Package 'statnet.common'

August 15, 2017

Version 4.0.0

Date 2017-08-15

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, methods

BugReports https://github.com/statnet/statnet.common/issues

License GPL-3 + file LICENSE

URL http://www.statnet.org

Roxygen list(markdown = TRUE)

RoxygenNote 6.0.1

R topics documented:

`all_identical` *Test if all items in a vector or a list are identical.*

Description

Test if all items in a vector or a list are identical.

Usage

```
all_identical(x)
```

Arguments

x	a vector or a list

Value

`TRUE` if all elements of `x` are identical to each other.

See Also

`base::identical()`

Examples

```
stopifnot(!all_identical(1:3))

stopifnot(all_identical(list("a", "a", "a")))
```

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

Description

`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).

Usage

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

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

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

Arguments

object	formula object to be updated
newterms	list of terms (names) to append to the formula, or a formula whose RHS terms will be used
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
new	new formula to be used in updating
...	Additional arguments. Currently unused.
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.

Value

append.rhs.formula each return an updated formula object

nonsimp.update.formula each return an updated formula object

terms.list.formula returns a list of formula terms, with an additional numerical vector attribute "sign" with of the same length, giving the corresponding term's sign as +1 or -1.

Functions

- nonsimp.update.formula: 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: 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.

Examples

```
## append.rhs.formula

(f1 <- append.rhs.formula(y~x,list(as.name("z1"),as.name("z2"))))
(f2 <- append.rhs.formula(~y,list(as.name("z"))))
(f3 <- append.rhs.formula(~y+x,list(as.name("z"))))
(f4 <- append.rhs.formula(~y,list(as.name("z")),TRUE))
(f5 <- append.rhs.formula(y~x,~z1+z2))
```

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.

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.

Note

In earlier versions, `OKnames` and `myname` were autodetected. This capability has been deprecated and results in a warning issued once per session. They now need to be set explicitly.

See Also

set.control.class, print.control.list

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

Description

`compress.data.frame` "compresses" a data frame, returning unique rows and a tally of the number of times each row is repeated, as well as a permutation vector that can reconstruct the original data frame. `decompress.data.frame` reconstructs the original data frame.

Usage

```
compress.data.frame(x)

decompress.data.frame(x)
```

Arguments

x	For compress.data.frame a data.frame to be compressed. For decompress.data.frame a list as returned by compress.data.frame.

Value

For compress.data.frame, a list with three 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
ordering	A vector such that if c is the compressed data frame, c$rows[c$ordering,,drop=FALSE] equals the original data frame, except for row names
rownames	Row names of x

For decompress.data.frame, the original data frame.

See Also

data.frame

Examples

```
(x <- data.frame(V1=sample.int(3,30,replace=TRUE),
                 V2=sample.int(2,30,replace=TRUE),
                 V3=sample.int(4,30,replace=TRUE)))

(c <- compress.data.frame(x))

stopifnot(all(decompress.data.frame(c)==x))
```

---

compress_rows *A generic function to compress a row-weighted table*

Description

Compress a matrix or a data frame with duplicated rows, updating row weights to reflect frequencies, or reverse the process, reconstructing a matrix like the one compressed (subject to permutation of rows and weights not adding up to an integer).

Usage

```
compress_rows(x, ...)

decompress_rows(x, target.nrows = NULL, ...)
```

Arguments

x	a weighted matrix or data frame.
...	extra arguments for methods.
target.nrows	the approximate number of rows the uncompressed matrix should have; if not achievable exactly while respecting proportionality, a matrix with a slightly different number of rows will be constructed.

Value

For compress_rows A weighted matrix or data frame of the same type with duplicated rows removed and weights updated appropriately.

---

control.list.accessor *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 N. Krivitsky

See Also

see getElement

control.remap *Overwrite control parameters of one configuration with another.*

Description

Given a `control.list`, and two prefixes, `from` and `to`, overwrite the elements starting with `to` with the corresponding elements starting with `from`.

Usage

```
control.remap(control, from, to)
```

Arguments

`control`	An object of class `control.list`.
`from`	Prefix of the source of control parameters.
`to`	Prefix of the destination of control parameters.

Value

An `control.list` object.

