

# Package ‘beachmat’

October 15, 2018

**Version** 1.2.1

**Date** 2018-04-25

**Title** Compiling Bioconductor to Handle Each Matrix Type

**Encoding** UTF-8

**Maintainer** Aaron Lun <alun@wehi.edu.au>

**Depends** R (>= 3.5)

**Imports** utils, methods, Rhdf5lib (>= 1.1.4), rhdf5, HDF5Array (>= 1.7.3), DelayedArray (>= 0.5.30), Rcpp (>= 0.12.14)

**Suggests** testthat, BiocStyle, knitr, rmarkdown, Matrix, devtools

**biocViews** DataRepresentation, DataImport, Infrastructure

**Description** Provides a consistent C++ class interface for a variety of commonly used matrix types, including sparse and HDF5-backed matrices.

**License** GPL-3

**NeedsCompilation** yes

**VignetteBuilder** knitr

**SystemRequirements** C++11

**LinkingTo** Rcpp, Rhdf5lib

**RoxygenNote** 6.0.1

**git\_url** <https://git.bioconductor.org/packages/beachmat>

**git\_branch** RELEASE\_3\_7

**git\_last\_commit** ebae817

**git\_last\_commit\_date** 2018-05-23

**Date/Publication** 2018-10-15

**Author** Aaron Lun [aut, cre],  
Hervé Pagès [aut],  
Mike Smith [aut]

## R topics documented:

getBestChunkDims . . . . .	2
pkgconfig . . . . .	2
rechunkByMargins . . . . .	3

<b>Index</b>	<b>5</b>
--------------	----------

---

getBestChunkDims      *Get best chunk dimensions*

---

### Description

Computes the optimal chunk dimensions for consecutive row/column access from a HDF5Matrix.

### Usage

```
getBestChunkDims(dims)
```

### Arguments

dims                    An integer vector of length 2 containing the dimensions of a HDF5Matrix object.

### Details

Consider a HDF5Matrix where access to consecutive rows or columns is requested. The optimal chunk dimensions ensure that the number of disk reads required is the same as that of a file layout with pure row or column chunks. This exploits the HDF5 chunk cache to store data from neighbouring rows/columns, avoiding the need to reread or rewrite the entire chunk for the next row/column. Obviously, this is not relevant to situations involving random row or column access.

### Value

An integer vector of length 2, containing the dimensions of each chunk in the HDF5 file.

### Author(s)

Aaron Lun

### Examples

```
getBestChunkDims(c(10340, 234))
getBestChunkDims(c(13400, 2068))
```

---

pkgconfig                    *Compiler configuration arguments for use of beachmat*

---

### Description

This function returns values for PKG\_LIBS and PKG\_CPPFLAGS variables for use in Makevars files. See vignette("beachmat", "beachmat") for details. Only PKG\_LIBS should be needed in most cases. The environment variable RBEACHMAT\_RPATH can be used to over-ride the inferred location of the installed package.

### Usage

```
pkgconfig(opt = c("PKG_LIBS", "PKG_CPPFLAGS"))
```

**Arguments**

opt                    A string specifying the compilation flags to print.

**Value**

Returns NULL and prints the corresponding value to stdout.

**Author(s)**

Aaron Lun

**Examples**

```
pkgconfig("PKG_LIBS")
pkgconfig("PKG_CPPFLAGS")
```

---

rechunkByMargins	<i>Rechunk by margins</i>
------------------	---------------------------

---

**Description**

Convert an existing HDF5Matrix into a pure column- or row-based chunk layout.

**Usage**

```
rechunkByMargins(x, size=5000, outfile=NULL, outname=NULL,
  outlevel=NULL, byrow=TRUE)
```

**Arguments**

x                    A HDF5Matrix object.

size                An integer scalar specifying the number of elements in each chunk.

outfile            A string containing the name for the output HDF5 file, chosen by [getHDF5DumpFile](#) if not specified.

outname            A string containing the name for the output HDF5 data set, chosen by [getHDF5DumpName](#) if not specified.

outlevel           An integer scalar specifying the compression level, chosen by [getHDF5DumpCompressionLevel](#) if not specified.

byrow              A logical scalar indicating if the output file should be row-chunked (default) or column-chunked.

**Details**

Pure column- or row-based chunk layouts are optimal for random column and row access, respectively, from a HDF5 file. This function can be used to convert a file into a pure row/column layout prior to calling other functions. In many cases, a small investment in rechunking time is repaid by a reduction in access times in downstream procedures.

**Value**

A HDF5Matrix object pointing to the HDF5 file containing the data from `x` but with the new chunk layout.

**Author(s)**

Aaron Lun

**Examples**

```
A <- as(matrix(runif(5000), nrow=100, ncol=50), "HDF5Array")
byrow <- rechunkByMargins(A, byrow=TRUE)
bycol <- rechunkByMargins(A, byrow=FALSE)
```

# Index

\*Topic **manip**

pkgconfig, [2](#)

getBestChunkDims, [2](#)

getHDF5DumpCompressionLevel, [3](#)

getHDF5DumpFile, [3](#)

getHDF5DumpName, [3](#)

pkgconfig, [2](#)

rechunkByMargins, [3](#)