

Package ‘SSBtools’

January 21, 2019

Type Package

Title Statistics Norway's Miscellaneous Tools

Version 0.4.0

Date 2019-01-21

Depends R (>= 3.2.0), Matrix

Imports stringr

Description Functions used by other packages from Statistics Norway are gathered. General data manipulation functions, and functions for hierarchical computations are included. The hierarchy specification functions are useful within statistical disclosure control.

License Apache License 2.0 | file LICENSE

URL <https://github.com/statisticsnorway/SSBtools>

BugReports <https://github.com/statisticsnorway/SSBtools/issues>

LazyData TRUE

RoxygenNote 6.1.1

Encoding UTF-8

Suggests testthat

NeedsCompilation no

Author Øyvind Langsrud [aut, cre],
Bjørn-Helge Mevik [cph]

Maintainer Øyvind Langsrud <oyl@ssb.no>

Repository CRAN

Date/Publication 2019-01-21 14:50:08 UTC

R topics documented:

AddLeadingZeros	2
AutoHierarchies	3
AutoSplit	5
CbindIdMatch	6

DimList2Hierarchy	7
DimList2Hrc	8
DummyHierarchy	9
FactorLevCorr	10
FindCommonCells	11
FindDimLists	12
FindTableGroup	13
FormulaSums	14
HierarchicalGroups	15
HierarchicalWildcardGlobbing	16
Hierarchies2ModelMatrix	18
Hierarchy2Formula	20
HierarchyCompute	21
HierarchyFix	23
MakeHierFormula	24
Match	25
matlabColon	26
Number	27
RbindAll	28
RowGroups	29
SSBtoolsData	30
Stack	31
UniqueIntegerSortMatrix	32
Unstack	32
WildcardGlobbing	33
WildcardGlobbingVector	34
Index	36

AddLeadingZeros	<i>Add leading zeros to numbers while preserving other text</i>
-----------------	---

Description

This function is created to fix problems caused by a serious bug in Excel. Editing csv files in that program causes leading zeros to disappear.

Usage

```
AddLeadingZeros(codes, places, warningText = NULL, viaFactor = TRUE,
  nWarning = 6, removeLeadingTrailingWhitespace = TRUE)
```

Arguments

codes	Character vector
places	Number of places for positive numbers. Minus sign is extra
warningText	When non-NULL, warning will be produced
viaFactor	When TRUE, the algorithm uses factor coding internally.
nWarning	Number of elements to be written before ... in warnings.
removeLeadingTrailingWhitespace	Remove leading and trailing whitespace

Value

Character vector

Author(s)

Øyvind Langsrud

Examples

```
AddLeadingZeros(c("1", "ABC", "12345", " 23", "-8", "45 ", " -9", " Agent ", "007",
                  "7 James Bond "), 10)
AddLeadingZeros(c("1", "ABC", "12345", " 23", "-8", "45 ", " -9", " Agent ", "007",
                  "7 James Bond "), 4)
AddLeadingZeros(c("1", "ABC", "12345", " 23", "-8", "45 ", " -9", " Agent ", "007",
                  "7 James Bond "), 4, removeLeadingTrailingWhitespace = FALSE)
AddLeadingZeros(c("1", "ABC", "12345", " 23", "-8", "45 ", " -9", " Agent ", "007",
                  "7 James Bond "), 4, warningText = "string changes")
AddLeadingZeros(c("1", "ABC", "12345", " 23", "-8", "45 ", " -9", " Agent ", "007",
                  "7 James Bond "), 4, warningText = "", nWarning = 2)
```

AutoHierarchies

Ensure standardized coding of hierarchies

Description

Automatic convert list of hierarchies coded in different ways to standardized to-from coding

Usage

```
AutoHierarchies(hierarchies, data = NULL, total = "Total",
                hierarchyVarNames = c(mapsFrom = "mapsFrom", mapsTo = "mapsTo", sign =
                "sign", level = "level"), combineHierarchies = TRUE,
                unionComplement = FALSE)
```

```
FindHierarchies(data, total = "Total")
```

Arguments

hierarchies	List of hierarchies
data	Matrix or data frame with data containing codes of relevant variables
total	Within AutoHierarchies: Vector of total codes (possibly recycled) used when running Hrc2DimList .
hierarchyVarNames	Variable names in the hierarchy tables as in HierarchyFix
combineHierarchies	Whether to combine several hierarchies for same variable into a single hierarchy
unionComplement	Logical vector as in Hierarchies2ModelMatrix . The parameter is only in use when hierarchies are combined.

Details

Input can be to-from coded hierarchies, hierarchies/dimList as in `sdcTable`, TauArgus coded hierarchies or formulas. Automatic coding from data is also supported. Output is on a from ready for input to [HierarchyCompute](#). `FindHierarchies` wraps [FindDimLists](#) and `AutoHierarchies` into a single function. A single string as hierarchy input is assumed to be a total code. Then, the hierarchy is created as a simple hierarchy where all codes in data sum up to this total. For consistence with `HierarchyCompute`, the codes "rowFactor" and "colFactor" are unchanged. An empty string is recoded to "rowFactor".

Value

List of hierarchies

Author(s)

Øyvind Langsrud

See Also

[DimList2Hierarchy](#), [Hierarchy2Formula](#).

Examples

```
# First, create different types of input
z <- SSBtoolsData("sprt_emp_withEU")
yearFormula <- c("y_14 = 2014", "y_15_16 = y_all - y_14", "y_all = 2014 + 2015 + 2016")
yearHier <- Formula2Hierarchy(yearFormula)
geoDimList <- FindDimLists(z[, c("geo", "eu")], total = "Europe")[[1]]
geoDimList2 <- FindDimLists(z[, c("geo", "eu")])[[1]]
geoHrc <- DimList2Hrc(geoDimList)
ageHier <- SSBtoolsData("sprt_emp_ageHier")

h1 <- AutoHierarchies(list(age = ageHier, geo = geoDimList, year = yearFormula))
h2 <- AutoHierarchies(list(age = "Y15-64", geo = geoHrc, year = yearHier), data = z,
  total = "Europe")
```

```

h3 <- AutoHierarchies(list(age = "Total", geo = geoDimList2, year = "Total"), data = z)
h4 <- FindHierarchies(z[, c(1, 2, 3, 5)])
h5 <- AutoHierarchies(list(age = "Total", geo = "", year = "colFactor"), data = z)
identical(h1, h2)
identical(h3, h4)

FindHierarchies(z[, c("geo", "eu", "age")])

```

AutoSplit

Creating variables by splitting the elements of a character vector without needing a split string

Description

Creating variables by splitting the elements of a character vector without needing a split string

Usage

```

AutoSplit(s, split = NULL, border = "_", revBorder = FALSE,
  noSplit = FALSE, varNames = paste("var", 1:100, sep = ""),
  tryReverse = TRUE)

```

Arguments

<code>s</code>	The character vector
<code>split</code>	Split string. When NULL (default), automatic splitting without a split string.
<code>border</code>	A split character or an integer (move split) to be used when the exact split position is not unique.
<code>revBorder</code>	When border is integer the split position is moved from the other side.
<code>noSplit</code>	No splitting when TRUE.
<code>varNames</code>	Variable names of the created variables (too many is ok)
<code>tryReverse</code>	When TRUE, the automatic method tries to find more variables by splitting from reversed strings.