Author(s)

Pavel N. Krivitsky

See Also

`print.control.list`

Examples

```
(l <- set.control.class("test", list(a.x=1, a.y=2)))
control.remap(l, "a", "b")
```

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
```

---

logspace.utils *Utilities for performing calculations on logarithmic scale.*

Description

A small suite of functions to compute sums, means, and weighted means on logarithmic scale, minimizing loss of precision.

Usage

```
log_sum_exp(logx, use_ldouble = FALSE)

log_mean_exp(logx, use_ldouble = FALSE)

lweighted.mean(x, logw)

lweighted.var(x, logw)
```

Arguments

logx	Numeric vector of $\log(x)$, the natural logarithms of the values to be summed or averaged.
use_ldouble	Whether to use long double precision in the calculation. If TRUE, 's C built-in logspace_sum() is used. If FALSE, the package's own implementation based on it is used, using double precision, which is (on most systems) several times faster, at the cost of precision.
x	Numeric vector of x, the (raw) values to be summed or averaged. For lweighted.mean, x may also be a matrix, in which case the weighted mean will be computed for each column of x.
logw	Numeric vector of $\log(w)$, the natural logarithms of the weights.

Value

The functions return the equivalents of the following R expressions, but faster and with less loss of precision:

`log_sum_exp(logx)` `log(sum(exp(logx)))`

`log_mean_exp(logx)` `log(mean(exp(logx)))`

`lweighted.mean(x,logw)` `sum(x*exp(logw))/sum(exp(logw))` for `x` scalar and `colSums(x*exp(logw))/sum(exp(` for `x` matrix

`lweighted.var(x,logw)` `crossprod(x*exp(logw/2))/sum(exp(logw))`

Author(s)

Pavel N. Krivitsky

Examples

```
logx <- rnorm(1000)
stopifnot(all.equal(log(sum(exp(logx))), log_sum_exp(logx)))
stopifnot(all.equal(log(mean(exp(logx))), log_mean_exp(logx)))

x <- rnorm(1000)
logw <- rnorm(1000)
stopifnot(all.equal(m <- sum(x*exp(logw))/sum(exp(logw)),lweighted.mean(x, logw)))
stopifnot(all.equal(sum((x-m)^2*exp(logw))/sum(exp(logw)),
                    lweighted.var(x, logw), check.attributes=FALSE))

x <- cbind(x, rnorm(1000))
stopifnot(all.equal(m <- colSums(x*exp(logw))/sum(exp(logw)),
                    lweighted.mean(x, logw), check.attributes=FALSE))
stopifnot(all.equal(crossprod(t(t(x)-m)*exp(logw/2))/sum(exp(logw)),
                    lweighted.var(x, logw), check.attributes=FALSE))
```

NVL *Convenience functions for handling NULL objects.*

Description

Convenience functions for handling NULL objects.

Usage

```
NVL(...)

NVL(x) <- value
```

Arguments

`...`	Expressions to be tested.
`x`	an object to be overwritten if NULL.
`value`	new value for `x`.

Functions

- NVL: Inspired by SQL function NVL, NVL(...) returns the first argument that is not NULL, or NULL if all arguments are NULL.
- NVL<-: Assigning to NVL overwrites its first argument if that argument is NULL. Note that it will *always* return the right-hand-side of the assignment (value), regardless of what x is.

See Also

NULL, 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

NVL(a) <- 2
a # 2
NVL(b) <- 2
b # still 1
```

---

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.

---

order *Implement the* sort *and* order *methods for* data.frame *and* matrix*, sorting it in lexicographic order.*

Description

These function return a data frame sorted in lexcographic order or a permutation that will rearrange it into lexicographic order: first by the first column, ties broken by the second, remaining ties by the third, etc..

Usage

```
order(..., na.last = TRUE, decreasing = FALSE)

## Default S3 method:
order(..., na.last = TRUE, decreasing = FALSE)

## S3 method for class 'data.frame'
order(..., na.last = TRUE, decreasing = FALSE)

## S3 method for class 'matrix'
order(..., na.last = TRUE, decreasing = FALSE)

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

Arguments

...	Ignored for sort. For order, first argument is the data frame to be ordered. (This is needed for compatibility with order.)
na.last	See order documentation.
decreasing	Whether to sort in decreasing order.
x	A data.frame to sort.

Value

For sort, a data frame, sorted lexicographically. For order, a permutation I (of a vector 1:nrow(x)) such that x[I,,drop=FALSE] equals x ordered lexicographically.

See Also

data.frame, sort, order, matrix

Examples

```
data(iris)

head(iris)

head(order(iris))

head(sort(iris))

stopifnot(identical(sort(iris),iris[order(iris),]))
```

`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

rle.utils *RLE utilities*

Description

Simple utilities for operations on RLE-encoded vectors.

Usage

```
## S3 method for class 'rle'
c(...)

