 16

*Topic **utilities**
  - as.character.formula, 2
  - terms.call, 16

.lhs.vars (get.vars), 8
.replacement.lhs.plural (formula.parts), 4
.replacement.lhs.plural (formula.parts), 4
.replacement.rhs.plural (formula.parts), 4
.replacement.rhs.singular (formula.parts), 4
.replacement.rhs.singular (formula.parts), 4
.rhs.singular (formula.parts), 4
.rhs.vars (get.vars), 8

.isNoneNsided, <-method (formula.parts), 4

.all.vars, 7, 8, 10
.as.character (as.character.formula), 2
.as.character.formula, 2
.deparse, 3
.env, 3

.formula.parts, 4

get.vars, 8
get.vars, ANY, ANY-method (get.vars), 8
get.vars, ANY, ANY-methods (get.vars), 8
get.vars, call, ANY-method (get.vars), 8
get.vars, expression, missing-method (get.vars), 8
get.vars, formula, ANY-method (get.vars), 8
get.vars, name, ANY-method (get.vars), 8
get.vars, NULL, ANY-method (get.vars), 8
get.vars, NULL, ANY-methods (get.vars), 8

.invert, 10
.invert, call-method (invert), 10
.invert, expression-method (invert), 10
.is, 12
.is.cat, 11
.is.cat, ANY-method (is.cat), 11
.is.cat, character-method (is.cat), 11
.is.cat, factor-method (is.cat), 11
.is.cat, logical-method (is.cat), 11
.is.cont (is.cat), 11
.is.cont, ANY-method (is.cat), 11
.is.cont, complex-method (is.cat), 11
.is.cont, Date-method (is.cat), 11
.is.cont, factor-method (is.cat), 11
.is.cont, integer-method (is.cat), 11
.is.cont, numeric-method (is.cat), 11
.is.cont, POSIXct-method (is.cat), 11
.is.one.sided, 13
.is.one.sided, <--method (is.one.sided), 13
.is.one.sided, ANY-method (is.one.sided), 13
.is.one.sided, call-method (is.one.sided), 13
.is.one.sided, expression-method (is.one.sided), 13
.is.one.sided, formula-method (is.one.sided), 13
.is.one.sided, list-method (is.one.sided), 13
.is.two.sided (is.one.sided), 13
.is.two.sided, <--method (is.one.sided), 13
.is.two.sided, ANY-method (is.one.sided), 13
.is.two.sided, call-method (is.one.sided), 13
.is.two.sided, expression-method (is.one.sided), 13
is.two.sided, formula-method
   (is.one.sided), 13
is.two.sided, list-method
   (is.one.sided), 13

lhs (formula.parts), 4
lhs,<-method (formula.parts), 4
lhs, call-method (formula.parts), 4
lhs, expression-method (formula.parts), 4
lhs, formula-method (formula.parts), 4
lhs, list-method (formula.parts), 4
lhs vars (get.vars), 8
lhs vars, call-method (get.vars), 8
lhs vars, expression-method (get.vars), 8
lhs <- (formula.parts), 4
lhs <-,<-method (formula.parts), 4
lhs <-, call-method (formula.parts), 4
lhs <-, expression-method (formula.parts), 4
lhs <-, formula-method (formula.parts), 4
lhs <-, list-method (formula.parts), 4

op, 10, 15
op (formula.parts), 4
op,<-method (formula.parts), 4
op, call-method (formula.parts), 4
op, expression-method (formula.parts), 4
op, formula-method (formula.parts), 4
op, list-method (formula.parts), 4
op, name-method (formula.parts), 4
op.type, 10, 15
op.type, <-method (op.type), 15
op.type, ANY-method (op.type), 15
op.type, call-method (op.type), 15
op.type, expression-method (op.type), 15
op.type, formula-method (op.type), 15
op.type, list-method (op.type), 15
op.type, methods (op.type), 15
op <- (formula.parts), 4
op <-,<-method (formula.parts), 4
op <-, call-method (formula.parts), 4
op <-, expression-method (formula.parts), 4
op <-, formula-method (formula.parts), 4
op <-, list-method (formula.parts), 4
operator.type, 15
operators, 7

rhs (formula.parts), 4
 rhs,<-method (formula.parts), 4
 rhs, call-method (formula.parts), 4
 rhs, expression-method (formula.parts), 4
 rhs, formula-method (formula.parts), 4
 rhs, list-method (formula.parts), 4
 rhs vars (get.vars), 8
 rhs vars, call-method (get.vars), 8
 rhs vars, expression-method (get.vars), 8
 rhs <- (formula.parts), 4
 rhs <-,<-method (formula.parts), 4
 rhs <-, call-method (formula.parts), 4
 rhs <-, expression-method (formula.parts), 4
 rhs <-, formula-method (formula.parts), 4
 rhs <-, list-method (formula.parts), 4

setOperator, 7

terms (terms.call), 16
terms.call, 16
terms.formula, 16
terms.object, 16

which, 12
which.cat (is.cat), 11
which.cont (is.cat), 11