Package 'DatABEL'

March 7, 2010

Type Package

Title file-based access to huge matrices stored in binary format

Version 0.0-5

Date 2010-02-16

Author Yurii Aulchenko

Maintainer Yurii Aulchenko <i.aoultchenko@erasmusmc.nl>

Depends R (>= 2.4.0), methods

Description a package providing interface to DatABELCPP/filevector library facilitating analysis of large (giga- to tera-bytes) amounts of data; matrix storage is organized in a way that either columns or rows are quickly accessible; primarily aimed to support genome-wide association analyzes e.g. using GenABEL

License GPL (>= 2)

R topics documented:

DatABEL-package *DatABEL package...*

Description

DatABEL package

Details

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

See Also

`make_empty_fvf`, `matrix2databel_base_R`, `databel_base_R-class`, `databel_filtered_R-class`

apply2dfo *applies a function to 'databel_filtered_R' object...*

Description

applies a function to 'databel_filtered_R' object

Usage

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

Arguments

`dfodata`	'databel_filtered-R' object which is iterated over
`anFUN`	user-defined analysis function
`MAR`	which margin to iteracte 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_filtered_R")
`outfile`	if output class is "databel_filtered_R", the generated object is bond to the outfile

`type`	if output class is "databel_filtered_R", what data tyoe to use for storage
`transpose`	whether to transpose the output
`...`	arguments passed to the anFUN

Details

An iterator applying a user-defined function to databel_filtered_R - class object

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_filtered_R")
apply(a,FUN="sum",MAR=2)
apply2dfo(SNP,dfodata=b,anFUN="sum")
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")
```

`databel_base_R` *initiates databel_base_R object...*

Description

initiates databel_base_R object

Usage

```
databel_base_R(filename, cachesizeMb=64)
```

Arguments

`filename`	name of the file containing the data
`cachesizeMb`	what cache size (size of RAM) to be used

Details

initiates databel_base_R object

Author(s)

Yurii Aulchenko

databel_base_R-class *Class "databel_base_R"*

Description

A class 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.

Objects from the Class

Objects can be created by calls of the form `new("databel_base_R", filename, cachesizeMb)`. FILEVECTOR data are stored using files of form BASE.fvi (index) and BASE.fvd (data). "filename" is the BASE name.

Slots

`backingfilename`**:** Object of class `"character"` providing BASE name

`cachesizeMb`**:** Object of class `"integer"` size of cache to be used to access the data. If cache is equal to the data size, the object is stored in RAM

`data`**:** Object of class `"externalptr"`, pointer to AbstractFileVector C++ object

Methods

[`signature(x = "databel_base_R")`: sub-setting object

[<- `signature(x = "databel_base_R")`: setting the values in the object

backingfilename `signature(object = "databel_base_R")`: returns BASE FILEVECTOR file name used to store the data

cachesizeMb `signature(object = "databel_base_R")`: returns the cache size used

cachesizeMb<- `signature(x = "databel_base_R")`: sets new cache size

connect `signature(object = "databel_base_R")`: connects the data files to R object (calls constructor of AbstractFileVeector object)

dim `signature(x = "databel_base_R")`: returns dimensions of the matrix

dimnames `signature(x = "databel_base_R")`: dummy, returns NULL

disconnect `signature(object = "databel_base_R")`: disconnects the R object from data files (calls destructor of AbstractFileVeector object)

get_dimnames `signature(object = "databel_base_R")`: returns the names of rows and columns, which may be non-unique

length `signature(x = "databel_base_R")`: returns number of elements in the matrix

save_as `signature(x = "databel_base_R")`: saves (a sub-set of) the object

set_dimnames<- `signature(x = "databel_base_R")`: set row and column names, which may be non-unique

Author(s)

Yurii Aulchenko

References

http://mga.bionet.nsc.ru/~yurii/ABEL/

See Also

`make_empty_fvf`, `databel_filtered_R-class`

Examples

```
showClass("databel_base_R")
```

`databel_filtered_R` *initiates databel_filtered_R object...*

Description

initiates databel_filtered_R object

Usage

```
databel_filtered_R(file_or_baseobject, cachesizeMb)
```

Arguments

`file_or_baseobject`	name of the file or databel_base_R-class object
`cachesizeMb`	cache size (amount of RAM) to be used

Details

initiates databel_filtered_R object

Author(s)

Yurii Aulchenko

`databel_filtered_R-class` *Class "databel_filtered_R"*

Description

A class 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. The 'filtered' class is based on 'base' class, see methods for that class for more details.

Objects from the Class

Objects can be created by calls of the form `new("databel_filtered_R", baseobject)`. FILEVECTOR data are stored using files of form BASE.fvi (index) and BASE.fvd (data). "baseobject" is either the BASE name, or object of class `"databel_base_R"`.

Slots

- `usedRowIndex`: Object of class "`integer`" which rows are used
- `usedColIndex`: Object of class "`integer`" which columns are used
- `unique.names`: Object of class "`logical`" if all dimnames are unique
- `unique.colnames`: Object of class "`logical`" if column names are unique
- `unique.rownames`: Object of class "`logical`" if row names are unique
- `backingfilename`: Object of class "`character`" providing BASE name
- `cachesizeMb`: Object of class "`integer`" size of cache to be used to access the data. If cache is equal to the data size, the object is stored in RAM
- `data`: Object of class "`externalptr`", pointer to AbstractFileVector C++ object

Extends

Class "`databel_base_R`", directly, with explicit coerce.

Methods

- **[** `signature(x = "databel_filtered_R")`: sub-setting object
- **[<-** `signature(x = "databel_filtered_R")`: setting the values in the object
- **connect** `signature(object = "databel_filtered_R")`: connects the data files to R object (calls constructor of AbstractFileVeector object, selects rows and columns)
- **dim** `signature(x = "databel_filtered_R")`: returns dimensions of the matrix
- **dimnames** `signature(x = "databel_filtered_R")`: returns row and column names
- **dimnames<-** `signature(x = "databel_filtered_R")`: sets row and column names
- **length** `signature(x = "databel_filtered_R")`: returns number of elements in the matrix
- **save_as** `signature(x = "databel_filtered_R")`: saves (a sub-set of) the object

Author(s)

Yurii Aulchenko

References

`http://mga.bionet.nsc.ru/~yurii/ABEL/`

See Also

`make_empty_fvf`, `matrix2databel_base_R`, `databel_base_R-class`

Examples