## S3 method for class 'rle'
!x

binop.rle(e1, e2, FUN)

## S3 method for class 'rle'
e1 | e2

## S3 method for class 'rle'
e1 & e2

compact.rle(x)

## S3 method for class 'rle'
any(..., na.rm = FALSE)
```

```
## S3 method for class 'rle'
all(..., na.rm = FALSE)

## S3 method for class 'rle'
e1 * e2

## S3 method for class 'rle'
e1 / e2

## S3 method for class 'rle'
e1 - e2

## S3 method for class 'rle'
e1 + e2

## S3 method for class 'rle'
e1 ^ e2

## S3 method for class 'rle'
e1 %% e2

## S3 method for class 'rle'
e1 %/% e2

## S3 method for class 'rle'
e1 == e2

## S3 method for class 'rle'
e1 > e2

## S3 method for class 'rle'
e1 < e2

## S3 method for class 'rle'
e1 != e2

## S3 method for class 'rle'
e1 <= e2

## S3 method for class 'rle'
e1 >= e2

## S3 method for class 'rle'
rep(x, ..., scale = c("element", "run"))
```

Arguments

`...`	For `c`, objects to be concatenated. The first object must be of class `rle()`. For `rep`, see documentation for `rep()`.
`x, e1, e2`	Arguments to unary (`x`) and binary (`e1` and `e2`) operators.
`FUN`	A binary function or operator or a name of one. It is assumed to be vectorized: it expects two vectors of equal lengths and outputs a vector of the same length.

na.rm	see documentation for any() and all().
scale	whether to replicate the elements of the RLE-compressed vector or the runs.

Value

All functions return an rle() object. By default, the functions and the operators do not merge adjacent runs with the same value. This must be done explicitly with compact.rle().

Functions

- binop.rle: Perform an arbitrary binary operation on the pair of vectors represented by the rle() objects.
- compact.rle: Compact the rle() object by merging adjacent runs.

Note

Since rle() stores run lengths as integers, compact.rle() will not merge runs that add up to lengths greater than what can be represented by a 32-bit signed integer (2147483647).

The rep() method for rle() objects is very limited at this time: . Even though the default setting is to replicate elements of the vector, only the run-replicating functionality is implemented at this time.

Examples

```
x <- rle(as.logical(rbinom(10,1,.7)))
y <- rle(as.logical(rbinom(10,1,.3)))

stopifnot(c(inverse.rle(x),inverse.rle(y))==inverse.rle(c(x,y)))

stopifnot((!inverse.rle(x))==inverse.rle(!x))
stopifnot((inverse.rle(x)|inverse.rle(y))==inverse.rle(x|y))
stopifnot((inverse.rle(x)&inverse.rle(y))==inverse.rle(x&y))
stopifnot(identical(rle(inverse.rle(x)&inverse.rle(y)),compact.rle(x&y)))

big <- structure(list(lengths=as.integer(rep(.Machine$integer.max/4,6)),
                      values=rep(TRUE,6)), class="rle")

stopifnot(all(aggregate(as.numeric(lengths)~values,
                        data=as.data.frame(unclass(big)),FUN=sum)
              ==
              aggregate(as.numeric(lengths)~values,
                        data=as.data.frame(unclass(compact.rle(big))),
                        FUN=sum)))

x <- rle(as.logical(rbinom(10,1,.9)))
y <- rle(as.logical(rbinom(10,1,.1)))

stopifnot(any(x)==any(inverse.rle(x)))
stopifnot(any(y)==any(inverse.rle(y)))


stopifnot(all(x)==all(inverse.rle(x)))
stopifnot(all(y)==all(inverse.rle(y)))
```

