Package: bigtabulate (via r-universe)

August 28, 2024

Version 1.1.9

Title Table, Apply, and Split Functionality for Matrix and 'big.matrix' Objects

Author Michael J. Kane <kaneplusplus@gmail.com> and John W. Emerson <jayemerson@gmail.com>

Maintainer Michael J. Kane <bigmemoryauthors@gmail.com>

Contact Jay and Mike <bigmemoryauthors@gmail.com>

Depends bigmemory (>= 4.0.0), biganalytics

LinkingTo Rcpp, BH, bigmemory

Description Extend the bigmemory package with 'table', 'tapply', and 'split' support for 'big.matrix' objects. The functions may also be used with native R matrices for improving speed and memory-efficiency.

URL http://www.bigmemory.org

LazyLoad yes

Biarch yes

RoxygenNote 7.1.2

NeedsCompilation yes

Repository CRAN

Date/Publication 2022-04-11 14:12:43 UTC

Contents

bigtabulate 2

Index **5**

bigtabulate *Extended Tabular Operations for Both matrix and big.matrix Objects*

Description

This package extends the **bigmemory** package, but the functions may also be used with traditional R `matrix` and `data.frame` objects. The function `bigtabulate` is exposed, but we expect most users will prefer the higher-level functions `bigtable`, `bigtsummary`, and `bigsplit`. Each of these functions provides functionality based on a specified conditional structure. In other words, for every cell of a (possibly multidimensional) contingency table, they provide (or tabulate) some useful conditional behavior (or statistic(s)) of interest. At the most basic level, this provides an extremely fast and memory-efficient alternative to `table` for matrices and data frames.

Usage

```
bigtabulate(
  x,
  ccols,
  breaks = vector("list", length = length(ccols)),
  table = TRUE,
  useNA = "no",
  summary.cols = NULL,
  summary.na.rm = FALSE,
  splitcol = NULL,
  splitret = "list"
)

bigsplit(
  x,
  ccols,
  breaks = vector("list", length = length(ccols)),
  useNA = "no",
  splitcol = NA,
  splitret = "list"
)

bigtable(
  x,
  ccols,
  breaks = vector("list", length = length(ccols)),
  useNA = "no"
)

bigtsummary(
  x,
  ccols,
  breaks = vector("list", length = length(ccols)),
```

```
  useNA = "no",
  cols,
  na.rm = FALSE
)
```

Arguments

x	a big.matrix or a data.frame or a matrix.
ccols	a vector of column indices or names specifying which columns should be used for conditioning (e.g. for building a contingency table or structure for tabulation).
breaks	a vector or list of length(ccols). If a vector, NA indicates that the associated column should be treated like a factor (categorical variable), while an integer value indicates that the range of the associated column should be broken into a specified number of evenly-spaced bins (histogram-like). If a list, NA triggers the factor-like handling, a single number triggers bin-like behavior, while a triplet (min,max,breaks) indicates that the bin-like behavior should be on a restricted range rather than on the range of data for that column. See binit for similar specification of this option.
table	if TRUE, a list of table counts will be returned.
useNA	whether to include extra 'NA' levels in the table.
summary.cols	column(s) for which table summaries will be calculated.
summary.na.rm	if TRUE, NAs are removed from table summary calculations.
splitcol	if NA, the indices which correspond to table-levels are returned. If numeric, the corresponding column values will be returned in a list corresponding to table-levels. If NULL, then there is no splitting at all.
splitret	if "list", the splitcol value is returned as a list. When splitcol is NA, splitret may be "vector". Finally, "sparselist" may be a useful option when the full-blown splitting structure has many unrepresented "cells"; this is like using the drop=TRUE option to split.
cols	with bigtsummary, which column(s) should be conditionally summarized? This (or these) will be passed on as summary.cols.
na.rm	an obvious option for summaries.

Details

This package concentrates on conditional stuctures and calculations, much like table, tapply, and split. The functions are juiced-up versions of the base R functions; they work on both regular R matrices and data frames, but are specialized for use with **bigmemory** and (for more advanced usage) **foreach**. They are particularly fast and memory-efficient. We have found that bigsplit followed by lapply or sapply can be particularly effective, when the subsets produced by the split are of reasonable size. For intensive calculations, subsequent use of foreach can be helpful (think: parallel apply-like behavior).

When x is a matrix or a data.frame, some additional work may be required. For example, a character column of a data.frame will be converted to a factor and then coerced to numeric values (factor level numberings).

The conditional structure is specified via `ccols` and `breaks`. This differs from the design of the base R functions but is at the root of the gains in speed and memory-efficiency. The `breaks` may seem distracting, as most users will simply condition on categorical-like columns. However, it provides the flexibility to "bin" "continuous", column(s) much like a histogram. See `binit` for another example of this type of option, which can be particularly valuable with massive data sets.

A word of caution: if a "continuous" variable is not "binned", it will be treated like a factor and the resulting conditional structure will be large (perhaps immensely so). The function uses left-closed intervals [a,b) for the "binning" behavior, when specified, except in the right-most bin, where the interval is entirely closed.

Finally, `bigsplit` is somewhat more general than `split`. The default behavior (`splitcol=NA`) returns a split of `1:nrow(x)` as a list based on the specified conditional structure. However, it may also return a vector of cell (or category) numbers. And of course it may conduct a split of `x[,splitcol]`.

Value

array-like object(s), each similar to what is returned by `tapply` and the associated R functions.

Examples

```
data(iris)

# First, break up column 2 into 5 groups, and leave column 5 as a
# factor (which it is).  Note that iris is a data.frame, which is
# fine.  A matrix would also be fine.  A big.matrix would also be fine!
bigtable(iris, ccols=c(2, 5), breaks=list(5, NA))

iris[,2] <- round(iris[,2]) # So columns 2 and 5 will be factor-like
                            # for convenience in these examples, below:

ans1 <- bigtable(iris, c(2, 5))
ans1
# Same answer, but with nice factor labels from table(), because
# table() handles factors.  bigtable() uses the numeric factor
# levels only.
table(iris[,2], iris[,5])

# Here, our formulation is simpler than split's, and is faster and
# more memory-efficient:
ans2 <- bigsplit(iris, c(2, 5), splitcol=1)
ans2[1:3]
split(iris[,1], list(col2=factor(iris[,2]), col5=iris[,5]))[1:3]
```

Index

big.matrix, *3*
bigsplit *(*bigtabulate*)*, 2
bigtable *(*bigtabulate*)*, 2
bigtabulate, *2*, 2
bigtsummary *(*bigtabulate*)*, 2
binit, *3*, *4*

data.frame, *3*

factor, *3*

lapply, *3*

matrix, *3*

sapply, *3*
split, *3*

table, *2*, *3*
tapply, *3*, *4*