Value

A data frame with `s` as row names.

Author(s)

Øyvind Langsrud

Examples

```
s <- c("A12-3-A-x", "A12-3-B-x", "B12-3-A-x", "B12-3-B-x",
      "A12-3-A-y", "A12-3-B-y", "B12-3-A-y", "B12-3-B-y")
AutoSplit(s)
AutoSplit(s, border="-")
AutoSplit(s, split="-")
AutoSplit(s, border=1)
AutoSplit(s, border=2)
AutoSplit(s, border=2, revBorder=TRUE)
AutoSplit(s, noSplit=TRUE)
AutoSplit(s, varNames=c("A", "B", "C", "D"))
```

CbindIdMatch

Combine several data frames by using id variables to match rows

Description

Combine several data frames by using id variables to match rows

Usage

```
CbindIdMatch(..., addName = names(x), sep = "_", idNames = sapply(x,
  function(x) names(x)[1]), idNames1 = idNames, addLast = FALSE)
```

Arguments

...	Several data frames as several input parameters or a list of data frames
addName	NULL or vector of strings used to name columns according to origin frame
sep	A character string to separate when addName apply
idNames	Names of a id variable within each data frame
idNames1	Names of variables in first data frame that correspond to the id variable within each data frame
addLast	When TRUE addName will be at end

Details

The first data frame is the basis and the other frames will be matched by using id-variables. The default id-variables are the first variable in each frame. Corresponding variables with the same name in first frame is assumed. An id-variable is not needed if the number of rows is one or the same as the first frame. Then the element of idNames can be set to a string with zero length.

Value

A single data frame

Author(s)

Øyvind Langsrud

See Also[RbindAll](#) (same example data)**Examples**

```

zA=data.frame(idA=1:10, idB=rep(10*(1:5), 2),
              idC=rep(c(100, 200), 5), idC2=c(100, rep(200, 9)), idC3=rep(100, 10),
              idD=99, x=round(rnorm(10), 3), xA=round(runif(10), 2))
zB=data.frame(idB=10*(1:5), x=round(rnorm(5), 3), xB=round(runif(5), 2))
zC=data.frame(idC=c(100, 200), x=round(rnorm(2), 3), xC=round(runif(2), 2))
zD=data.frame(idD=99, x=round(rnorm(1), 3), xD=round(runif(1), 2))
CbindIdMatch(zA, zB, zC, zD)
CbindIdMatch(a=zA, b=zB, c=zC, d=zD, idNames=c("", "idB", "idC", ""))
CbindIdMatch(a=zA, b=zB, c=zC, d=zD, idNames1=c("", "idB", "idC2", ""))
CbindIdMatch(a=zA, b=zB, c=zC, d=zD, idNames1=c("", "idB", "idC3", ""))
CbindIdMatch(zA, zB, zC, zD, addName=c("", "bbb", "ccc", "ddd"), sep=".", addLast=TRUE)
try(CbindIdMatch(X=zA, Y=zA[, 4:5], Z=zC, idNames=NULL)) # Error
CbindIdMatch(X=zA, Y=zA[, 4:5], Z=zD, idNames=NULL)      # Ok since equal NROW or NROW==1
CbindIdMatch(list(a=zA, b=zB, c=zC, d=zD))              # List is alternative input

```

DimList2Hierarchy

*DimList2Hierarchy***Description**From hierarchy/dimList as in `sdcTable` to to-from coded hierarchy**Usage**`DimList2Hierarchy(x)`**Arguments**x An element of a `dimList` as in `sdcTable`**Value**

Data frame with to-from coded hierarchy

Author(s)

Øyvind Langsrud

Examples

```
# First generate a dimList element
x <- FindDimLists(SSBtoolsData("sprt_emp_withEU")[, c("geo", "eu")], , total = "Europe")[[1]]

DimList2Hierarchy(x)
```

DimList2Hrc

DimList2Hrc/Hrc2DimList

Description

Conversion between hierarchies/dimList as in sdcTable and TauArgus coded hierarchies

Usage

```
DimList2Hrc(dimList)

Hrc2DimList(hrc, total = "Total")
```

Arguments

dimList	List of data frames according to the specifications in sdcTable
hrc	List of character vectors
total	String used to name totals.

Value

See Arguments

Author(s)

Øyvind Langsrud

Examples

```
# First generate dimList
dimList <- FindDimLists(SSBtoolsData("sprt_emp_withEU")[, c("geo", "eu", "age")])
hrc <- DimList2Hrc(dimList)
dimList2 <- Hrc2DimList(hrc)
identical(dimList, dimList2)
```

DummyHierarchy	<i>Converting hierarchy specifications to a (signed) dummy matrix</i>
----------------	---

Description

A matrix for mapping input codes (columns) to output codes (rows) are created.

Usage

```
DummyHierarchy(mapsFrom, mapsTo, sign, level, mapsInput = NULL,
  inputInOut = FALSE, keepCodes = mapsFrom[integer(0)],
  unionComplement = FALSE)
```

Arguments

mapsFrom	Character vector from hierarchy table
mapsTo	Character vector from hierarchy table
sign	Numeric vector of either 1 or -1 from hierarchy table
level	Numeric vector from hierarchy table
mapsInput	All codes in mapsTo not in mapsFrom (created automatically when NULL) and possibly other codes in input data.
inputInOut	When FALSE all output rows represent codes in mapsTo
keepCodes	To prevent some codes to be removed when inputInOut = TRUE
unionComplement	When TRUE, sign means union and complement instead of addition or subtraction (see note)

Details

The elements of the matrix specify how columns contribute to rows.

