Package: grr (via r-universe)

November 13, 2026

Title Alternate Implementations of Base R Functions

Version 0.9.9002

Author Craig Varrichio [aut, cre]

Maintainer Craig Varrichio <canthony427@gmail.com>

Description Alternate implementations of some base R functions, including sort, order, and match. Functions are faster and/or have been otherwise augmented. Compatibility updates by Renova.

Depends R (>= 3.5.0)

License GPL-3

LazyData true

RoxygenNote 5.0.1

NeedsCompilation yes

Repository https://renova-r.r-universe.dev

Date/Publication 2026-11-13 19:48:29 UTC

RemoteUrl https://github.com/renova-r/grr

RemoteRef renova-0.9.9002

RemoteSha 4f657eeb3c0580b3e20d89e3789ecd7123df40a9

Contents

extract 2
grr 3
matches 3
order2 5
sort2 6

Index **8**

`extract` *Extract/return parts of objects*

Description

Alternative to built-in `Extract` or `[`. Allows for extraction operations that are ambivalent to the data type of the object. For example, `extract(x,i)` will work on lists, vectors, data frames, matrices, etc.

Usage

```
extract(x, i = NULL, j = NULL)
```

Arguments

`x`	object from which to extract elements
`i, j`	indices specifying elements to extract. Can be `numeric`, `character`, or `logical` vectors.

Details

Extraction is 2-100x faster on data frames than with the built in operation - but does not preserve row names.

Examples

```
#Typically about twice as fast on normal subselections
orders<-data.frame(orderNum=1:1e5,
 sku=sample(1e3, 1e5, TRUE),
 customer=sample(1e4,1e5,TRUE))
a<-sample(1e5,1e4)
system.time(b<-orders[a,])
system.time(c<-extract(orders,a))
rownames(b)<-NULL
rownames(c)<-NULL
identical(b,c)

#Speedup increases to 50-100x with oversampling
a<-sample(1e5,1e6,TRUE)
system.time(b<-orders[a,])
system.time(c<-extract(orders,a))
rownames(b)<-NULL
rownames(c)<-NULL
identical(b,c)

#Can create function calls that work for multiple data types
alist<-as.list(1:50)
avector<-1:50
extract(alist,1:5)
```

```
extract(avector,1:5)
extract(orders,1:5)#'

## Not run:
orders<-data.frame(orderNum=as.character(sample(1e5, 1e6, TRUE)),
 sku=sample(1e3, 1e6, TRUE),
 customer=sample(1e4,1e6,TRUE))
system.time(a<-sample(1e6,1e7,TRUE))
system.time(b<-orders[a,])
system.time(c<-extract(orders,a))

## End(Not run)
```

---

grr *Alternate Implementations of Base R Functions*

Description

Alternate implementations of some base R functions, including sort, order, and match. Functions are faster and/or have been otherwise augmented. See the documentation of individual functions for details and benchmarks.

---

matches *Value Matching*

Description

Returns a lookup table or list of the positions of ALL matches of its first argument in its second and vice versa. Similar to match, though that function only returns the first match.

Usage

```
matches(x, y, all.x = TRUE, all.y = TRUE, list = FALSE, indexes = TRUE,
  nomatch = NA)
```

Arguments

x	vector. The values to be matched. Long vectors are not currently supported.
y	vector. The values to be matched. Long vectors are not currently supported.
all.x	logical; if TRUE, then each value in x will be included even if it has no matching values in y
all.y	logical; if TRUE, then each value in y will be included even if it has no matching values in x
list	logical. If TRUE, the result will be returned as a list of vectors, each vector being the matching values in y. If FALSE, result is returned as a data frame with repeated values for each match.

indexes	logical. Whether to return the indices of the matches or the actual values.
nomatch	the value to be returned in the case when no match is found. If not provided and indexes=TRUE, items with no match will be represented as NA. If set to NULL, items with no match will be set to an index value of length+1. If indexes=FALSE, they will default to NA.

Details

This behavior can be imitated by using joins to create lookup tables, but matches is simpler and faster: usually faster than the best joins in other packages and thousands of times faster than the built in merge.

all.x/all.y correspond to the four types of database joins in the following way:

left all.x=TRUE, all.y=FALSE

right all.x=FALSE, all.y=TRUE

inner all.x=FALSE, all.y=FALSE

full all.x=TRUE, all.y=TRUE

Examples

```
one<-as.integer(1:10000)
two<-as.integer(sample(1:10000,1e3,TRUE))
system.time(a<-lapply(one, function (x) which(two %in% x)))
system.time(b<-matches(one,two,all.y=FALSE,list=TRUE))

one<-round(runif(1e3),3)
two<-round(runif(1e3),3)
system.time(a<-lapply(one, function (x) which(two %in% x)))
system.time(b<-matches(one,two,all.y=FALSE,list=TRUE))

one<-as.character(1:1e5)
two<-as.character(sample(1:1e5,1e5,TRUE))
system.time(b<-matches(one,two,list=FALSE))
system.time(c<-merge(data.frame(key=one),data.frame(key=two),all=TRUE))

## Not run:
one<-as.integer(1:1000000)
two<-as.integer(sample(1:1000000,1e5,TRUE))
system.time(b<-matches(one,two,indexes=FALSE))
if(requireNamespace("dplyr",quietly=TRUE))
 system.time(c<-dplyr::full_join(data.frame(key=one),data.frame(key=two)))
if(require(data.table,quietly=TRUE))
 system.time(d<-merge(data.table(data.frame(key=one))
             ,data.table(data.frame(key=two))
             ,by='key',all=TRUE,allow.cartesian=TRUE))

one<-as.character(1:1000000)
two<-as.character(sample(1:1000000,1e5,TRUE))
system.time(a<-merge(one,two)) #Times out
system.time(b<-matches(one,two,indexes=FALSE))
```

```
if(requireNamespace("dplyr",quietly=TRUE))
 system.time(c<-dplyr::full_join(data.frame(key=one),data.frame(key=two)))#'
if(require(data.table,quietly=TRUE))
{
 system.time(d<-merge(data.table(data.frame(key=one))
             ,data.table(data.frame(key=two))
             ,by='key',all=TRUE,allow.cartesian=TRUE))
 identical(b[,1],as.character(d$key))
}

## End(Not run)
```

---

order2 *Ordering vectors*

Description

Alternative to order. For large vectors, typically is about 3x faster for numbers and 20x faster for characters.

Usage

```
order2(x)
```

Arguments

x	a vector of class numeric, integer, character, factor, or logical.

Examples

```
chars<-as.character(sample(1e3,1e4,TRUE))
system.time(a<-order(chars))
system.time(b<-order2(chars))
identical(chars[a],chars[b])

ints<-as.integer(sample(1e3,1e4,TRUE))
system.time(a<-order(ints))
system.time(b<-order2(ints))
identical(ints[a],ints[b])

nums<-runif(1e4)
system.time(a<-order(nums))
system.time(b<-order2(nums))
identical(nums[a],nums[b])

logs<-as.logical(sample(0:1,1e6,TRUE))
system.time(a<-order(logs))
system.time(b<-order2(logs))
identical(logs[a],logs[b])
```

```
facts<-as.factor(as.character(sample(1e3,1e4,TRUE)))
system.time(a<-order(facts))
system.time(b<-order2(facts))
identical(facts[a],facts[b])

## Not run:
chars<-as.character(sample(1e5,1e6,TRUE))
system.time(a<-order(chars))
system.time(b<-order2(chars))

ints<-as.integer(sample(1e5,1e6,TRUE))
system.time(result<-order(ints))
system.time(result<-order2(ints))

nums<-runif(1e6)
system.time(result<-order(nums))
system.time(result<-order2(nums))

logs<-as.logical(sample(0:1,1e7,TRUE))
system.time(result<-order(logs))
system.time(result<-order2(logs))

facts<-as.factor(as.character(sample(1e5,1e6,TRUE)))
system.time(a<-order(facts))
system.time(b<-order2(facts))
identical(facts[a],facts[b])

## End(Not run)
```

`sort2` *Sorting vectors*

Description

Simplified implementation of sort that is much faster than base::sort. For large vectors, typically is about 2x faster for numbers and 20x faster for characters and factors.

Usage

```
sort2(x)
```

Arguments

x	a vector of class numeric, integer, character, factor, or logical.

Examples

```
chars<-as.character(sample(1e3,1e4,TRUE))
system.time(a<-sort(chars))
system.time(b<-sort2(chars))
```

```
identical(a,b)

ints<-as.integer(sample(1e3,1e4,TRUE))
system.time(a<-sort(ints))
system.time(b<-sort2(ints))
identical(a,b)

nums<-runif(1e4)
system.time(a<-sort(nums))
system.time(b<-sort2(nums))
identical(a,b)

logs<-as.logical(sample(0:1,1e6,TRUE))
system.time(result<-sort(logs))
system.time(result<-sort2(logs))

facts<-as.factor(as.character(sample(1e3,1e4,TRUE)))
system.time(a<-sort(facts))
system.time(b<-sort2(facts))
identical(a,b)

## Not run:
chars<-as.character(sample(1e5,1e6,TRUE))
system.time(a<-sort(chars))
system.time(b<-sort2(chars))

ints<-as.integer(sample(1e5,1e6,TRUE))
system.time(result<-sort(ints))
system.time(result<-sort2(ints))

nums<-runif(1e6)
system.time(result<-sort(nums))
system.time(result<-sort2(nums))

logs<-as.logical(sample(0:1,1e7,TRUE))
system.time(result<-sort(logs))
system.time(result<-sort2(logs))

facts<-as.factor(as.character(sample(1e5,1e6,TRUE)))
system.time(a<-sort(facts))
system.time(b<-sort2(facts))

## End(Not run)
```

Index

Extract, *2*
extract, 2

grr, 3
grr-package *(grr)*, 3

match, *3*
matches, 3
merge, *4*

order, *5*
order2, 5

sort2, 6