Package 'DatABEL'

March 31, 2015

Type Package

Title File-Based Access to Large Matrices Stored on HDD in Binary Format

Version 0.9-6

Date 2015-03-30

Author Yurii Aulchenko, Stepan Yakovenko, Erik Roos, Marcel Kempenaar, Maksim Struchalin, Lennart C. Karssen

Maintainer Lennart Karssen <lennart@karssen.org>

Depends R (>= 2.4.0), methods, utils

Suggests GenABEL, RUnit

Description Provides an interface to the C++ FILEVECTOR library facilitating analysis using large (giga- to tera-bytes) matrices. Matrix storage is organized in a way that either columns or rows are quickly accessible. DatABEL is primarily aimed to support genome-wide association analyses e.g. using GenABEL, MixABEL and ProbABEL.

License GPL (>= 2)

NeedsCompilation yes

Repository CRAN

Date/Publication 2015-03-31 07:46:13

R topics documented:

`DatABEL-package` *DatABEL package for fast consecutive access to large out-of-RAM stored matrices*

Description

A package interfacing FILEVECTOR C++ library for storage of and fast consecutive access to large data matrices in out-of-RAM disk mode with regulated cache size. Columns of matrix are accessible very quickly.

Author(s)

Yurii Aulchenko (R code), Stepan Yakovenko (R and C++ code), Andrey Chernyh (C++ code)

See Also

`apply2dfo`, `databel2matrix`, `databel2text`, `extract_text_file_columns`, `matrix2databel`, `text2databel`, `databel`

`apply2dfo` *applies a function to a 'databel' object*

Description

An iterator applying a user-defined function to an object of 'databel-class'.

Usage

```
apply2dfo(..., dfodata, anFUN = "lm", MAR = 2, procFUN,
  outclass = "matrix", outfile, type = "DOUBLE", transpose = TRUE)
```

Arguments

`...`	arguments passed to the `anFUN`
`dfodata`	'databel' object which is iterated over
`anFUN`	user-defined analysis function
`MAR`	which margin to iterate over (default = 2, usually these are 'columns' used to store SNP data)

procFUN	function to process the output and present that as a fixed-number-of-columns matrix or fixed-length vector. Can be missing if standard functions listed below are used. Pre-defined processors included are "process_lm_output" (can process functions "lm", "glm", "coxph") and "process_simple_output" (process output from "sum", "prod", "sum_not_NA" [no. non-missing obs], "sum_NA" [no. missing obs.])
outclass	output to ("matrix" or "databel")
outfile	if output class is "databel", the generated object is bound to the outfile
type	if output class is "databel", what data type to use for storage
transpose	whether to transpose the output

Value

A matrix (or 'databel'-matrix) containing results of applying the function

Author(s)

Yurii Aulchenko

Examples

```
a <- matrix(rnorm(50), 10, 5)
rownames(a) <- paste("id", 1:10, sep="")
colnames(a) <- paste("snp", 1:5, sep="")
b <- as(a, "databel")
apply(a, FUN="sum", MAR=2)
apply2dfo(SNP, dfodata=b, anFUN="sum")
tA <- apply2dfo(SNP, dfodata=b, anFUN="sum",
                outclass="databel", outfile="tmpA")
tA
as(tA, "matrix")
apply2dfo(SNP, dfodata=b, anFUN="sum", transpose=FALSE)
tB <- apply2dfo(SNP, dfodata=b, anFUN="sum", transpose=FALSE,
                outclass="databel", outfile="tmpB")
tB
as(tB, "matrix")

sex <- 1*(runif(10)>.5)
trait <- rnorm(10) + sex + as(b[, 2], "vector") +
         as(b[, 2], "vector") * sex * 5
apply2dfo(trait~SNP*sex, dfodata=b, anFUN="lm")
tC <- apply2dfo(trait ~ SNP * sex, dfodata=b, anFUN="lm",
                outclass="databel", outfile="tmpC")
tC
as(tC, "matrix")
apply2dfo(trait ~ SNP * sex, dfodata=b, anFUN="lm", transpose=FALSE)
tD <- apply2dfo(trait ~ SNP * sex, dfodata=b, anFUN="lm",
                transpose=FALSE, outclass="databel", outfile="tmpD")
tD
as(tD, "matrix")
```

```
rm(tA, tB, tC, tD)
gc()
unlink("tmp*")
```

