Package 'statnet.common'

October 24, 2015

Version 3.3.0

Date 2015-10-23

Title Common R Scripts and Utilities Used by the Statnet Project Software

Description Non-statistical utilities used by the software developed by the Statnet Project. They may also be of use to others.

Imports utils

License GPL-3 + file LICENSE

URL http://www.statnet.org

NeedsCompilation no

Author Pavel N. Krivitsky [aut, cre], Skye Bender-deMoll [ctb]

Maintainer Pavel N. Krivitsky <pavel@uow.edu.au>

Repository CRAN

Date/Publication 2015-10-24 11:50:36

R topics documented:

check.control.class 2
compress.data.frame 2
ERRVL 3
formula.utils 4
NVL 5
opttest 6
paste.and 7
print.control.list 8
set.control.class 8
sort.data.frame 9
statnet.cite 9
statnetStartupMessage 10
vector.namesmatch 11
$.control.list 12

Index **14**

`check.control.class` *Check if the class of the control list is one of those that can be used by the calling function*

Description

This function can be called to check that the control list passed is appropriate for the function to be controlled. It does so by looking up the class of the `control` argument (defaulting to the `control` variable in the calling function) and checking if it matches a list of acceptable classes (defaulting to the name of the calling function with "control." prepended).

Usage

```
check.control.class(OKnames = {
    sc <- sys.calls()
    as.character(sc[[length(sc) - 1]][[1]])
}, myname = {
    sc <- sys.calls()
    as.character(sc[[length(sc) - 1]][[1]])
}, control = get("control", pos = parent.frame()))
```

Arguments

`OKnames`	List of control function names which are acceptable.
`myname`	Name of the calling function (used in the error message).
`control`	The control list. Defaults to the `control` variable in the calling function.

See Also

set.control.class, print.control.list

`compress.data.frame` *"Compress" a data frame*

Description

This function "compresses" a data frame, returning unique rows and a tally of the number of times each row is repeated

Usage

```
compress.data.frame(x)
```

Arguments

x	A data frame.

Value

A list with two elements:

rows	Unique rows of x
frequencies	A vector of the same length as the number or rows, giving the number of times the corresponding row is repeated

See Also

data.frame

Examples

```
data(faithful)

head(faithful)

lapply(compress.data.frame(faithful),head)
```

ERRVL — *Return the first argument passed (out of any number) that is not a* try-error *(result of* try *encountering an error.*

Description

This function is inspired by NVL, and simply returns the first argument that is not a try-error, raising an error if all arguments are try-errors.

Usage

```
ERRVL(...)
```

Arguments

...	Expressions to be tested; usually outputs of try.

Value

The first argument that is not a try-error. Stops with an error if all are.

See Also

try, inherits

Examples

```
print(ERRVL(1,2,3)) # 1
print(ERRVL(try(solve(0)),2,3)) # 2
```

`formula.utils` *Functions for Querying, Validating and Extracting from ERGM Formulas*

Description

These are all functions that are generally not called directly by users, but may be employed by other depending packages.

Usage

```
nonsimp.update.formula(object, new, ..., from.new = FALSE)

term.list.formula(rhs, sign = +1)

append.rhs.formula(object, newterms, keep.onesided = FALSE)
```

Arguments

`object`	formula object to be updated
`new`	new formula to be used in updating
`from.new`	logical or character vector of variable names. controls how environment of formula gets updated.
`rhs`	a formula-style call containing the right hand side of formula, obtained by `fmla[[3]]` for a two-sided formula and `fmla[[2]]` for a one-sided formula.
`sign`	an internal parameter used by `term.list.formula` when calling itself recursively
`newterms`	list of terms (names) to append to the formula
`keep.onesided`	if the initial formula is one-sided, keep it whether to keep it one-sided or whether to make the initial formula the new LHS
`...`	Additional arguments. Currently unused.

Details

- `nonsimp.update.formula` is a reimplementation of `update.formula` that does not simplify. Note that the resulting formula's environment is set as follows. If `from.new==FALSE`, it is set to that of object. Otherwise, a new sub-environment of object, containing, in addition, variables in new listed in from.new (if a character vector) or all of new (if TRUE).
- `term.list.formula` returns a list containing terms in a given formula, handling `+` and `-` operators and parentheses, and keeping track of whether a term has a plus or a minus sign.
- `append.rhs.formula` appends a list of terms to the RHS of a formula. If the formula is one-sided, the RHS becomes the LHS, if `keep.onesided==FALSE` (the default).

Value

terms.list.formula returns a list of formula terms, each of witch having an additional attribute "sign".

nonsimp.update.formula and append.rhs.formula each return an updated formula object

Examples

```
## append.rhs.formula

append.rhs.formula(y~x,list(as.name("z1"),as.name("z2"))) # y~x+z1+z2
append.rhs.formula(~y,list(as.name("z"))) # y~z
append.rhs.formula(~y+x,list(as.name("z"))) # y+x~z
append.rhs.formula(~y,list(as.name("z")),TRUE) # ~y+z
```

NVL *Return the first argument passed (out of any number) that is not* NULL.

