Package 'ffbase'

April 18, 2012

Maintainer Edwin de Jonge <edwindjonge@gmail.com>

License GPL-3

Title Basic statistical functions for package ff

Type Package

LazyLoad yes

Author Edwin de Jonge, Jan Wijffels

Description Implementation for min,max, mean, tabulate,table, with, within. for package ff

Version 0.4

URL http://code.google.com/p/fffunctions/

Date 2011-11-1

Depends ff (>= 2.2-0)

Suggests testthat, LaF

Collate
'auxilary.R' 'chunkify.R' 'compact.R' 'cut_ff.R' 'diff_ff.R''droplevels.R' 'ffappend.R' 'ffdfsave.R' 'mean.R' 'pkg.R''rispl

R topics documented:

`ffbase-package` *Basic statistical functions for ff*

Description

Basic statistical functions for `ff` vectors and `ffdf` data.frames. The aim of ffbase is to make working with ff vectors and ffdf data.frame a little bit easier.

Details

- `cut.ff`
- `tabulate.ff`
- `table.ff`
- `unique.ff`
- `transform.ffdf`
- `with.ffdf`
- `within.ffdf`

Examples

```
ffdat <- as.ffdf(data.frame(x=1:10, y=10:1))

# add a new ff vector z to the ffdf data.frame
within(ffdat, {z <- x+y})[]

# add a new ff vector z to the ffdf data.frame using transform
transform(ffdat, z=x+y)[]
```

```
cut(ffdat$x, breaks=3)[]

tabulate.ff(ffdat$x)
```

---

all.ff *Summary methods for ff objects*

Description

Summary methods for ff objects

Usage

```
all.ff(x, ..., na.rm = FALSE, range = NULL)
```

Arguments

x	a ff object
...	optional other (ff) objects
na.rm	should NA be removed?
range	a ri or an integer vector of length==2 giving a range restriction for chunked processing

Value

TRUE, FALSE or NA

---

any.ff *Summary methods for ff objects*

Description

Summary methods for ff objects

Usage

```
## S3 method for class 'ff'
any(x, ..., na.rm = FALSE, range = NULL)
```

Arguments

x	a ff object
...	optional other (ff) objects
na.rm	should NA be removed?
range	a ri or an integer vector of length==2 giving a range restriction for chunked processing

Value

TRUE, FALSE or NA

---

c.ff *Concatenate* ff *vectors*

Description

Concatenate ff vectors

Usage

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

Arguments

...	ff ff vectors to be combined

Value

a new ff object, data is physically copied

See Also

link{ffappend}

---

chunkify *Chunkify an element-wise function*

Description

Chunkify creates a new function that operates on a ff vector. It creates chunks from the ff vector and calls the orginal function fun on each chunk.

Usage

```
chunkify(fun)
```

Arguments

fun	function to be 'chunkified', the function must accept a vector and return a vector of the same length

Value

'chunkified' function that accepts a ff vector as its first argument.

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

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

Arguments

x	ff or ffdf object
use.na	logical if TRUE the resulting ff vector can contain NA, otherwise not
...	other parameters

Value

compact ff vector

cut.ff *Convert Numeric ff vector to factor ff*

Description

cut divides the range of x into intervals and codes the values in x according to which interval they fall. The leftmost interval corresponds to level one, the next leftmost to level two and so on.

Usage

```
## S3 method for class 'ff'
cut(x, breaks, ...)
```

Arguments

x	a (numeric) ff object that will be cut into pieces
breaks	specifies the breaks for cutting this
...	other parameters that can be given to cut.default

Details

The cut method for ff with the behaviour of link{cut}

Value

ff a new ff object with the newly created factor

See Also

cut

`droplevels.ff` *The function* `droplevels` *is used to drop unused levels from a* `ff` *factor or , more commonly, from factors in a* `ffdf`

Description

The function `droplevels` is used to drop unused levels from a `ff` factor or , more commonly, from factors in a `ffdf`

Usage

```
  ## S3 method for class 'ff'
droplevels(x, ..., inplace = FALSE)
```

Arguments

`x`	`ff` object
`...`	not used
`inplace`	if `TRUE` the columns will be physically changed, otherwise (default) a new `ff` vector will be created