`databel` *initiates databel object*

Description

this is a simple wrapper for the "new" function creating a databel object

Usage

```
databel(baseobject, cachesizeMb = 64, readonly = TRUE)
```

Arguments

`baseobject`	name of the file or databel-class object
`cachesizeMb`	cache size (amount of RAM) to be used
`readonly`	readonly flag

Author(s)

Yurii Aulchenko

`databel-class` *DatABEL class*

Description

DatABEL stores matrix-shape data in such a way that it can be retrieved fast.

Usage

```
## S4 method for signature 'databel'
show(object)

## S4 method for signature 'databel'
dim(x)

## S4 method for signature 'databel'
length(x)

## S4 method for signature 'databel'
dimnames(x)
```

```
## S4 replacement method for signature 'databel'
dimnames(x) <- value

## S4 method for signature 'databel'
x[i, j, drop]

## S4 replacement method for signature 'databel'
x[i, j] <- value

get_dimnames(object)

set_dimnames(x) <- value

backingfilename(object)

cachesizeMb(object)

cachesizeMb(x) <- value

save_as(x, rows, cols, file, cachesizeMb = 64, readonly = TRUE)

connect(object, readonly = TRUE)

disconnect(object)

setReadOnly(x) <- value
```

Arguments

object	A DatABEL object
x	A DatABEL object
value	Values to be replaced/inserted
i	Row index
j	Column index
drop	Boolean (FALSE by default); UNUSED
rows	Index for the rows
cols	Index for the columns
file	Filename to save to
cachesizeMb	Amount (in MB) of RAM to use for caching DatABEL data.
readonly	Boolean that specifies whether the file is to be used in read-only mode or not

Slots

usedRowIndex ("integer")

usedColIndex ("integer")

`uninames` (`"list"`)

`backingfilename` Name of the (stem of the) file that contains the data stored in DatABEL format (`"character"`)

`cachesizeMb` Amount (in MB) of RAM to use for caching DatABEL data. (`"integer"`)

`data` (`"externalptr"`)

Note

Will extend description here

Author(s)

Yurii Aulchenko

`databel2matrix` *converts 'databel' to matrix*

Description

Converts a `databel` object to a regular R matrix. This is the procedure used by the "as" converting to DatABEL objects, in which case a temporary file name is created.

Usage

```
databel2matrix(from, rows, cols)
```

Arguments

`from`	'databel' matrix
`rows`	which rows to include
`cols`	which columns to include

Value

object of `matrix` class

Author(s)

Stepan Yakovenko

databel2text *Exports DatABEL object to a text file*

Description

Exports DatABEL object to a text file

Usage

```
databel2text(databel, file, NAString = "NA", row.names = TRUE,
  col.names = TRUE, transpose = FALSE)
```

Arguments

databel	DatABEL object
file	output file name
NAString	string to replace NA with
row.names	export row names if TRUE
col.names	export col names if TRUE
transpose	whether the matrix should be transposed

Author(s)

Stepan Yakovenko

extract_text_file_columns *extracts columns from text file*

Description

Extracts a column from text file to a matrix. If in a particular file line the number of columns is less then a column specified, returns last column!

Usage

```
extract_text_file_columns(file, whichcols)
```

Arguments

file	file name
whichcols	which columns to extract

Value

matrix of strings with values from that columns

get_temporary_file_name *generates temporary file name*

Description

function to generate temporary file name

Usage

```
get_temporary_file_name(path = ".", withFVext = TRUE)
```

Arguments

path	path to directory where the temporary file will be located
withFVext	whether function should check presence of *FVD and *FVI files too

make_empty_fvf *makes empty filevector object*

Description

function to generate empty filevector object (and disk files)

Usage

```
make_empty_fvf(name, nvariables, nobservations, type = "DOUBLE",
  cachesizeMb = 64, readonly = FALSE)
```

Arguments

name	name fo the file to be assoiated with new object
nvariables	number of variables (R columns)
nobservations	number of observations (R rows)
type	data type of the object ("UNSIGNED_SHORT_INT", "SHORT_INT", "UNSIGNED_INT", "INT", "FLOAT", "DOUBLE", "CHAR", "UNSIGNED_CHAR")
cachesizeMb	what cache size to use for newly generated 'databel' object
readonly	whether to open new 'databel' in readonly mode

