Package: VBTree (via r-universe)

October 9, 2024

Type Package

Title Vector Binary Tree to Make Your Data Management More Efficient

Version 0.1.1

Date 2024-01-09

Description Vector binary tree provides a new data structure, to make your data visiting and management more efficient. If the data has structured column names, it can read these names and factorize them through specific split pattern, then build the mappings within double list, vector binary tree, array and tensor mutually, through which the batched data processing is achievable easily. The methods of array and tensor are also applicable. Detailed methods are described in Chen Zhang et al. (2020) <doi:10.35566/isdsa2019c8>.

Imports tensorA

Depends R (>= 2.10)

License GPL-3

URL https://github.com/CubicZebra/VBTree

BugReports https://github.com/CubicZebra/VBTree/issues

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

Repository https://cubiczebra.r-universe.dev

RemoteUrl https://github.com/cubiczebra/vbtree

RemoteRef HEAD

RemoteSha 7ce5c2b98874fb513d5ddd62aac41ad586ec5c37

Contents

`VBTree-package` *Vector Binary Tree to Make Your Data Management More Efficient*

Description

Vector binary tree provides a new data structure, to make your data visiting and management more efficient. If the data has structured column names, it can read these names and factorize them through specific split pattern, then build the mappings within double list, vector binary tree, array and tensor mutually, through which the batched data processing is achievable easily. The methods of array and tensor are also applicable. Detailed methods are described in Chen Zhang et al. (2020) <doi:10.35566/isdsa2019c8>.

Details

This package provide an efficient approach to manage data by structurizing the column names. A column name is generally seen as a character object, while if it has a very organized pattern, such as "*-*-*-*" for example (each * mark presents a different condition), it must has a certain mapping relationship to a specific tensor. This package uses two data structure: double list and vector binary tree, to implement the conversion between the character vector and tensor. It affords various inquiry methods, which was mainly drived by vector binary tree, to extract the highly customizable subset from original data.

Author(s)

Chen Zhang [aut, cre, cph] (<https://orcid.org/0009-0007-7689-5030>)

Maintainer: Chen Zhang <chen.zhang_06sept@foxmail.com>

References

Sedgewick, Robert & Wayne, Kevin (2011). Algorithms, 4th Edition.. Addison-Wesley

Prakash, P. K. S. & Rao, Achyutuni Sri Krishna (2016). R Data Structures and Algorithms. Packt Publishing

See Also

to.tensor, pos.tensor.

Examples

```
#View the data to be visited:
summary(datatest)
colnames(datatest)

#Structurize colnames of data into vector binary tree:
dl <- chrvec2dl(colnames(datatest))
vbt <- dl2vbt(dl)
vbt

#Setting subset in different forms, for example the pattern
#"Strain-(900~1100)-(0.01, 1)-0.6" is desired:
subunregdl <- list(c(1), c(1:5), c(2,4), c(1)) # undifined double list
subregdl <- advbtinq(vbt, subunregdl) # regularized double list
subvbt <- dl2vbt(subregdl) # sub vector binary tree
subts <- vbt2ts(subvbt) # tensor
subarr <- vbt2arr(subvbt) # array
subchrvec <- as.vector(subarr) # character vector

#Visit the data through different methods:
datavisit(datatest, subunregdl) # by handmade double list
datavisit(datatest, subregdl) # by defined double list
datavisit(datatest, subvbt) # by vector binary tree
datavisit(datatest, subts) # by tensor
datavisit(datatest, subarr) # by array
datavisit(datatest, subchrvec) # by character vector
```

---

advbtinq *Using double list to visit vector binary tree*

Description

Advanced visiting for the vector binary tree. Return a double list by specific assigment determined by the argument `inq`.

Usage

```
advbtinq(x, inq)
```

Arguments

`x`	The vector binary tree to be visited. Traversal is acheivable through invalid assignment in desired layer.
`inq`	An integer double list to determine the location to be visited. The length of `inq` should be the same as the layers of visited vector binary tree, while all elements in vector in each layer of `inq` should not over the intrinsic length of visited vector binary tree layer, otherwise all elements will be returned in this layer.

Value

Return a double list according to the argument `inq`.

See Also

`vbtinq`, `vbtsub`, `advbtsub`.