Value

`ff` object where levels of factors are dropped

See Also

`droplevels` `droplevels.ffdf`

`droplevels.ffdf` *The function* `droplevels` *is used to drop unused levels from factors in a* `ffdf`

Description

The function `droplevels` is used to drop unused levels from factors in a `ffdf`

Usage

```
## S3 method for class 'ffdf'
droplevels(x, except = NULL, ...,
  inplace = FALSE)
```

Arguments

`x`	`ffdf` object
`except`	specify which columns will be excluded from dropping levels
`...`	further arguments passed to `droplevels.ff`
`inplace`	if `TRUE` the columns will be physically changed, otherwise (default) new `ff` vectors will be created

Value

`ffdf` object where levels of factors are dropped

See Also

`droplevels` `droplevels.ff`

`ffappend` *Append a ff vector to another ff vector*

Description

Appends (`ff`) vector `y` to `ff` vector `x`. Please note that the data of `y` will be coerced to the type of `x`.

Usage

```
ffappend(x, y, ...)
```

Arguments

`x`	`ff` object where data will be appended to. If `x==NULL` a new `ff` object will be created
`y`	`ff` object or vector object

Value

ff object with same physical storage as x

See Also

c.ff

---

ffdfappend *append a dataframe to a ffdf*

Description

Appends (ff) vector to ff vector x. Please note that the data of y will be coerced to the type of x.

Usage

```
ffdfappend(x, dat, recode = TRUE, ...)
```

Arguments

x	ffdf object where data will be appended to. If x==NULL a new ff object will be created
dat	ffdf object or data.frame object
recode	should factors be recoded (default), or not (faster)
...	Further arguments passed to as.ffdf, when x==NULL

Value

ffdf object with same physical storage as x

See Also

c.ff

ffdfddply *Performs a split-apply-combine on an ffdf*

Description

Performs a split-apply-combine on an ffdf. Splits the x ffdf according to split and applies FUN to the data, stores the result of the FUN in an ffdf.
Remark that this function does not actually split the data. In order to reduce the number of times data is put into RAM for situations with a lot of split levels, the function extracts groups of split elements which can be put into RAM according to BATCHBYTES. Please make sure your FUN covers the fact that several split elements can be in one chunk of data on which FUN is applied.

Usage

```
ffdfddply(x, split, FUN,
  BATCHBYTES = getOption("ffbatchbytes"),
  RECORDBYTES = sum(.rambytes[vmode(x)]), trace = TRUE,
  ...)
```

Arguments

x	an ffdf
split	an ff vector which is part of the ffdf x
FUN	the function to apply to each split. This function needs to return a data.frame
BATCHBYTES	integer scalar limiting the number of bytes to be processed in one chunk
RECORDBYTES	optional integer scalar representing the bytes needed to process one row of x
trace	logical indicating to show on which split the function is computing
...	other parameters passed on to FUN

Value

an ffdf

See Also

grouprunningcumsum, table.ff

Examples

```
data(iris)
ffiris <- as.ffdf(iris)

result <- ffdfddply(x=ffiris,
split=x$Species,
FUN=function(x){
lowestbypetalwidth <- x[order(x$Petal.Width, decreasing=TRUE), ]
```

```
lowestbypetalwidth <- lowestbypetalwidth[!duplicated(lowestbypetalwidth[, c("Species","Petal.Width")]), ]
lowestbypetalwidth$group <- factor(x= "lowest", levels = c("lowest","highest"))
highestbypetalwidth <- x[order(x$Petal.Width, decreasing=FALSE), ]
highestbypetalwidth <- highestbypetalwidth[!duplicated(highestbypetalwidth[, c("Species","Petal.Width")]), ]
highestbypetalwidth$group <- factor(x= "highest", levels = c("lowest","highest"))
rbind(lowestbypetalwidth, highestbypetalwidth)
},
BATCHBYTES = 5000,
trace=TRUE)
class(result)
dim(result)
dim(iris)
result[1:10,]
```

`ffdfsave` *Save a ffdf data.frame in directory*

Description

`ffdfsave` saves a ffdf data.frame in the given filename (.rdata) and stores all `ff` columns in a subdirectory with the name "<filename>_ff". Each column will be named "<columnname>.ff". A saved ffdf data.frame is a .rdata file and can be loaded with the `load` function