```
x <- rle(sample(c(-1,+1), 10, c(.7,.3), replace=TRUE))
y <- rle(sample(c(-1,+1), 10, c(.3,.7), replace=TRUE))

stopifnot((inverse.rle(x)*inverse.rle(y))==inverse.rle(x*y))
stopifnot((inverse.rle(x)/inverse.rle(y))==inverse.rle(x/y))
stopifnot((-inverse.rle(y))==inverse.rle(-y))
stopifnot((inverse.rle(x)-inverse.rle(y))==inverse.rle(x-y))
stopifnot((+inverse.rle(y))==inverse.rle(+y))
stopifnot((inverse.rle(x)+inverse.rle(y))==inverse.rle(x+y))
stopifnot((inverse.rle(x)^inverse.rle(y))==inverse.rle(x^y))
stopifnot((inverse.rle(x)%%inverse.rle(y))==inverse.rle(x%%y))
stopifnot((inverse.rle(x)%/%inverse.rle(y))==inverse.rle(x%/%y))
stopifnot((inverse.rle(x)==inverse.rle(y))==inverse.rle(x==y))
stopifnot((inverse.rle(x)>inverse.rle(y))==inverse.rle(x>y))
stopifnot((inverse.rle(x)<inverse.rle(y))==inverse.rle(x<y))
stopifnot((inverse.rle(x)!=inverse.rle(y))==inverse.rle(x!=y))
stopifnot((inverse.rle(x)<=inverse.rle(y))==inverse.rle(x<=y))
stopifnot((inverse.rle(x)>=inverse.rle(y))==inverse.rle(x>=y))

x <- rle(sample(c(-1,+1), 10, c(.7,.3), replace=TRUE))
y <- rpois(length(x$lengths), 2)


stopifnot(all(rep(inverse.rle(x), rep(y, x$lengths))==inverse.rle(rep(x, y, scale="run"))))
```

`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.
`control`	Control list. Defaults to the `control` variable in the calling function.

Value

The control list with class set.

Note

In earlier versions, `OKnames` and `myname` were autodetected. This capability has been deprecated and results in a warning issued once per session. They now need to be set explicitly.

See Also

check.control.class, print.control.list

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

Description

These functions automate citation generation for Statnet Project packages.

Usage

```
statnet.cite.head(pkg)

statnet.cite.foot(pkg)

statnet.cite.pkg(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 '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)
```

`sweep_cols.matrix` *Suptract a elements of a vector from respective columns of a matrix*

Description

An optimized function equivalent to `sweep(x, 2, STATS)` for a matrix `x`.

Usage

```
sweep_cols.matrix(x, STATS, disable_checks = FALSE)
```

Arguments

`x`	a numeric matrix;
`STATS`	a numeric vector whose length equals to the number of columns of `x`.
`disable_checks`	if `TRUE`, do not check that `x` is a numeric matrix and its number of columns matches the length of `STATS`; set in production code for a significant speed-up.

Value

A matrix of the same attributes as `x`.

Examples

```
x <- matrix(runif(1000), ncol=4)
s <- 1:4

stopifnot(all.equal(sweep_cols.matrix(x, s), sweep(x, 2, s)))
```

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'))
```

wmatrix *A data matrix with row weights*

Description

A representation of a numeric matrix with row weights, represented on either linear (linwmatrix) or logarithmic (logwmatrix) scale.

Usage

```
logwmatrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE,
  dimnames = NULL, w = NULL)

linwmatrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE,
  dimnames = NULL, w = NULL)

is.wmatrix(x)

is.logwmatrix(x)

is.linwmatrix(x)

as.linwmatrix(x, ...)

as.logwmatrix(x, ...)

## S3 method for class 'linwmatrix'
as.linwmatrix(x, ...)

## S3 method for class 'logwmatrix'
as.linwmatrix(x, ...)

## S3 method for class 'logwmatrix'
as.logwmatrix(x, ...)

## S3 method for class 'linwmatrix'
as.logwmatrix(x, ...)

## S3 method for class 'matrix'
as.linwmatrix(x, w = NULL, ...)

## S3 method for class 'matrix'
as.logwmatrix(x, w = NULL, ...)

## S3 method for class 'wmatrix'
print(x, ...)

## S3 method for class 'logwmatrix'
print(x, ...)

## S3 method for class 'linwmatrix'
print(x, ...)

## S3 method for class 'logwmatrix'
compress_rows(x, ...)

## S3 method for class 'linwmatrix'
compress_rows(x, ...)

## S3 method for class 'wmatrix'
decompress_rows(x, target.nrows = NULL, ...)
```

```
## S3 method for class 'wmatrix'
x[i, j, ..., drop = FALSE]

## S3 replacement method for class 'wmatrix'
x[i, j, ...] <- value
```