Examples

```
#Make vector binary tree:
colnamevbt <- dl2vbt(chrvec2dl(colnames(datatest)))

#Visit by specific assignment:
visit <- list(c(2), c(3:6), c(2,4), 1)
advbtinq(colnamevbt, visit)

#Traversal of the second layers:
visit <- list(c(2), colnamevbt$dims[2]+1, c(2,4), 1)
advbtinq(colnamevbt, visit)

#Invalid assignments in 1st and 3rd layers:
visit <- list(c(3), c(3:6), c(5), 1)
advbtinq(colnamevbt, visit)
```

advbtsub *Using double list to generate sub tree from vector binary tree*

Description

Advanced visiting for the vector binary tree. Generating a sub tree from visited vector binary tree, through specific assigment determined by the argument `inq`.

Usage

```
advbtsub(x, inq)
```

Arguments

`x`	The vector binary tree to be visited. Traversal is acheivable through invalid assignment in desired layers.
`inq`	An integer double list to determine the visiting location. The length of `inq` should be the same as the layers of visited vector binary tree. If any assign element in specificed layer exceeds its intrinsic length of visited vector binary tree layer, all elements will be returned in this layer.

Value

Return a sub tree from visited vector binary tree, according to the argument `inq`.

See Also

`vbtinq`, `vbtsub`, `advbtinq`.

Examples

```
#Make vector binary tree:
colnamevbt <- dl2vbt(chrvec2dl(colnames(datatest)))

#Visit by specific assignment:
visit <- list(c(2), c(3:6), c(2,4), 1)
advbtsub(colnamevbt, visit)

#Traversal of the second layers:
visit <- list(c(2), colnamevbt$dims[2]+1, c(2,4), 1)
advbtsub(colnamevbt, visit)

#Invalid assignments in 1st and 3rd layers:
visit <- list(c(3), c(3:6), c(5), 1)
advbtsub(colnamevbt, visit)
```

arr2dl *Convert a structured character array to double list*

Description

Convert a structured character array to a double list. All character elements in array will be splited by a specific pattern then sorted intrinsically in each layer of the double list.

Usage

```
arr2dl(x, ...)
```

Arguments

x	A structured character array to be converted.
...	Argument in chrvec2dl to control split pattern.

Value

Return a double list based on the input array.

See Also

arr2vbt, chrvec2dl.

Examples

```
#Write the column names of datatest into a array:
arr <- dl2arr(chrvec2dl(colnames(datatest)))

#Recover the double list from character array:
arr2dl(arr)
```

arr2vbt *Convert a structured character array to double list*

Description

Convert a structured character array to a vector binary tree. All character elements in array will be splited by a specific pattern then sorted intrinsically in each layer of the vector binary tree.

Usage

```
arr2vbt(x, ...)
```

Arguments

x	A structured character array to be converted.
...	Argument in chrvec2dl to control split pattern.

Value

Return a vector binary tree based on the input array.

See Also

arr2dl, chrvec2dl.

Examples

```
#Write the column names of datatest into a array:
arr <- dl2arr(chrvec2dl(colnames(datatest)))

#Recover the vector binary tree from character array:
arr2vbt(arr)
```

---

chrvec2dl *Convert character vector to a double list*

Description

Structurize a character vector to a double list. Layers in the double list will be determined by the given pattern.

Usage

```
chrvec2dl(x, splt = "-")
```

Arguments

x	a character vector to be converted.
splt	a string pattern to make defination for spliting each layer of double list.

Value

return a character double list splited by defined pattern, the default pattern is "-".

Examples

```
#example using default dataset:

charvector <- colnames(datatest)
chrvec2dl(charvector, "-")
```

datatest *A test data structurized column names.*

Description

A test data with 56 different columns.

Usage

```
data("datatest")
```

Details

A test data structurized column names, with two data type "Strain" and "Stress", 7 different temperatures, 4 kinds strain rates and one level of compression rate.

Examples

```
datatest
```

datavisit *Extract subset of data using different methods*

Description

Extract the subset of data by column names using tensor, array, double list, integer vector, or vector binary tree.

Usage

```
datavisit(data, inq)
```

Arguments

`data`	A data.frame with structured column names.
`inq`	An argument to determine the subset to be extracted by column names. A tensor, array, double list, integer vector and vector binary tree is available format of `inq`.