Usage

```
ffdfsave(dat, filename)
```

Arguments

`dat`	`ffdf` data.frame, to be saved
`filename`	path where .rdata file will be save and <filename>_ff directory will be created

Examples

```
data(iris)

# create a ffdf data.frame from standard iris data set
ffiris <- as.ffdf(iris)
head(ffiris[,])

.fn <- tempfile()
ffdfsave(ffiris, .fn)

# clear everything
rm(list=ls())
ls()

# load ffdf into environment
```

```
load(file=.fn)
# and back in business!
head(ffiris[,])
```

`ffwhich` *Create an index from a filter statement* `ffwhich` *creates an* `ff` *integer index vector from a filter expression. The resulting vector can be used to index or subset a ffdf or ff vector.*

Description

Create an index from a filter statement `ffwhich` creates an `ff` integer index vector from a filter expression. The resulting vector can be used to index or subset a ffdf or ff vector.

Usage

```
ffwhich(x, expr, ...)
```

Arguments

`x`	`ff` or `ffdf` object
`expr`	R code that evaluates to a logical
`...`	not used

See Also

ffindexget ffindexset

Examples

```
# create a ff vector
x <- ff(10:1)
# make an ff index vector
idx <- ffwhich(x, x < 5)
# use it to retrieve values from x
x[idx][]

# create a ffdf data.frame
dat <- ffdf(x1=x, y1=x)
# create an ff index vector from a filter statement
idx <- ffwhich(dat, x1 < 5 & y1 > 2)
# use it to select data from the data.frame
dat[idx,][,]
```

grouprunningcumsum *Groups the input integer vector into several groups if the running cumulative sum increases a certain maximum number*

Description

Groups the input integer vector into several groups if the running cumulative sum increases a certain maximum number

Usage

```
grouprunningcumsum(x, max)
```

Arguments

x	an integer vector
max	the maximum running cumulative size before an extra grouping is done

Value

An integer vector of the same length of x, indicating groups

mean.ff *Mean of ff vector*

Description

Mean of ff vector

Usage

```
## S3 method for class 'ff'
mean(x, trim = 0, ..., range = NULL)
```

Arguments

x	a ff vector
trim	percentage of robustness, between 0 and 1
...	other arguments passed to mean
range	a ri or an integer vector of length==2 giving a range restriction for chunked processing

Value

mean value

Examples

```
# create a vector of length 10 million
x <- ff(vmode="double", length=1e7)

mean(x)
```

`mean.ffdf` *Mean of ffdf vector*

Description

Mean of ffdf vector

Usage

```
  ## S3 method for class 'ffdf'
mean(x, ..., range = NULL)
```

Arguments

`x`	a `ffdf`
`...`	other arguments passed to `mean.ff`
`range`	a `ri` or an `integer` vector of `length==2` giving a range restriction for chunked processing

Value

a vector with the mean values

`min.ff` *Minimum, maximum and range of ff vector*

Description

default behaviour of `min`,`max` and `range`

Usage

```
  ## S3 method for class 'ff'
min(x, ..., na.rm = FALSE, range = NULL)
```

Arguments

x	a ff object
...	optional other (ff) objects
na.rm	should NA be removed?
range	a ri or an integer vector of length==2 giving a range restriction for chunked processing

Value

minimun, maximum or range values

Examples

```
x <- ff(1:100)

min(x)
max(x)
range(x)

is.na(x) <- 10
min(x)
max(x)
range(x)


min(x, na.rm=TRUE)
max(x, na.rm=TRUE)
range(x, na.rm=TRUE)
```

---

subset.ff *Subsetting a ff vector or ffdfdata frame*

Description

Subsetting a ff vector or ffdfdata frame

Usage

```
  ## S3 method for class 'ff'
subset(x, subset, ...)
```

Arguments

x	ff vector or ffdf data.frame to be subset
subset	an expression, ri, bit or logical ff vector that can be used to index x
...	not used

Value

a new ff vector containing the subset, data is physically copied

---

`sum.ff` *Sum of* `ff` *vector Elements*

Description

`sum` returns the sum of all the values present in its arguments.