Value

databel object; also file is created in file system

matrix2databel *converts matrix to 'databel'*

Description

Converts regular R matrix to `databel` object. This is the procedure used by "as" converting to DatABEL objects, in which case a temporary file name is created

Usage

```
matrix2databel(from, filename, cachesizeMb = 64, type = "DOUBLE",
  readonly = FALSE)
```

Arguments

`from`	R matrix
`filename`	which FILEVECTOR BASE file name to use
`cachesizeMb`	cache size to be used when accessing the object
`type`	type of data to use for storage ("DOUBLE", "FLOAT", "INT", "UNSIGNED_INT", "UNSIGNED_SHORT_INT", "SHORT_INT", "CHAR", "UNSIGNED_CHAR")
`readonly`	whether to generate new 'databel' in read only mode

Value

object of class `databel`

Author(s)

Yurii Aulchenko

process_lm_output *'apply2dfo'-associated functions*

Description

A number of functions used in conjunction with 'apply2dfo'. Standardly supported apply2dfo's anFUN analysis functions include 'lm', 'glm', 'coxph', 'sum', 'prod', "sum_not_NA" (no. non-missing obs), and "sum_NA" (no. missing obs.). Pre-defined processing functions include "process_lm_output" (can process functions "lm", "glm", "coxph") and "process_simple_output" (process output from "sum", "prod", "sum_not_NA", "sum_NA")

Usage

```
process_lm_output(lmo, verbosity=2)
```

Arguments

lmo	object returned by analysis with "lm", "glm", etc.
verbosity	verbosity

See Also

apply2dfo

Examples

```
a <- matrix(rnorm(50),10,5)
rownames(a) <- paste("id",1:10,sep="")
colnames(a) <- paste("snp",1:5,sep="")
b <- as(a,"databel")
apply(a,FUN="sum",MAR=2)
apply2dfo(SNP,dfodata=b,anFUN="sum",procFUN="process_simple_output")
apply2dfo(SNP,dfodata=b,anFUN="sum",transpose=FALSE)

sex <- 1*(runif(10)>.5)
trait <- rnorm(10)+sex+as(b[,2],"vector")+as(b[,2],"vector")*sex*5
apply2dfo(trait~SNP*sex,dfodata=b,anFUN="lm",procFUN="process_lm_output")
```

---

text2databel *converts text file to filevector format*

Description

The file provides the data to be converted to filevector format. The file may provide the data only (no row and column names) in which case col/row names may be left empty or provided in separate files (in which case it is assumed that names are provided only for the imported columns/rows – see skip-options). There is an option to skip a number of first ros and columns. The row and column names may also be provided in the file itself, in which case one needs to tell the row/column number providing column/row names. Unless option "R_matrix" is set to TRUE, it is asumed that the number of columns is always the same acorss the file. If above option is provided, it is assumed that both column and row names are provided in the file, and the first line contains one column less than other lines (such is the case with files produced from R using the function `write.table(...,col.names=TRUE,row.names=TRUE)`.

Usage

```
text2databel(infile, outfile, colnames, rownames, skipcols, skiprows,
  transpose = FALSE, R_matrix = FALSE, type = "DOUBLE",
  cachesizeMb = 64, readonly = TRUE, naString = "NA",
  unlinkTmpTransposeFiles = TRUE)
```

Arguments

infile	input text file name
outfile	output filevector file name; if missing, it is set to infile+".filevector"
colnames	where are the column names stored? If missing, no column names; if integer, this denotes the row of the input file where the column names are specified; if character string then the string specifies the name of the file with column names
rownames	where are the row names stored? If missing, no row names; if integer, this denotes the column of the input file where the row names are specified; if character string then the string specifies the name of the file with row names
skipcols	how many columns of the input file to skip
skiprows	how many rows of the input file to skip
transpose	whether the file is to be transposed
R_matrix	if true, the file format is assumed to follow the format of R data matrix produced with `write.table(...,col.names=TRUE,row.names=TRUE)`
type	data DatABEL type to use ("DOUBLE", "FLOAT", "INT", "UNSIGNED_INT", "UNSIGNED_SHORT_INT", "SHORT_INT", "CHAR", "UNSIGNED_CHAR")
cachesizeMb	cache size for the resulting 'databel-class' object
readonly	whether the resulting 'databel-class' object should be opened in readonly mode
naString	the string used for missing data (default: NA)
unlinkTmpTransposeFiles	Boolean to indicate whether the intermediate "_fvtmp.fvi/d" files should be deleted. Default: TRUE. These intermediate files are generated while transposing the filevector files.