Value

A sparse matrix with row and column and names

Note

With unionComplement = FALSE (default), the sign of each mapping specifies the contribution as addition or subtraction. Thus, values above one and negative values in output can occur. With unionComplement = TRUE, positive is treated as union and negative as complement. Then 0 and 1 are the only possible elements in the output matrix.

Author(s)

Øyvind Langsrud

Examples

```
# A hierarchy table
h <- SSBtoolsData("FIFA2018ABCD")

DummyHierarchy(h$mapsFrom, h$mapsTo, h$sign, h$level)
DummyHierarchy(h$mapsFrom, h$mapsTo, h$sign, h$level, inputInOut = TRUE)
DummyHierarchy(h$mapsFrom, h$mapsTo, h$sign, h$level, keepCodes = c("Portugal", "Spain"))

# Extend the hierarchy table to illustrate the effect of unionComplement
h2 <- rbind(data.frame(mapsFrom = c("EU", "Schengen"), mapsTo = "EUandSchengen",
                      sign = 1, level = 3), h)

DummyHierarchy(h2$mapsFrom, h2$mapsTo, h2$sign, h2$level)
DummyHierarchy(h2$mapsFrom, h2$mapsTo, h2$sign, h2$level, unionComplement = TRUE)

#' # Extend mapsInput - leading to zero columns.
DummyHierarchy(h$mapsFrom, h$mapsTo, h$sign, h$level,
              mapsInput = c(h$mapsFrom[!(h$mapsFrom %in% h$mapsTo)], "Norway", "Finland"))
```

FactorLevCorr

*Factor level correlation***Description**

A sort of correlation matrix useful to detect (hierarchical) relationships between the levels of factor variables.

Usage

```
FactorLevCorr(x)
```

Arguments

x Input matrix or data frame containing the variables

Value

Output is a sort of correlation matrix.

Here we refer to n_i as the number of present levels of variable i (the number of unique elements) and we refer to n_{ij} as the number of present levels obtained by crossing variable i and variable j (the number unique rows of $x[,c(i,j)]$).

The diagonal elements of the output matrix contains the number of present levels of each variable ($=n_i$).

The absolute values of off-diagonal elements:

0 when $n_{ij} = n_i * n_j$
 1 when $n_{ij} = \max(n_i, n_j)$

Other values Computed as $(n_i * n_j - n_{ij}) / (n_i * n_j - \max(n_i, n_j))$

So 0 means that all possible level combinations exist in the data and 1 means that the two variables are hierarchically related.

The sign of off-diagonal elements:

positive when $n_i < n_j$
 negative when $n_i > n_j$

In cases where $n_i = n_j$ elements will be positive above the diagonal and negative below.

Author(s)

Øyvind Langsrud

Examples

```
x <- rep(c("A", "B", "C"), 3)
y <- rep(c(11, 22, 11), 3)
z <- c(1, 1, 1, 2, 2, 2, 3, 3, 3)
zy <- paste(z, y, sep="")
m <- cbind(x, y, z, zy)
FactorLevCorr(m)
```

FindCommonCells *Finding commonCells*

Description

Finding lists defining common cells as needed for the input parameter commonCells to the function protectLinkedTables in package sdcTable. The function handles two tables based on the same main variables but possibly different aggregating variables.

Usage

```
FindCommonCells(dimList1, dimList2)
```

Arguments

dimList1 As input parameter dimList to the function makeProblem in package sdcTable.
 dimList2 Another dimList with the same names and using the same level names.

Value

Output is a list according to the specifications in sdcTable.

Author(s)

Øyvind Langsrud

Examples

```

x <- rep(c('A','B','C'),3)
y <- rep(c(11,22,11),3)
z <- c(1,1,1,2,2,2,3,3,3)
zy <- paste(z,y,sep='')
m <- cbind(x,y,z,zy)
fg <- FindTableGroup(m,findLinked=TRUE)
dimLists <- FindDimLists(m,fg$groupVarInd)
# Using table1 and table2 in this example cause error,
# but in other cases this may work well
try(FindCommonCells(dimLists[fg$table$table1],dimLists[fg$table$table2]))
FindCommonCells(dimLists[c(1,2)],dimLists[c(1,3)])

```

FindDimLists

*Finding dimList***Description**

Finding lists of level-hierarchy as needed for the input parameter dimList to the function makeProblem in package sdcTable

Usage

```

FindDimLists(x, groupVarInd = HierarchicalGroups(x = x),
  addName = FALSE, sep = ".", xReturn = FALSE, total = "Total")

```

Arguments

x	Matrix or data frame containing the variables (micro data or cell counts data).
groupVarInd	List of vectors of indices defining the hierarchical variable groups.
addName	When TRUE the variable name is added to the level names, except for variables with most levels.
sep	A character string to separate when addName apply.
xReturn	When TRUE x is also in output, possibly changed according to addName.
total	String used to name totals.

Value

Output is a list according to the specifications in sdcTable. When xReturn is TRUE output has an extra list level and x is the first element.

Author(s)

Øyvind Langsrud

Examples

```
x <- rep(c('A','B','C'),3)
y <- rep(c(11,22,11),3)
z <- c(1,1,1,2,2,2,3,3,3)
zy <- paste(z,y,sep='')
m <- cbind(x,y,z,zy)
FindDimLists(m)
```

FindTableGroup	<i>Finding table(s) of hierarchical variable groups</i>
----------------	---

Description

A single table or two linked tables are found

Usage

```
FindTableGroup(x = NULL, findLinked = FALSE, mainName = TRUE,
  fCorr = FactorLevCorr(x), CheckHandling = warning)
```

Arguments

x	Matrix or data frame containing the variables
findLinked	When TRUE, two linked tables can be in output
mainName	When TRUE the groupVarInd output is named according to first variable in group.
fCorr	When non-null x is not needed as input.
CheckHandling	Function (warning or stop) to be used in problematic situations.

Value

Output is a list with items

groupVarInd	List defining the hierarchical variable groups. First variable has most levels.
table	List containing one or two tables. These tables are coded as indices referring to elements of groupVarInd.