Usage

```
 ## S3 method for class 'ff'
sum(x, ..., na.rm = FALSE, range = NULL)
```

Arguments

`x`	a `ff` object
`...`	optional other (`ff`) objects
`na.rm`	should `NA` be removed?
`range`	a `ri` or an `integer` vector of `length==2` giving a range restriction for chunked processing

Value

sum of elements

`table.ff` *table.ff uses the cross-classifying factors to build a contingency table of the counts at each combination of factor levels.*

Description

Details

Usage

```
table.ff(..., exclude = if (useNA == "no") c(NA, NaN),
  useNA = c("no", "ifany", "always"),
  dnn = list.names(...), deparse.level = 1)
```

Arguments

...	ff factors
exclude	see table
useNA	see table
dnn	see table
deparse.level	see table

Value

table object

See Also

table

---

tabulate.ff *Tabulation for ff vectors*

Description

tabulate.ff takes the integer-valued ff vector bin and counts the number of times each integer occurs in it.

Usage

```
tabulate.ff(bin, nbins = max(bin, 1, na.rm = TRUE))
```

Arguments

bin	factor to be binned.
nbins	number of bins

Details

Behaviour of tabulate

Value

integer vector or if FFRETURN is TRUE a ff vector

Examples

```
#create a vector of 10 million
x <- ff(vmode="integer", length=1e7)

# fill first 200 with values
x[1:100] <- 1
x[101:200] <- 2

# lets count
tabulate.ff(x)
```

`transform.ffdf` *Transform a ffdf data.frame*

Description

Same functionality as `transform`, but on a `ffdf` object. Please note that you should write your expression as if it is a normal `data.frame`. The resulting data.frame however will be a `ffdf` data.frame.

Usage

```
  ## S3 method for class 'ffdf'
transform(`_data`, ...)
```

Arguments

`_data`	`ffdf` data object to be transformed.
`...`	named arguments that will be added to the `ffdf` data.frame

Value

a modified clone of `_data`.

Examples

```
transform(as.ffdf(airquality), Ozone = -Ozone)
transform(as.ffdf(airquality), new = -Ozone, Temp = (Temp-32)/1.8)
```

unique.ff *Unique values*

Description

Unique values

Usage

```
## S3 method for class 'ff'
unique(x, incomparables = FALSE,
   fromLast = FALSE, ...)
```

Arguments

x	ff object
incomparables	a vector of values that cannot be compared. FALSE is a special value, meaning that all values can be compared, and may be the only value accepted for methods other than the default. It will be coerced internally to the same type as x.
fromLast	logical indicating if duplication should be considered from the last, i.e., the last (or rightmost) of identical elements will be kept
...	Further arguments passed to unique

Value

vector with unique values of x

with.ffdf *Evaluate an expression in a ffdf data environment*

Description

Evaluate an R expression in an environment constructed from a ffdata data frame, possibly modifying the original data. (see with). Please note that you should write your expression as if it is a normal data.frame. The resulting return value however will be a ff object.

Usage

```
## S3 method for class 'ffdf'
with(data, expr, ...)
```

Arguments

data	ffdf data object used as an environment for evaluation.
expr	expression to evaluate.
...	arguments to be passed to future methods.

Value

if expression is a `vector` a newly created `ff` vector will be returned otherwise if the expression is a data.frame a newly created `ffdf` object will be returned.

Examples

```
dat <- data.frame(x=1:10, y=10:1)

ffdat <- as.ffdf(dat)

with(dat, {x+y})
```

`within.ffdf` *Evaluate an expression in a ffdf data environment*

Description

Same functionality as `within`. Please note that you should write your expression as if it is a normal `data.frame`. The resulting data.frame however will be a `ffdf` data.frame.

Usage

```
  ## S3 method for class 'ffdf'
within(data, expr, ...)
```

Arguments

`data`	`ffdf` data object used as an environment for evaluation.
`expr`	expression to evaluate.
`...`	arguments to be passed to future methods.

Value

a modified clone of `data`.

Examples

```
ffdat <- as.ffdf(data.frame(x=1:10, y=10:1))
# add z to the ffdat
within(ffdat, {z <- x+y})
```

Index