Value

The converted file is stored in the file system, a databel-class object connection to the file is returned.

Author(s)

Yurii Aulchenko

Examples

```
cat("this is an example which you can run if you can write to the
file system\n")

## Not run:

# create matrix
NC <- 5
NR <- 10
data <- matrix(rnorm(NC*NR),ncol=NC,nrow=NR)
rownames(data) <- paste("r",1:NR,sep="")
colnames(data) <- paste("c",1:NC,sep="")
data
```

```
# create text files
write.table(data, file="test_matrix_dimnames.dat", row.names=TRUE,
            col.names=TRUE, quote=FALSE)
write.table(data, file="test_matrix_colnames.dat", row.names=FALSE,
            col.names=TRUE, quote=FALSE)
write.table(data, file="test_matrix_rownames.dat", row.names=TRUE,
            col.names=FALSE, quote=FALSE)
write.table(data, file="test_matrix_NOnames.dat", row.names=FALSE,
            col.names=FALSE, quote=FALSE)
write(colnames(data), file="test_matrix.colnames")
write(rownames(data), file="test_matrix.rownames")

# generate identical data
text2databel(infile="test_matrix_dimnames.dat",
             outfile="test_matrix_dimnames", R_matrix=TRUE)
x <- databel("test_matrix_dimnames")
data <- as(x, "matrix")
data

# convert text two filevector format

text2databel(infile="test_matrix_NOnames.dat",
             outfile="test_matrix_NOnames.fvf",
             colnames="test_matrix.colnames",
             rownames="test_matrix.rownames")
x <- databel("test_matrix_NOnames.fvf")
if (!identical(data, as(x, "matrix"))) stop("not identical data")

text2databel(infile="test_matrix_NOnames.dat",
             outfile="test_matrix_NOnames_T.fvf",
             colnames="test_matrix.colnames",
             rownames="test_matrix.rownames", transpose=TRUE)
x <- databel("test_matrix_NOnames_T.fvf")
if (!identical(data, t(as(x, "matrix")))) stop("not identical data")

text2databel(infile="test_matrix_rownames.dat",
             outfile="test_matrix_rownames.fvf",
             rownames=1, colnames="test_matrix.colnames")
x <- databel("test_matrix_rownames.fvf")
if (!identical(data, as(x, "matrix"))) stop("not identical data")

text2databel(infile="test_matrix_colnames.dat",
             outfile="test_matrix_colnames.fvf",
             colnames=1, rownames="test_matrix.rownames")
x <- databel("test_matrix_colnames.fvf")
if (!identical(data, as(x, "matrix"))) stop("not identical data")

text2databel(infile="test_matrix_dimnames.dat",
             outfile="test_matrix_dimnames.fvf", R_matrix=TRUE)
x <- databel("test_matrix_dimnames.fvf")
if (!identical(data, as(x, "matrix"))) stop("not identical data")
```

```
# stupid extended matrix in non-R format
newmat <- matrix(-100, ncol=NC+3, nr=NR+2)
newmat[3:(NR+2), 4:(NC+3)] <- data
newmat[2, 4:(NC+3)] <- paste("c", 1:NC, sep="")
newmat[3:(NR+2), 3] <- paste("r", 1:NR, sep="")
newmat
write.table(newmat, file="test_matrix_strange.dat",
            col.names=FALSE, row.names=FALSE, quote=FALSE)

text2databel(infile="test_matrix_strange.dat",
             outfile="test_matrix_strange.fvf",
             colnames=2, rownames=3)
x <- databel("test_matrix_strange.fvf")
if (!identical(data, as(x, "matrix"))) stop("not identical data")


## End(Not run)
```

Index