Author(s)

Øyvind Langsrud

Examples

```
x <- rep(c('A','B','C'),3)
y <- rep(c(11,22,11),3)
z <- c(1,1,1,2,2,2,3,3,3)
zy <- paste(z,y,sep='')
m <- cbind(x,y,z,zy)
FindTableGroup(m)
FindTableGroup(m,findLinked=TRUE)
```

 FormulaSums

Sums (aggregates) and/or sparse model matrix with possible cross table

Description

By default this function return sums if the formula contains a response part and a model matrix otherwise

Usage

```
FormulaSums(data, formula, makeNames = TRUE, crossTable = FALSE,
  total = "Total", printInc = TRUE, dropResponse = FALSE,
  makeModelMatrix = NULL, sep = "-", sepCross = ":")
```

Arguments

data	data frame
formula	A model formula
makeNames	Column/row names made when TRUE
crossTable	Cross table in output when TRUE
total	String used to name totals
printInc	Printing "..." to console when TRUE
dropResponse	When TRUE response part of formula ignored.
makeModelMatrix	Make model matrix when TRUE. NULL means automatic.
sep	String to separate when creating column names
sepCross	String to separate when creating column names involving crossing

Details

The model matrix is constructed by calling `fac2sparse()` repeatedly. The sums are computed by calling `aggregate()` repeatedly. Hierarchical variables handled when constructing cross table. Column names constructed from the cross table.

Value

A matrix of sums, a sparse model matrix or a list of three elements (model matrix, cross table and sums).

Author(s)

Øyvind Langsrud

Examples

```
x <- SSBtoolsData("sprt_emp_withEU")

FormulaSums(x, ths_per ~ year*geo + year*eu)
FormulaSums(x, ~ year*age*eu)
FormulaSums(x, ths_per ~ year*age*geo + year*age*eu, crossTable = TRUE, makeModelMatrix = TRUE)
FormulaSums(x, ths_per ~ year:age:geo -1)
```

HierarchicalGroups *Finding hierarchical variable groups*

Description

According to the (factor) levels of the variables

Usage

```
HierarchicalGroups(x = NULL, mainName = TRUE, eachName = FALSE,
  fCorr = FactorLevCorr(x))
```

Arguments

x	Matrix or data frame containing the variables
mainName	When TRUE output list is named according to first variable in group.
eachName	When TRUE variable names in output instead of indices.
fCorr	When non-null x is not needed as input.

Value

Output is a list containing the groups. First variable has most levels.

Author(s)

Øyvind Langsrud

Examples

```
x <- rep(c("A","B","C"),3)
y <- rep(c(11,22,11),3)
z <- c(1,1,1,2,2,2,3,3,3)
zy <- paste(z,y,sep="")
m <- cbind(x,y,z,zy)
HierarchicalGroups(m)
```

HierarchicalWildcardGlobbing

Find variable combinations by advanced wildcard/globbing specifications.

Description

Find combinations present in an input data frame or, when input is a list, find all possible combinations that meet the requirements.

Usage

```
HierarchicalWildcardGlobbing(z, wg, useUnique = NULL,
  useFactor = FALSE, makeWarning = TRUE, printInfo = FALSE,
  useMatrixToDataFrame = TRUE)
```

Arguments

<code>z</code>	list or data.frame
<code>wg</code>	data.frame with data globbing and wildcards
<code>useUnique</code>	Logical variable about recoding within the algorithm. By default (NULL) an automatic decision is made.
<code>useFactor</code>	When TRUE, internal factor recoding is used.
<code>makeWarning</code>	When TRUE, warning is made in cases of unused variables. Only variables common to <code>z</code> and <code>wg</code> are used.
<code>printInfo</code>	When TRUE, information is printed during the process.
<code>useMatrixToDataFrame</code>	When TRUE, special functions (<code>DataFrameToMatrix</code> / <code>MatrixToDataFrame</code>) for improving speed and memory is utilized.

Details

The final variable combinations must meet the requirements in each positive sign group and must not match the requirements in the negative sign groups. The function is implemented by calling [WildcardGlobbing](#) several times within an algorithm that uses hierarchical clustering (`hclust`).

Value

data.frame

Author(s)

Øyvind Langsrud

Examples

```

# useUnique=NULL betyr valg ut fra antall rader i kombinasjonsfil
data(precip)
data(mtcars)
codes <- as.character(c(100, 200, 300, 600, 700, 101, 102, 103, 104, 134, 647, 783,
                        13401, 13402, 64701, 64702))

# Create list input
zList <- list(car = rownames(mtcars), wt = as.character(1000 * mtcars$wt),
             city = names(precip), code = codes)

# Create data.frame input
m <- cbind(car = rownames(mtcars), wt = as.character(1000 * mtcars$wt))
zFrame <- data.frame(m[rep(1:NROW(m), each = 35), ],
                    city = names(precip), code = codes, stringsAsFactors = FALSE)

# Create globbing/wildcards input
wg <- data.frame(rbind(c("Merc*", "" , "" , "?00" ),
                      c("F*" , "" , "" , "?????"),
                      c("" , "??0", "C*" , "" ),
                      c("" , "" , "!Co*" , "" ),
                      c("" , "" , "?i*" , "???2"),
                      c("" , "" , "?h*" , "???1")),
               sign = c("+", "+", "+", "+", "-", "-"), stringsAsFactors = FALSE)
names(wg)[1:4] <- names(zList)

# =====
# Finding unique combinations present in the input data frame
# =====

# Using first row of wg. Combinations of car starting with Merc
# and three-digit code ending with 00
HierarchicalWildcardGlobbing(zFrame[, c(1, 4)], wg[1, c(1, 4, 5)])

# Using first row of wg. Combinations of all four variables
HierarchicalWildcardGlobbing(zFrame, wg[1, ])

# More combinations when using second row also
HierarchicalWildcardGlobbing(zFrame, wg[1:2, ])

# Less combinations when using third row also
# since last digit of wt must be 0 and only cities starting with C
HierarchicalWildcardGlobbing(zFrame, wg[1:3, ])

```

```

# Less combinations when using fourth row also since city cannot start with Co
HierarchicalWildcardGlobbing(zFrame, wg[1:4, ])

# Less combinations when using fourth row also
# since specific combinations of city and code are removed
HierarchicalWildcardGlobbing(zFrame, wg)

# =====
# Using list input to create all possible combinations
# =====

dim(HierarchicalWildcardGlobbing(zList, wg))

# same result with as.list since same unique values of each variable
dim(HierarchicalWildcardGlobbing(as.list(zFrame), wg))

```

Hierarchies2ModelMatrix

Model matrix representing crossed hierarchies

Description

Make a model matrix, x , that corresponds to data and represents all hierarchies crossed. This means that aggregates corresponding to numerical variables can be computed as $x \%*\% y$, where y is a matrix with one column for each numerical variable.