Value

Return a list which contains the item index, column name, column coordinate and the data in corresponding column for each element contained in the assignment of `inq`.

See Also

`vbtinq`, `advbtinq`, `trvseleinq`, `trvsidxinq`, `trvssubinq`.

Examples

```
#View the data to be visited:
summary(datatest)
colnames(datatest)

#Structurize colnames of data into vector binary tree:
dl <- chrvec2dl(colnames(datatest))
vbt <- dl2vbt(dl)
vbt

#Setting subset in different forms, for example the pattern
#"Strain-(900~1100)-(0.01, 1)-0.6" is desired:
subunregdl <- list(c(1), c(1:5), c(2,4), c(1)) # undefined double list
subregdl <- advbtinq(vbt, subunregdl) # regularized double list
subvbt <- dl2vbt(subregdl) # sub vector binary tree
subts <- vbt2ts(subvbt) # tensor
subarr <- vbt2arr(subvbt) # array
subchrvec <- as.vector(subarr) # character vector

#Visit the data through different methods:
datavisit(datatest, subunregdl) # by handmade double list
datavisit(datatest, subregdl) # by defined double list
datavisit(datatest, subvbt) # by vector binary tree
datavisit(datatest, subts) # by tensor
datavisit(datatest, subarr) # by array
datavisit(datatest, subchrvec) # by character vector
```

---

dl2arr *Convert a double list to array*

Description

Convert a double list to an array. The pure numeric layers will be sorted intrinsically then all elements will be bound in certain order as one character element, and filled into the proper location in the array.

Usage

```
dl2arr(x)
```

Arguments

x	A double list to be converted.

Value

Return an array filled with the binding character elements.

See Also

dl2vbt, dl2ts.

Examples

```
#Make column names of datatest into double list:
dl <- chrvec2dl(colnames(datatest), "-")

#Convert the double list to a tensor:
dl2arr(dl)
```

---

dl2ts *Convert a double list to tensor*

Description

Convert a double list to a tensor. The pure numeric layers will be sorted intrinsically then all elements will be bound in certain order as one character element, and filled into the proper location in the tensor.

Usage

```
dl2ts(x)
```

Arguments

x	A double list to be converted.

Value

Return a tensor filled with the binding character elements.

See Also

dl2vbt, dl2arr.

Examples

```
#Make column names of datatest into double list:
dl <- chrvec2dl(colnames(datatest), "-")

#Convert the double list to a tensor:
dl2ts(dl)
```

dl2vbt *Convert a double list to vector binary tree*

Description

Convert a double list to vector binary tree. The pure numeric layers will be sorted intrinsically then all elements be exported in character form.

Usage

```
dl2vbt(x, regularize = TRUE, splt = "-")
```

Arguments

x	A double list to be converted.
regularize	A boolean value to control the treatment of empty layers of double listed to be converted. The default value TRUE will fill the empty layer by mark "*". The default value is recommanded.
splt	A string pattern to split the binding elements in each layer if the sub-constructure exists. The default pattern uses "-".

Value

Return a vector binary tree.

See Also

vbtinq, vbtsub, advbtinq, advbtsub, trvssubinq, dl2ts, dl2arr.

Examples

```
#Structurize the column names of datatest:
colname <- colnames(datatest)
colnamedl <- chrvec2dl(colname, "-")
colnamevbt <- dl2vbt(colnamedl)

#Simple data cleaning for sub-constructure existing double list;
#Make unregulated double list:
unregdl <- list(c("7", 2, 10), c("chr", "5"), c(),
c("var2", "var1", "var3"), c("M-8-9", "3-2"), c("6-3", "2-7"))
regvbt <- dl2vbt(unregdl)
regvbt2 <- dl2vbt(unregdl, FALSE) # not recommended
```

hello *Welcome message*

Description

Welcome message

Usage

```
hello()
```

Value

exit code of zero

Examples

```
hello()
```

trvs *Make traversal from vector binary tree*

Description

Generating a table of traversal from given vector binary tree, in order to construct correct mapping relationships within double list, vector binary tree, array and tensor.

Usage

```
trvs(x)
```

Arguments

x	A vector binary tree.

Value

Return a traversal table from the given vector binary tree.

