Package 'Stack'

February 19, 2015

Type Package

Title Stylized concatenation of data.frames or ffdfs

Version 2.0-1

Date 2014-03-20

Author Mike Malecki <Mike.Malecki at yougov.com>

Maintainer Mike Malecki <Mike.Malecki@yougov.com>

Description Stacks rectangular datasets on top of each other, possibly performing several type coercions along the way. For large datasets, depends on the ff package. Provides an aggressive version of ffbase::compact for data that may appear be real-typed but is in fact int/short/byte. For many purposes plyr::rbind.fill may be more appropriate, but for some kinds of survey data, the rules here work better.

License MIT + file LICENSE

Imports plyr, stringr, methods, ff, ffbase, bit

NeedsCompilation no

Repository CRAN

Date/Publication 2014-03-21 01:19:40

R topics documented:

.preparedf *Prepare a data.frame for as.ffdf to not break*

Description

Presently only converts char to factor. Any other cleaning of things?

Usage

```
.preparedf(df)
```

Arguments

df	a data.frame or a list

Value

df ready for as.ffdf

.upgradevmode *Upgrade vmode of ff column while stacking*

Description

Similar to aux function coerce_to_highest_vmode in ffbase, except this does range/unique values checking rather than simple type checking.

Usage

```
.upgradevmode(x, y, use.na = TRUE)
```

Arguments

x	ff
y	ff or vector
use.na	require a vmode that supports NAs? default=TRUE

compact *Compact a ff vector or ffdf data frame*

Description

Compact takes a ff vector and tries to use the smallest binary data type for this vector.

Usage

```
.compact(x, use.na = TRUE, ...)

## S3 method for class 'ff'
.compact(x, use.na = TRUE, ...)

## S3 method for class 'ffdf'
.compact(x, use.na = TRUE, ...)
```

Arguments

x	ff or ffdf object
use.na	logical if TRUE the resulting ff vector can contain NA, otherwise this is not checked
...	other parameters, not actually passed but needed to muffle check.

Value

compact cloned ff vector, or original if no compacting can be done

ensureDate *Ensure that x is a Date*

Description

Ensure that x is a Date

Usage

```
ensureDate(x, origin = "1970-01-01", dates = NULL)
```

Arguments

x	Date, fac, chr, num, data.frame, or list that might be or contain something that should be a Date
origin	for Dates that may have lost their attributes, and coerced to integer, the origin for reasserting their Date-ness. default=1970-01-01
dates	for data.frame method, columns of x that should themselves be ensureDates

Value

x encoded as a Date *unless all of x is numbers less than 1000,* for datamart numeric survey coding.

`ffStack` *Merge a data.frame into an ffdf data.frame-alike*

Description

For fast operations on large data frames, we want to turn them into ffdf. This is a special case of `Stack` where the first arugment is already an ffdf, and we are Stacking another data.frame into it, expanding factor levels as needed, and possibly enlarging vmodes of existing ff columns.

Usage

```
ffStack(ffdf, df, verbose = FALSE, ...)

## S4 method for signature 'ffdf,data.frame'
ffStack(ffdf, df, verbose = FALSE, ...)
```

Arguments

`ffdf`	an `ffdf`
`df`	a `data.frame`
`verbose`	print extra information about columns as they stack
`...`	further arguments

Value

An ffdf.

`find_min_vmode` *Find minimum vmode for non-ff x*

Description

Find minimum vmode for non-ff x

Usage

```
find_min_vmode(x, use.na = TRUE)
```

Arguments

`x`	any R type
`use.na`	require that the vmode handle NAs

Value

int index of ff vmode that will hold x.

`matrixIntegrate` *Integrate matrices based on their dim names*

Description

Integrate matrices based on their dim names

Usage

```
matrixIntegrate(x, y, x.names = dimnames(x), y.names = dimnames(y),
  joinFunction = pasteExp)
```

Arguments

`x`	matrix
`y`	matrix
`x.names`	a list of length 2
`y.names`	a list of length 2
`joinFunction`	function used to join things

Value

union of x and y, with *write something about expected return ordering*.

`padNA` *Add a length of NAs before or after x (used in Stacking)*

Description

Add a length of NAs before or after x (used in Stacking)

Usage

```
padNA(x, before = 0, after = 0)
```

Arguments

`x`	a data vector
`before`	length of NA to insert
`after`	length of NA to insert

Value

same vector padded with NAs.

Stack *Stack Combine data frames with column names that do not completely intersect*

Description

Stack Combine data frames with column names that do not completely intersect

Usage

```
Stack(df1, df2, return.data.frame = TRUE, dates = c("wave", "date"),
  origin = "1970-01-01", mixed.chr.factor = c("factor", "character"),
  verbose = FALSE)
```

Arguments

df1	a data.frame or a list
df2	a data.frame or a list
return.data.frame	defaults to TRUE and uses quickdf; regular data.frame checks are not done.
dates	character names of columns that are or should be class Date. These columns will be coerced to Date, handling the odd case where they are quoted chr days-since-origin.
origin	Origin for dates that are chr like "15218", which ought to be 2011-09-01. This is not uncommon for dates coming from SPSS.
mixed.chr.factor	when an element is mixed factor/character in df1 and df2, what should the result type be? Default="factor".
verbose	print extra information along the way about each column.

Value

a list of data vectors suitable for coercion to a data.frame. Because as.data.frame is extremely costly and memory-intensive, and Stacking often involves many such column-wise combinations, avoid this by returning a list.

Examples

```
testdf1 <- data.frame(both.int=1:4,
                      expand.factor=c("blue", "yellow"),
                      mixed.fac.int=factor(letters[1:4]),
                      date=as.Date("1983-11-22"),
                      df1only=rep(1:2, each=2),
                      mixed.fac.chr=I(c("a","b","NA",NA)))
testdf2 <- data.frame(both.int=5:24,
                      expand.factor=factor(rep(c(1:4, NA), 4)),
                      mixed.fac.int=1:4,
```

```
                      date=as.Date("1981-09-24"),
                      df2only=factor(c("c", "d")),
                      mixed.fac.chr=c("a","b",NA,"c"))

levels(testdf2$mixed.fac.chr) <- letters #overleveled
## put levels in a different order than sort(levels) would do, but
## don't make it an ordered factor. Result needs to be ordered
## to preserve this.
levels(testdf2$expand.factor) <- c("green", "blue", "red", "yellow")
Stack(testdf1, testdf2)
Stack(testdf2, testdf1)
```

`vecEqual` *Test equality and order of a vector*

Description

Test equality and order of a vector

Usage

```
vecEqual(x, y, order = FALSE)
```

Arguments

`x`	character
`y`	character
`order`	logical. stricter test for order of x and y

Value

logical

`vectorIntegrate` *Integrate characters with careful ordering of result*

Description

Integrate characters with careful ordering of result

Usage

```
vectorIntegrate(x, y)
```

Arguments

x	character vector
y	character vector

Value

union of x and y, with *write something about expected return ordering*.

Index