Usage

```

Hierarchies2ModelMatrix(data, hierarchies, inputInOutput = TRUE,
  crossTable = FALSE, total = "Total", hierarchyVarNames = c(mapsFrom =
  "mapsFrom", mapsTo = "mapsTo", sign = "sign", level = "level"),
  unionComplement = FALSE)

```

Arguments

<code>data</code>	Matrix or data frame with data containing codes of relevant variables
<code>hierarchies</code>	List of hierarchies, which can be converted by AutoHierarchies . Thus, the variables can also be coded by "rowFactor" or "", which correspond to using the categories in the data.
<code>inputInOutput</code>	Logical vector (possibly recycled) for each element of hierarchies. TRUE means that codes from input are included in output. Values corresponding to "rowFactor" or "" are ignored.
<code>crossTable</code>	Cross table in output when TRUE
<code>total</code>	Vector of total codes (possibly recycled) used when running Hrc2DimList

hierarchyVarNames
 Variable names in the hierarchy tables as in [HierarchyFix](#)

unionComplement
 Logical vector (possibly recycled) for each element of hierarchies. When TRUE, sign means union and complement instead of addition or subtraction. Values corresponding to "rowFactor" and "colFactor" are ignored.

Details

This function makes use of [AutoHierarchies](#) and [HierarchyCompute](#) via [HierarchyComputeDummy](#)

Value

A sparse model matrix or a list of two elements (model matrix and cross table)

Author(s)

Øyvind Langsrud

Examples

```

# Create some input
z <- SSBtoolsData("sprt_emp_withEU")
ageHier <- SSBtoolsData("sprt_emp_ageHier")
geoDimList <- FindDimLists(z[, c("geo", "eu")], total = "Europe")[[1]]

# First example has list output
Hierarchies2ModelMatrix(z, list(age = ageHier, geo = geoDimList), inputInOutput = FALSE,
  crossTable = TRUE)

m1 <- Hierarchies2ModelMatrix(z, list(age = ageHier, geo = geoDimList), inputInOutput = FALSE)
m2 <- Hierarchies2ModelMatrix(z, list(age = ageHier, geo = geoDimList))
m3 <- Hierarchies2ModelMatrix(z, list(age = ageHier, geo = geoDimList, year = ""),
  inputInOutput = FALSE)
m4 <- Hierarchies2ModelMatrix(z, list(age = ageHier, geo = geoDimList, year = "allYears"),
  inputInOutput = c(FALSE, FALSE, TRUE))

# Illustrate the effect of unionComplement, geoHier2 as in the examples of HierarchyCompute
geoHier2 <- rbind(data.frame(mapsFrom = c("EU", "Spain"), mapsTo = "EUandSpain", sign = 1),
  SSBtoolsData("sprt_emp_geoHier")[, -4])
m5 <- Hierarchies2ModelMatrix(z, list(age = ageHier, geo = geoHier2, year = "allYears"),
  inputInOutput = FALSE) # Spain is counted twice
m6 <- Hierarchies2ModelMatrix(z, list(age = ageHier, geo = geoHier2, year = "allYears"),
  inputInOutput = FALSE, unionComplement = TRUE)

# Compute aggregates
ths_per <- as.matrix(z[, "ths_per", drop = FALSE]) # matrix with the values to be aggregated
t(m1) %*% ths_per # crossprod(m1, ths_per) is equivalent and faster
t(m2) %*% ths_per
t(m3) %*% ths_per
t(m4) %*% ths_per
t(m5) %*% ths_per

```

```
t(m6) %*% ths_per
```

Hierarchy2Formula *Hierarchy2Formula*

Description

Conversion between to-from coded hierarchy and formulas written with =, - and +.

Usage

```
Hierarchy2Formula(x, hierarchyVarNames = c(mapsFrom = "mapsFrom", mapsTo  
= "mapsTo", sign = "sign", level = "level"))
```

```
Formula2Hierarchy(s)
```

Arguments

x Data frame with to-from coded hierarchy

hierarchyVarNames Variable names in the hierarchy tables as in [HierarchyFix](#).

s Character vector of formulas written with =, - and +.

Value

See Arguments

Author(s)

Øyvind Langsrud

Examples

```
x <- SSBtoolsData("sprt_emp_geoHier")  
s <- Hierarchy2Formula(x)  
Formula2Hierarchy(s)
```

HierarchyCompute *Hierarchical Computations*

Description

This function computes aggregates by crossing several hierarchical specifications and factorial variables.

Usage

```
HierarchyCompute(data, hierarchies, valueVar, rowSelect = NULL,
  colSelect = NULL, inputInOutput = FALSE, output = "data.frame",
  autoLevel = TRUE, unionComplement = FALSE,
  constantsInOutput = NULL, hierarchyVarNames = c(mapsFrom =
  "mapsFrom", mapsTo = "mapsTo", sign = "sign", level = "level"),
  selectionByMultiplicationLimit = 10^7, colNotInDataWarning = TRUE,
  useMatrixToDataFrame = TRUE, handleDuplicated = "sum",
  asInput = FALSE, verbose = FALSE)
```

Arguments

data	The input data frame
hierarchies	A named (names in data) list with hierarchies. Variables can also be coded by "rowFactor" and "colFactor".
valueVar	Name of the variable to be aggregated.
rowSelect	Data frame specifying variable combinations for output. The colFactor variable is not included.
colSelect	Vector specifying categories of the colFactor variable for output.
inputInOutput	Logical vector (possibly recycled) for each element of hierarchies. TRUE means that codes from input are included in output. Values corresponding to "rowFactor" and "colFactor" are ignored.
output	One of "data.frame" (default), "dummyHierarchies", "outputMatrix", "dataDummyHierarchy", "valueMatrix", "fromCrossCode", "toCrossCode", "crossCode" (as toCrossCode), "outputMatrixWithCrossCode", "matrixComponents".
autoLevel	Logical vector (possibly recycled) for each element of hierarchies. When TRUE, level is computed by automatic method as in HierarchyFix . Values corresponding to "rowFactor" and "colFactor" are ignored.
unionComplement	Logical vector (possibly recycled) for each element of hierarchies. When TRUE, sign means union and complement instead of addition or subtraction as in DummyHierarchy . Values corresponding to "rowFactor" and "colFactor" are ignored.
constantsInOutput	A single row data frame to be combine by the other output.
hierarchyVarNames	Variable names in the hierarchy tables as in HierarchyFix .