Description

This function is inspired by SQL function NVL, and simply returns the first argument that is not NULL, or NULL if all arguments are NULL.

Usage

```
NVL(...)
```

Arguments

...	Expressions to be tested.

Details

Note that an earlier version of this function took only two arguments: EXPR, that was tested and returned if not NULL and NULLV, which was returned if EXPR was NULL. The new version produces identical output for the same (two-argument) input, but tests any number of expressions sequentially.

Value

The first argument that is not NULL, or NULL if all are.

See Also

is.null, if

Examples

```
a <- NULL

print(a) # NULL
print(NVL(a,0)) # 0

b <- 1

print(b) # 1
print(NVL(b,0)) # 1

# Also,
print(NVL(NULL,1,0)) # 1
print(NVL(NULL,0,1)) # 0
print(NVL(NULL,NULL,0)) # 0
print(NVL(NULL,NULL,NULL)) # NULL
```

`opttest` *Optionally test code depending on environment variable.*

Description

A convenience wrapper to run code based on whether an environment variable is defined.

Usage

```
opttest(expr, testname = NULL, testvar = "ENABLE_statnet_TESTS",
        yesvals=c("y","yes","t","true","1"), lowercase=TRUE)
```

Arguments

`expr`	An expression to be evaluated only if `testvar` is set to a non-empty value.
`testname`	Optional name of the test. If given, and the test is skipped, will print a message to that end, including the name of the test, and instructions on how to enable it.
`testvar`	Environment variable name. If set to one of the `yesvals`, `expr` is run. Otherwise, an optional message is printed.
`yesvals`	A character vector of strings considered affirmative values for `testvar`.
`lowercase`	Whether to convert the value of `testvar` to lower case before comparing it to `yesvals`.

paste.and	*Concatenates the elements of a vector (optionaly enclosing them in quotation marks or parentheses) adding appropriate punctuation and unions.*

Description

A vector x becomes "x[1]", "x[1] and x[2]", or "x[1], x[2], and x[3]", depending on the langth of x.

Usage

```
paste.and(x, oq = "", cq = "")
```

Arguments

x	A vector.
oq	Opening quotation symbol. (Defaults to none.)
cq	Closing quotation symbol. (Defaults to none.)

Value

A string with the output.

See Also

paste, cat

Examples

```
print(paste.and(c()))

print(paste.and(1))

print(paste.and(1:2))

print(paste.and(1:3))

print(paste.and(1:4))
```

print.control.list *Pretty print the control list*

Description

This function prints the control list, including what it can control and the elements.

Usage

```
## S3 method for class 'control.list'
print(x, ...)
```

Arguments

x	A list generated by a control.* function.
...	Unused at this time.

See Also

check.control.class, set.control.class

set.control.class *Set the class of the control list*

Description

This function sets the class of the control list, with the default being the name of the calling function.

Usage

```
set.control.class(myname = {
    sc <- sys.calls()
    as.character(sc[[length(sc) - 1]][[1]])
}, control = get("control", pos = parent.frame()))
```

Arguments

myname	Name of the class to set. Defaults to the name of the calling function.
control	Control list. Defaults to the control variable in the calling function.

Value

The control list with class set.

See Also

check.control.class, print.control.list

`sort.data.frame` *Implements a* `sort` *method for* `data.frame`*, sorting it in lexicographic order.*

Description

This function returns a data frame sorted in lexcographic order: first by the first column, ties broken by the second, remaining ties by the third, etc..

Usage

```
## S3 method for class 'data.frame'
sort(x, decreasing = FALSE, ...)
```

Arguments

`x`	A `data.frame` to sort.
`decreasing`	Whether to sort in decreasing order.
`...`	Ignored. Needed for compatibility with the generic.

Value

A data frame, sorted lexicographically.

See Also

data.frame, sort

Examples

```
data(iris)

head(iris)

head(sort(iris))
```

`statnet.cite` `CITATION` *file utilities for Statnet packages*

Description

These functions automate citation generation for Statnet Project packages.

Usage

```
statnet.cite.pkg(pkg)

statnet.cite.head(pkg)

statnet.cite.foot(pkg)
```

Arguments

pkg	Name of the package whose citation is being generated.

Value

For `statnet.cite.head` and `statnet.cite.foot`, an object of type `citationHeader` and `citationFooter`, respectively, understood by the `citation` function, with package name substituted into the template.

For `statnet.cite.pkg`, an object of class `bibentry` containing a 'software manual' citation for the package constructed from the current version and author information in the `DESCRIPTION` and a template.