Arguments

`data, nrow, ncol, byrow, dimnames`	passed to `matrix()`.
`w`	row weights on the appropriate scale.
`x`	an object to be coerced or tested.
`...`	extra arguments, currently unused.
`target.nrows`	see `decompress_rows()`.
`i, j, value`	rows and columns and values for extraction or replacement; as `matrix()`.
`drop`	Used for consistency with the generic. Ignored, and always treated as `FALSE`.

Value

An object of class `linwmatrix`/`logwmatrix` and `wmatrix`, which is a `matrix()` but also has an attribute `w` containing row weights on the linear or the natural-log-transformed scale.

Note

Note that `wmatrix` itself is an "abstract" class: you cannot instantiate it.

Note that at this time, `wmatrix` is designed as, first and foremost, as class for storing compressed data matrices, so most methods that operate on matrices may not handle the weights correctly and may even cause them to be lost.

See Also

`rowweights()`, `lrowweights()`, `compress_rows()`

Examples

```
(m <- matrix(1:3, 2, 3, byrow=TRUE))
(m <- rbind(m, 3*m, 2*m, m))
(mlog <- as.logwmatrix(m))
(mlin <- as.linwmatrix(m))
(cmlog <- compress_rows(mlog))
(cmlin <- compress_rows(mlin))

stopifnot(all.equal(as.linwmatrix(cmlog),cmlin))

cmlog[2,] <- 1:3
(cmlog <- compress_rows(cmlog))
stopifnot(sum(rowweights(cmlog))==nrow(m))

(m3 <- matrix(c(1:3,(1:3)*2,(1:3)*3), 3, 3, byrow=TRUE))
(rowweights(m3) <- c(4, 2, 2))

stopifnot(all.equal(compress_rows(as.logwmatrix(m)), as.logwmatrix(m3),check.attributes=FALSE))
```

```
stopifnot(all.equal(rowweights(compress_rows(as.logwmatrix(m))),
                    rowweights(as.logwmatrix(m3)),check.attributes=FALSE))
```

---

wmatrix_weights *Set or extract weighted matrix row weights*

Description

Set or extract weighted matrix row weights

Usage

```
rowweights(x, ...)

## S3 method for class 'linwmatrix'
rowweights(x, ...)

## S3 method for class 'logwmatrix'
rowweights(x, ...)

lrowweights(x, ...)

## S3 method for class 'logwmatrix'
lrowweights(x, ...)

## S3 method for class 'linwmatrix'
lrowweights(x, ...)

rowweights(x, ...) <- value

## S3 replacement method for class 'linwmatrix'
rowweights(x, update = TRUE, ...) <- value

## S3 replacement method for class 'logwmatrix'
rowweights(x, update = TRUE, ...) <- value

lrowweights(x, ...) <- value

## S3 replacement method for class 'linwmatrix'
lrowweights(x, update = TRUE, ...) <- value

## S3 replacement method for class 'logwmatrix'
lrowweights(x, update = TRUE, ...) <- value

## S3 replacement method for class 'matrix'
rowweights(x, ...) <- value

## S3 replacement method for class 'matrix'
lrowweights(x, ...) <- value
```

Arguments

x	a linwmatrix(), a logwmatrix(), or a matrix(); a matrix() is coerced to a weighted matrix of an appropriate type.
...	extra arguments for methods.
value	weights to set, on the appropriate scale.
update	if TRUE (the default), the old weights are updated with the new weights (i.e., corresponding weights are multiplied on linear scale or added on on log scale); otherwise, they are overwritten.

Value

For the accessor functions, the row weights or the row log-weights; otherwise, a weighted matrix with modified weights. The type of weight (linear or logarithmic) is converted to the required type and the type of weighting of the matrix is preserved.

Index