<code>selectionByMultiplicationLimit</code>	With non-NULL <code>rowSelect</code> and when the number of elements in <code>dataDummyHierarchy</code> exceeds this limit, the computation is performed by a slower but more memory efficient algorithm.
<code>colNotInDataWarning</code>	When TRUE, warning produced when elements of <code>colSelect</code> are not in data.
<code>useMatrixToDataFrame</code>	When TRUE (default) special functionality for saving time and memory is used.
<code>handleDuplicated</code>	Handling of duplicated code rows in data. One of: "sum" (default), "sumByAggregate", "sumWithWarning", "stop" (error), "single" or "singleWithWarning". With no <code>colFactor</code> sum and <code>sumByAggregate</code> / <code>sumWithWarning</code> are different (original values or aggregates in "valueMatrix"). When single, only one of the values is used (by matrix subsetting).
<code>asInput</code>	When TRUE (FALSE is default) output matrices match input data. Thus <code>valueMatrix = Matrix(data[, valueVar], ncol=1)</code> . Only possible when no <code>colFactor</code> .
<code>verbose</code>	Whether to print information during calculations. FALSE is default.

Details

A key element of this function is the matrix multiplication: `outputMatrix = dataDummyHierarchy %*% valueMatrix`. The matrix, `valueMatrix` is a re-organized version of the `valueVar` vector from input. In particular, if a variable is selected as `colFactor`, there is one column for each level of that variable. The matrix, `dataDummyHierarchy` is constructed by crossing dummy coding of hierarchies ([DummyHierarchy](#)) and factorial variables in a way that matches `valueMatrix`. The code combinations corresponding to rows and columns of `dataDummyHierarchy` can be obtained as `toCrossCode` and `fromCrossCode`. In the default data frame output, the `outputMatrix` is stacked to one column and combined with the code combinations of all variables.

Value

As specified by the parameter `output`

Author(s)

Øyvind Langsrud

Examples

```
# Data and hierarchies used in the examples
x <- SSBtoolsData("sprt_emp") # Employment in sport in thousand persons from Eurostat database
geoHier <- SSBtoolsData("sprt_emp_geoHier")
ageHier <- SSBtoolsData("sprt_emp_ageHier")

# Two hierarchies and year as rowFactor
HierarchyCompute(x, list(age = ageHier, geo = geoHier, year = "rowFactor"), "ths_per")

# Same result with year as colFactor (but columns ordered differently)
HierarchyCompute(x, list(age = ageHier, geo = geoHier, year = "colFactor"), "ths_per")
```

```

# Internally the computations are different as seen when output='matrixComponents'
HierarchyCompute(x, list(age = ageHier, geo = geoHier, year = "rowFactor"), "ths_per",
  output = "matrixComponents")
HierarchyCompute(x, list(age = ageHier, geo = geoHier, year = "colFactor"), "ths_per",
  output = "matrixComponents")

# Include input age groups by setting inputInOut = TRUE for this variable
HierarchyCompute(x, list(age = ageHier, geo = geoHier, year = "colFactor"), "ths_per",
  inputInOut = c(TRUE, FALSE))

# Only input age groups by switching to rowFactor
HierarchyCompute(x, list(age = "rowFactor", geo = geoHier, year = "colFactor"), "ths_per")

# Select some years (colFactor) including a year not in input data (zeros produced)
HierarchyCompute(x, list(age = ageHier, geo = geoHier, year = "colFactor"), "ths_per",
  colSelect = c("2014", "2016", "2018"))

# Select combinations of geo and age including a code not in data or hierarchy (zeros produced)
HierarchyCompute(x, list(age = ageHier, geo = geoHier, year = "colFactor"), "ths_per",
  rowSelect = data.frame(geo = "EU", age = c("Y0-100", "Y15-64", "Y15-29")))

# Extend the hierarchy table to illustrate the effect of unionComplement
# Omit level since this is handled by autoLevel
geoHier2 <- rbind(data.frame(mapsFrom = c("EU", "Spain"), mapsTo = "EUandSpain", sign = 1),
  geoHier[, -4])

# Spain is counted twice
HierarchyCompute(x, list(age = ageHier, geo = geoHier2, year = "colFactor"), "ths_per")

# Can be seen in the dataDummyHierarchy matrix
HierarchyCompute(x, list(age = ageHier, geo = geoHier2, year = "colFactor"), "ths_per",
  output = "matrixComponents")

# With unionComplement=TRUE Spain is not counted twice
HierarchyCompute(x, list(age = ageHier, geo = geoHier2, year = "colFactor"), "ths_per",
  unionComplement = TRUE)

# With constantsInOut
HierarchyCompute(x, list(age = ageHier, geo = geoHier, year = "colFactor"), "ths_per",
  constantsInOut = data.frame(c1 = "AB", c2 = "CD"))

```

HierarchyFix

Change the hierarchy table to follow the standard

Description

Make sure that variable names and sign coding follow an internal standard. Level may be computed automatically

Usage

```
HierarchyFix(hierarchy, hierarchyVarNames = c(mapsFrom = "mapsFrom",
  mapsTo = "mapsTo", sign = "sign", level = "level"), autoLevel = TRUE)
```

Arguments

```
hierarchy      data frame with hierarchy table
hierarchyVarNames
                variable names
autoLevel      When TRUE, level is computed by automatic method
```

Value

data frame with hierarchy table

Author(s)

Øyvind Langsrud

Examples

```
# Make input data by changing variable names and sign coding.
h <- SSBtoolsData("FIFA2018ABCD")[, 1:3]
names(h)[1:2] <- c("from", "to")
minus <- h$sign < 0
h$sign <- "+"
h$sign[minus] <- "-"

# Run HierarchyFix - Two levels created
HierarchyFix(h, c(mapsFrom = "from", mapsTo = "to", sign = "sign"))

# Extend the hierarchy table
h2 <- rbind(data.frame(from = c("Oceania", "Asia", "Africa", "America", "Europe"),
  to = "World", sign = "+"),
  data.frame(from = c("World", "Europe"),
  to = "nonEurope", sign = c("+", "-")), h)

# Run HierarchyFix - Three levels created
HierarchyFix(h2, c(mapsFrom = "from", mapsTo = "to", sign = "sign"))
```

MakeHierFormula

Make model formula from data taking into account hierarchical variables

Description

Make model formula from data taking into account hierarchical variables

Usage

```
MakeHierFormula(data = NULL, hGroups = HierarchicalGroups2(data),  
  n = length(hGroups), sim = TRUE)
```

Arguments

data	data frame
hGroups	Output from HierarchicalGroups2()
n	Interaction level or 0 (all levels)
sim	Include "~" when TRUE

Value

Formula as character string

Author(s)

Øyvind Langsrud

Examples

```
x <- SSBtoolsData("sprt_emp_withEU")[, -4]  
MakeHierFormula(x)  
MakeHierFormula(x, n = 2)  
MakeHierFormula(x, n = 0)
```

Match

Matching rows in data frames

Description

The algorithm is based on converting variable combinations to whole numbers. The final matching is performed using [match](#).