```
showClass("databel_filtered_R")
```

extract_text_file_columns *extract_text_file_columns*

Description

extracts columns from text file

Usage

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

Arguments

`file`	file name
`whichcols`	which columns to extract

Details

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!

Value

matrix of strings with values from that columns

get_temporary_file_name *generates temporary file name...*

Description

generates 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

Details

function to generate temporary file name

`make_empty_fvf` *makes empty filevector object...*

Description

makes empty filevector object

Usage

```
make_empty_fvf(name, nvariables, nobservations, type="DOUBLE")
```

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")

Details

function to generate empty filevector object (and disk files)

Value

databel_filtered_R object; also file is created in file system

`matrix2databel_base_R` *converts matrix to 'databel_base_R'...*

Description

converts matrix to 'databel_base_R'

Usage

```
matrix2databel_base_R(from, filename, cachesizeMb=64, type="DOUBLE")
```

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")

Details

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

Value

object of class `databel_base_R-class`

Author(s)

Yurii Aulchenko

`process_lm_output` *'apply2dfo'-associated functions...*

Description

'apply2dfo'-associated functions

Usage

```
process_lm_output(lmo,verbosity=2)
process_simple_output(o)
sum_not_NA(x)
sum_NA(x)
```

Arguments

`lmo`	object returned by analysis with "lm", "glm", etc.
`verbosity`	verbosity
`o`	output for processing
`x`	vector of data on which function is applied

Details

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")

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_filtered_R")
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")
```

`text2filevector` *converts text file to filevector format...*

Description

converts text file to filevector format

Usage

```
text2filevector(infile, outfile, colnames, rownames, skipcols,
    skiprows, transpose=FALSE, R_matrix=FALSE, type="DOUBLE")
```

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.mnames=TRUE,row.names=TRUE)"
`type`	data DatABEL type to use ("DOUBLE", "FLOAT", "INT", "UNSIGNED_INT", "UNSIGNED_SHORT_INT", "SHORT_INT")

Details

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 file produced from R using function "write.table(...,col.mnames=TRUE,row.names=TRUE)"

Value

file converted is stored in file system, databel_filtered_R object connection to the file

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=FALS
write.table(data,file="test_matrix_colnames.dat",row.names=FALSE,col.names=TRUE,quote=FAL
write.table(data,file="test_matrix_rownames.dat",row.names=TRUE,col.names=FALSE,quote=FAL
write.table(data,file="test_matrix_NOnames.dat",row.names=FALSE,col.names=FALSE,quote=FAL
write(colnames(data),file="test_matrix.colnames")
write(rownames(data),file="test_matrix.rownames")

# generate identical data
text2filevector(infile="test_matrix_dimnames.dat",outfile="test_matrix_dimnames",R_matrix
x <- databel_filtered_R("test_matrix_dimnames")
data <- as(x,"matrix")
data

# convert text two filevector format

text2filevector(infile="test_matrix_NOnames.dat",outfile="test_matrix_NOnames.fvf",
colnames="test_matrix.colnames",rownames="test_matrix.rownames")
x <- databel_filtered_R("test_matrix_NOnames.fvf")
if (!identical(data,as.matrix(x))) stop("not identical data")
```

```
text2filevector(infile="test_matrix_NOnames.dat",outfile="test_matrix_NOnames_T.fvf",
colnames="test_matrix.colnames",rownames="test_matrix.rownames",transpose=TRUE)
x <- databel_filtered_R("test_matrix_NOnames_T.fvf")
if (!identical(data,t(as.matrix(x)))) stop("not identical data")

text2filevector(infile="test_matrix_rownames.dat",outfile="test_matrix_rownames.fvf",
rownames=1,colnames="test_matrix.colnames")
x <- databel_filtered_R("test_matrix_rownames.fvf")
if (!identical(data,as.matrix(x))) stop("not identical data")

text2filevector(infile="test_matrix_colnames.dat",outfile="test_matrix_colnames.fvf",
colnames=1,rownames="test_matrix.rownames")
x <- databel_filtered_R("test_matrix_colnames.fvf")
if (!identical(data,as.matrix(x))) stop("not identical data")

text2filevector(infile="test_matrix_dimnames.dat",outfile="test_matrix_dimnames.fvf",R_ma
x <- databel_filtered_R("test_matrix_dimnames.fvf")
if (!identical(data,as.matrix(x))) 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=F

text2filevector(infile="test_matrix_strange.dat",outfile="test_matrix_strange.fvf",
colnames=2,rownames=3)
x <- databel_filtered_R("test_matrix_strange.fvf")
if (!identical(data,as.matrix(x))) stop("not identical data")


## End(Not run)
```

Index