Examples

```
#Make vector binary tree:
colnamevbt <- dl2vbt(chrvec2dl(colnames(datatest)))

#Construct traversal table:
trvs(colnamevbt)
```

trvseleinq *Using character element to visit the traversal table*

Description

Visit the traversal table generated from a vector binary tree through the character element determined by the argument inq, and return an inquiry result containing its numeric item index, the character pattern and its corresponding coordinate.

Usage

```
trvseleinq(trvs, inq)
```

Arguments

trvs	The traversal table to be visited, which should be generated from the vector binary tree by the function trvs().
inq	A desired character element to match the traversal table.

Value

Return an inquiry result with a numeric item index, a character pattern and its coordinate in form of integer vector.

Examples

```
#Make traversal table:
trav <- trvs(dl2vbt(chrvec2dl(colnames(datatest))))

#Visit specific element by character pattern:
trvseleinq(trav,"Strain-1100-0.001-0.6")
```

trvsidxinq *Using vector to visit the traversal table*

Description

Visit the traversal table generated from a vector binary tree through the coordinate determined by the argument inq, and return an inquiry result containing its numeric item index, its corresponding character pattern and the coordinate.

Usage

```
trvsidxinq(trvs, inq)
```

Arguments

trvs	The traversal table to be visited, which should be generated from the vector binary tree by the function trvs().
inq	An integer vector to assign the coordinate corresponding to the element to be visited.

Value

Return an inquiry result with a numeric item index, a character pattern and its coordinate in form of integer vector.

Examples

```
#Make traversal table:
trav <- trvs(dl2vbt(chrvec2dl(colnames(datatest))))

#Visit specific element by its coordinate:
trvsidxinq(trav,c(1,2,3,1))
```

---

trvssubinq *Using sub vector binary tree to visit the traversal table*

Description

Visit the traversal table generated from a vector binary tree through the sub vector binary tree determined by the argument inq, and return an inquiry list containing the numeric index, the character pattern and the corresponding coordinate for each item.

Usage

```
trvssubinq(trvs, inq)
```

Arguments

trvs	The traversal table to be visited, which should be generated from the vector binary tree by the function trvs().
inq	A sub tree generated from the original vector binary tree, to determine the subset of elements to be visited.

Value

Return a list containing the numeric index, the character pattern and the corresponding coordinate for each item.

See Also

vbtsub, advbtsub.

Examples

```
#Make original vector binary tree and its traversal table:
vbt <- dl2vbt(chrvec2dl(colnames(datatest)))
trav <- trvs(vbt)

#Visit all elements defined by sub vector binary tree:
#example 1: visit all "Stress-*-*-*" patterns;
#make sub vector binary tree through vbtsub() then execute inquiry:
subvbt <- vbtsub(vbt, c(2,-1,-1,-1))
trvssubinq(trav, subvbt)

#example 2: visit all "Strain-("950", "1050")-("0.001", "0.1")-*" patterns;
#make sub vector binary tree through advbtsub() then execute inquiry:
subvbt <- advbtsub(vbt, list(1, c(2,4), c(1,3), 1))
trvssubinq(trav, subvbt)
```

---

ts2dl *Convert a structured character tensor to double list*

Description

Convert a structured character tensor to a double list. All character elements in tensor will be splited by a specific pattern then sorted intrinsically in each layer of the double list.

Usage

```
ts2dl(x, ...)
```

Arguments

x	A structured character tensor to be converted.
...	Argument in chrvec2dl to control split pattern.

Value

Return a double list based on the input tensor.

See Also

ts2vbt, chrvec2dl.

Examples

```
#Write the column names of datatest into a tensor:
ts <- dl2ts(chrvec2dl(colnames(datatest)))

#Recover the double list from character tensor:
ts2dl(ts)
```

ts2vbt *Convert a structured character tensor to double list*

Description

Convert a structured character tensor to a vector binary tree. All character elements in tensor will be splited by a specific pattern then sorted intrinsically in each layer of the vector binary tree.

Usage

```
ts2vbt(x, ...)
```

Arguments

x	A structured character tensor to be converted.
...	Argument in chrvec2dl to control split pattern.

Value

Return a vector binary tree based on the input tensor.

See Also

ts2dl, chrvec2dl.