Usage

```
Match(x, y)
```

Arguments

x	data frame
y	data frame

Details

When the result of multiplying together the number of unique values in each column of x exceeds 9E15 (largest value stored exactly by the numeric data type), the algorithm is recursive.

Value

An integer vector giving the position in y of the first match if there is a match, otherwise NA.

Author(s)

Øyvind Langsrud

Examples

```
a <- data.frame(x = c("a", "b", "c"), y = c("A", "B"), z = 1:6)
b <- data.frame(x = c("b", "c"), y = c("B", "K", "A", "B"), z = c(2, 3, 5, 6))

Match(a, b)
Match(b, a)

# Slower alternative
match(data.frame(t(a), stringsAsFactors = FALSE), data.frame(t(b), stringsAsFactors = FALSE))
match(data.frame(t(b), stringsAsFactors = FALSE), data.frame(t(a), stringsAsFactors = FALSE))

# More comprehensive example (n, m and k may be changed)
n <- 10^4
m <- 10^3
k <- 10^2
data(precip)
data(mtcars)
y <- data.frame(car = sample(rownames(mtcars), n, replace = TRUE),
                 city = sample(names(precip), n, replace = TRUE),
                 n = rep_len(1:k, n), a = rep_len(c("A", "B", "C", "D"), n),
                 b = rep_len(as.character(rnorm(1000)), n),
                 d = sample.int(k + 10, n, replace = TRUE),
                 e = paste(sample.int(k * 2, n, replace = TRUE),
                           rep_len(c("Green", "Red", "Blue"), n), sep = "_"),
                 r = rnorm(k)^99)
x <- y[sample.int(n, m), ]
row.names(x) <- NULL
ix <- Match(x, y)
```

matlabColon

Simulate Matlab's ':'

Description

Functions to generate increasing sequences

Usage

```
matlabColon(from, to)
```

```
SeqInc(from, to)
```

Arguments

from numeric. The start value
to numeric. The end value.

Details

matlabColon(a,b) returns a:b (R's version) unless a > b, in which case it returns integer(0). SeqInc(a,b) is similar, but results in error when the calculated length of the sequence (1+to-from) is negative.

Value

A numeric vector, possibly empty.

Author(s)

Bjørn-Helge Mevik (matlabColon) and Øyvind Langsrud (SeqInc)

See Also

[seq](#)

Examples

```
identical(3:5, matlabColon(3, 5)) ## => TRUE  
3:1 ## => 3 2 1  
matlabColon(3, 1) ## => integer(0)  
try(SeqInc(3, 1)) ## => Error  
SeqInc(3, 2)        ## => integer(0)
```

Number *Adding leading zeros*

Description

Adding leading zeros

Usage

```
Number(n, width = 3)
```

Arguments

n numeric vector of whole numbers
width width

Value

Character vector

Author(s)

Øyvind Langsrud

Examples

```
Number(1:3)
```

RbindAll

Combining several data frames when the columns don't match

Description

Combining several data frames when the columns don't match

Usage

```
RbindAll(...)
```

Arguments

... Several data frames as several input parameters or a list of data frames

Value

A single data frame

Note

The function is an extended version of `rbind.all.columns` at <https://amywhiteheadresearch.wordpress.com/2013/05/13/combining-dataframes-when-the-columns-dont-match/>

Author(s)

Øyvind Langsrud

See Also

[CbindIdMatch](#) (same example data)

Examples

```

zA=data.frame(idA=1:10,idB=rep(10*(1:5),2),
              idC=rep(c(100,200),5),idC2=c(100,rep(200,9)),idC3=rep(100,10),
              idD=99,x=round(rnorm(10),3),xA=round(runif(10),2))
zB=data.frame(idB=10*(1:5),x=round(rnorm(5),3),xB=round(runif(5),2))
zC=data.frame(idC=c(100,200),x=round(rnorm(2),3),xC=round(runif(2),2))
zD=data.frame(idD=99,x=round(rnorm(1),3),xD=round(runif(1),2))
RbindAll(zA,zB,zC,zD)
RbindAll(list(zA,zB,zC,zD))

```

RowGroups

Create numbering according to unique rows

Description

Create numbering according to unique rows

Usage

```
RowGroups(x, returnGroups = FALSE, returnGroupsId = FALSE)
```

Arguments

x Data frame or matrix
returnGroups When TRUE unique rows are returned
returnGroupsId When TRUE Index of unique rows are returned

Value

A vector with the numbering or, according to the arguments, a list with more output.

Author(s)

Øyvind Langsrud

Examples

```

a <- data.frame(x = c("a", "b"), y = c("A", "B", "A"), z = rep(1:4, 3))
RowGroups(a)
RowGroups(a, TRUE)
RowGroups(a[, 1:2], TRUE, TRUE)
RowGroups(a[, 1, drop = FALSE], TRUE)

```

SSBtoolsData

Function that returns a dataset

Description

Function that returns a dataset

Usage

```
SSBtoolsData(dataset)
```

Arguments

dataset Name of data set within the SSBtools package

Details

FIFA2018ABCD: A hierarchy table based on countries within groups A-D in the football championship, 2018 FIFA World Cup.

sprt_emp: Employment in sport in thousand persons. Data from <https://ec.europa.eu/eurostat/web/sport/employment-in-sport/data/database>

sprt_emp_geoHier: Country hierarchy for the employment in sport data.

sprt_emp_ageHier: Age hierarchy for the employment in sport data.

sprt_emp_withEU: The data set sprt_emp extended with a EU variable.