See Also

citation, citHeader, citFooter, bibentry

Examples

```
statnet.cite.head("statnet.common")

statnet.cite.pkg("statnet.common")

statnet.cite.foot("statnet.common")
```

---

statnetStartupMessage *Construct a "standard" startup message to be printed when the package is loaded.*

Description

This function uses information returned by `packageDescription` to construct a standard package startup message according to the policy of the Statnet Project. To determine institutional affiliation, it uses a lookup table that maps domain names to institutions. (E.g., *.uw.edu or *.washington.edu maps to University of Washington.)

Usage

```
statnetStartupMessage(pkgname, friends, nofriends)
```

Arguments

pkgname	Name of the package whose information is used.
friends	This argument is required, but will only be interpreted if the Statnet Project policy makes use of "friendly" package information. A character vector of names of packages whose attribution information incorporates the attribution information of this package, or TRUE. (This may, in the future, lead the package to suppress its own startup message when loaded by a "friendly" package.) If TRUE, the package considers all other packages "friendly". (This may, in the future, lead the package to suppress its own startup message when loaded by another package, but print it when loaded directly by the user.)
nofriends	This argument controls the startup message if the Statnet Project policy does not make use of "friendly" package information but does make use of whether or not the package is being loaded directly or as a dependency. If TRUE, the package is willing to suppress its startup message if loaded as a dependency. If FALSE, it is not.

Value

A string containing the startup message, to be passed to the packageStartupMessage call or NULL, if policy prescribes printing R's default startup message. (Thus, if statnetStartupMessage returns NULL, the calling package should not call packageStartupMessage at all.)

Note that arguments to friends and nofriends are merely requests, to be interpreted (or ignored) by the statnetStartupMessage according to the Statnet Project policy.

See Also

packageDescription

Examples

```
## Not run:
.onAttach <- function(lib, pkg){
  sm <- statnetStartupMessage("ergm", friends=c("statnet","ergm.count","tergm"), nofriends=FALSE)
  if(!is.null(sm)) packageStartupMessage(sm)
}

## End(Not run)
```

---

vector.namesmatch *reorder vector v into order determined by matching the names of its elements to a vector of names*

Description

A helper function to reorder vector v (if named) into order specified by matching its names to the argument names

Usage

```
vector.namesmatch(v, names, errname = NULL)
```

Arguments

v	a vector (or list) with named elements, to be reorderd
names	a character vector of element names, corresponding to names of v, specificying desired orering of v
errname	optional, name to be reported in any error messages. default to deparse(substitute(v))

Details

does some checking of appropriateness of arguments, and reorders v by matching its names to character vector names

Value

returns v, with elements reordered

Note

earlier versions of this function did not order as advertiased

Examples

```
test<-list(c=1,b=2,a=3)
vector.namesmatch(test,names=c('a','c','b'))
```

$.control.list *Named element accessor for ergm control lists*

Description

Utility method that overrides the standard '$' list accessor to disable partial matching for ergm control.list objects

Usage

```
## S3 method for class 'control.list'
object$name
```

Arguments

object	list-coearceable object with elements to be searched
name	literal character name of list element to search for and return

Details

Executes `getElement` instead of `$` so that element names must match exactly to be returned and partially matching names will not return the wrong object.

Value

Returns the named list element exactly matching `name`, or `NULL` if no matching elements found

Author(s)

pavel

See Also

see `getElement`

Examples

```
## The function is currently defined as
function (object, name)
{
    if (isS4(object))
        slot(object, name)
    else object[[name, exact = TRUE]]
  }
```

Index

∗Topic **debugging**
  opttest, 6
∗Topic **environment**
  opttest, 6
∗Topic **manip**
  compress.data.frame, 2
  sort.data.frame, 9
∗Topic **utilities**
  check.control.class, 2
  ERRVL, 3
  NVL, 5
  opttest, 6
  paste.and, 7
  print.control.list, 8
  set.control.class, 8
  statnet.cite, 9
  statnetStartupMessage, 10
$, *13*
$.control.list, 12

append.rhs.formula *(*formula.utils*)*, 4

bibentry, *10*

check.control.class, 2
citation, *10*
compress.data.frame, 2

data.frame, *9*

ERRVL, 3

formula.utils, 4

getElement, *13*

if, *5*
inherits, *3*
is.null, *5*

list, *3*

nonsimp.update.formula *(*formula.utils*)*, 4
NVL, *3*, 5

opttest, 6

packageDescription, *10*
packageStartupMessage, *11*
paste.and, 7
print.control.list, 8

set.control.class, 8
sort, *9*
sort.data.frame, 9
statnet.cite, 9
statnetStartupMessage, 10

term.list.formula *(*formula.utils*)*, 4
try, *3*

update.formula, *4*

vector.namesmatch, 11