Examples

```
#Write the column names of datatest into a tensor:
ts <- dl2ts(chrvec2dl(colnames(datatest)))

#Recover the vector binary tree from character tensor:
ts2vbt(ts)
```

vbt2arr *Convert a vector binary tree to array*

Description

Convert a vector binary tree to an array. The pure numeric layers will be sorted intrinsically then all elements will be bound in certain order as one character element, and filled into the proper location in the array.

Usage

```
vbt2arr(x)
```

Arguments

x	A vector binary tree to be converted.

Value

Return an array filled with the binding character elements.

See Also

vbt2dl, vbt2ts.

Examples

```
#Make column names of datatest into vector binary tree:
vbt <- dl2vbt(chrvec2dl(colnames(datatest), "-"))

#Convert the vector binary tree to an array:
vbt2arr(vbt)
```

---

vbt2dl *Convert a vector binary tree to double list*

Description

Recover a vector binary tree to double list for easy visualization. Empty layers in vector binary tree will be marked by the symbol "*" as default.

Usage

```
vbt2dl(x)
```

Arguments

x	A vector binary tree to be converted.

Value

Return a double list based on input vector binary tree.

See Also

vbtinq, vbtsub, advbtinq, advbtsub, trvssubinq, vbt2ts, vbt2arr.

Examples

```
#Recover vector binary tree to a double list for easy visualization:
vbt <- dl2vbt(chrvec2dl(colnames(datatest))) #make vector binary tree
vbt2dl(vbt)
```

vbt2ts *Convert a vector binary tree to tensor*

Description

Convert a vector binary tree to a tensor. The pure numeric layers will be sorted intrinsically then all elements will be bound in certain order as one character element, and filled into the proper location in the tensor.

Usage

```
vbt2ts(x)
```

Arguments

x	A vector binary tree to be converted.

Value

Return a tensor filled with the binding character elements.

See Also

vbt2dl, vbt2arr.

Examples

```
#Make column names of datatest into vector binary tree:
vbt <- dl2vbt(chrvec2dl(colnames(datatest), "-"))

#Convert the vector binary tree to a tensor:
vbt2ts(vbt)
```

vbtinq *Using vector to visit vector binary tree*

Description

Visit the vector binary tree and return a double list through specific assigment determined by the argument inq.

Usage

```
vbtinq(x, inq)
```

Arguments

x	The vector binary tree to be visited. Traversal is available by setting -1 in desired layer.
inq	An integer vector to determine desired location. The length of inq should be the same as the layers of visited vector binary tree. If any assignment in specificed layer exceeds its intrinsic length of visited vector binary tree layer, all elements will be returned in this layer.

Value

Return a double list according to the argument inq.

See Also

vbtsub, advbtinq, advbtsub.

Examples

```
#Make vector binary tree:
colnamevbt <- dl2vbt(chrvec2dl(colnames(datatest)))

#Visit by specific assignment:
vbtinq(colnamevbt, c(2, 3, 1, 1))

#Traversal of the second layers:
vbtinq(colnamevbt, c(2, -1, 1, 1))

#Invalid assignments in 1st and 3rd layers:
vbtinq(colnamevbt, c(4, 3, 7, 1))
```

---

vbtsub *Using vector to generate sub tree from vector binary tree*

Description

Visit the vector binary tree and generate a sub tree from visited vector binary tree, through specific assigment determined by the argument inq.

Usage

```
vbtsub(x, inq)
```

Arguments

x	The vector binary tree to be visited. Traversal is available by setting -1 in desired layer.
inq	An integer vector to determine the visiting location. The length of inq should be the same as the layers of visited vector binary tree. If any assignment in specificed layer exceeds its intrinsic length of visited vector binary tree layer, all elements will be returned in this layer.

Value

Return a sub tree from visited vector binary tree, according to the argument inq.

See Also

vbtinq, advbtinq, advbtsub.

Examples

```
#Make vector binary tree:
colnamevbt <- dl2vbt(chrvec2dl(colnames(datatest)))

#Generating sub tree by specific assignment:
vbtsub(colnamevbt, c(2, 3, 1, 1))

#Generating sub tree with traversal in the second layers:
vbtsub(colnamevbt, c(2, -1, 1, 1))

#Generating sub tree with invalid assignments in 1st and 3rd layers:
vbtsub(colnamevbt, c(4, 3, 7, 1))
```

Index