Value

data frame

Author(s)

Øyvind Langsrud

Examples

```
SSBtoolsData("FIFA2018ABCD")
SSBtoolsData("sprt_emp")
SSBtoolsData("sprt_emp_geoHier")
SSBtoolsData("sprt_emp_ageHier")
SSBtoolsData("sprt_emp_withEU")
```

Stack *Stack columns from a data frame and include variables.*

Description

Stack columns from a data frame and include variables.

Usage

```
Stack(data, stackVar = 1:NCOL(data), blockVar = integer(0),
      rowData = data.frame(stackVar)[, integer(0), drop = FALSE],
      valueName = "values", indName = "ind")
```

Arguments

data	A data frame
stackVar	Indices of variables to be stacked
blockVar	Indices of variables to be replicated
rowData	A separate data frame where NROW(rowData)=length(stackVar) such that each row may contain multiple information of each stackVar variable. The output data frame will contain an extended variant of rowData.
valueName	Name of the stacked/concatenated output variable
indName	Name of the output variable with information of which vector in x the observation originated. When indName is NULL this variable is not included in output.

Value

A data frame where the variable ordering corresponds to: blockVar, rowData, valueName, indName

Author(s)

Øyvind Langsrud

See Also

[Unstack](#)

Examples

```
z <- data.frame(n=c(10,20,30), ssb=c('S','S','B'),
               Ayes=1:3,Ano=4:6,Byes=7:9,Bno=10:12)
zRow <- data.frame(letter=c('A','A','B','B'),answer=c('yes','no','yes','no'))

x <- Stack(z,3:6,1:2,zRow)

Unstack(x,6,3:4,numeric(0),1:2)
```

```
Unstack(x,6,5,numeric(0),1:2)
Unstack(x,6,3:4,5,1:2)
```

UniqueIntegerSortMatrix

Title UniqueIntegerSortMatrix

Description

Title UniqueIntegerSortMatrix

Usage

```
UniqueIntegerSortMatrix(x)
```

Arguments

x x

Value

Matrix

Unstack *Unstack a column from a data frame and include additional variables.*

Description

Unstack a column from a data frame and include additional variables.

Usage

```
Unstack(data, mainVar = 1, stackVar = (1:NCOL(data))[-mainVar],
  extraVar = integer(0), blockVar = integer(0), sep = "_",
  returnRowData = TRUE, sorted = FALSE)
```

Arguments

data	A data frame
mainVar	Index of the variable to be unstacked
stackVar	Index of variables defining the unstack grouping
extraVar	Indices of within-replicated variables to be added to the rowData output
blockVar	Indices of between-replicated variables to be added to the data output
sep	A character string to separate when creating variable names
returnRowData	When FALSE output is no list, but only data
sorted	When TRUE the created variables is in sorted order. Otherwise input order is used.

Value

When returnRowData=TRUE output is list of two elements.

data	Unstacked data
rowData	A separate data frame with one row for each unstack grouping composed of the stackVar variables

Author(s)

Øyvind Langsrud

See Also

[Stack](#) (examples)

WildcardGlobbing	<i>Row selection by wildcard/globbing</i>
------------------	---

Description

The selected rows match combined requirements for all variables.

Usage

```
WildcardGlobbing(x, wg, sign = TRUE, invert = "!")
```

Arguments

x	data.frame with character data
wg	data.frame with wildcard/globbing
sign	When FALSE, the result is inverted.
invert	Character to invert each single selection.

Details

This function is used by [HierarchicalWildcardGlobbing](#) and [WildcardGlobbingVector](#) and make use of [grepl](#) and [glob2rx](#).

Value

Logical vector defining subset of rows.

Author(s)

Øyvind Langsrud

Examples

```

# Create data input
data(precip)
data(mtcars)
x <- data.frame(car = rownames(mtcars)[rep(1:NROW(mtcars), each = 35)], city = names(precip),
                stringsAsFactors = FALSE)

# Create globbing/wildcards input
wg <- data.frame(rbind(c("Merc*", "C*"), c("F*", "??????"), c("!????????*?", "!????????*")),
                stringsAsFactors = FALSE)
names(wg) <- names(x)

# Select the following combinations:
# - Cars starting with Merc and cities starting with C
# - Cars starting with F and six-letter cities
# - Cars with less than nine letters and cities with less than seven letters
x[WildcardGlobbing(x, wg), ]

```

WildcardGlobbingVector

Selection of elements by wildcard/globbing

Description

Selection of elements by wildcard/globbing

Usage

```
WildcardGlobbingVector(x, wg, negSign = "-", invert = "!")
```

Arguments

x	Character vector
wg	Character vector with wildcard/globbing
negSign	Character representing selection to be removed
invert	Character to invert each single selection.

Value

vector with selected elements of x

Author(s)

Øyvind Langsrud

Examples

```
data(precip)
x <- names(precip)

# Select the cities starting with B, C and Sa.
WildcardGlobbingVector(x, c("B*", "C*", "Sa*"))

# Remove from the selection cities with o and t in position 2 and 4, respectively.
WildcardGlobbingVector(x, c("B*", "C*", "Sa*", "-?o*", "-???t*"))

# Add to the selection cities not having six or more letters.
WildcardGlobbingVector(x, c("B*", "C*", "Sa*", "-?o*", "-???t*", "!?????*""))
```

Index

AddLeadingZeros, 2
AutoHierarchies, 3, 18, 19
AutoSplit, 5

CbindIdMatch, 6, 28

DimList2Hierarchy, 4, 7
DimList2Hrc, 8
DummyHierarchy, 9, 21, 22

FactorLevCorr, 10
FindCommonCells, 11
FindDimLists, 4, 12
FindHierarchies (AutoHierarchies), 3
FindTableGroup, 13
Formula2Hierarchy (Hierarchy2Formula),
20
FormulaSums, 14

glob2rx, 33
grepl, 33

hclust, 16
HierarchicalGroups, 15
HierarchicalWildcardGlobbing, 16, 33
Hierarchies2ModelMatrix, 4, 18
Hierarchy2Formula, 4, 20
HierarchyCompute, 4, 19, 21
HierarchyComputeDummy, 19
HierarchyFix, 4, 19–21, 23
Hrc2DimList, 4, 18
Hrc2DimList (DimList2Hrc), 8

MakeHierFormula, 24
Match, 25
match, 25
matlabColon, 26

Number, 27

RbindAll, 7, 28

RowGroups, 29

seq, 27
SeqInc (matlabColon), 26
SSBtoolsData, 30
Stack, 31, 33

UniqueIntegerSortMatrix, 32
Unstack, 31, 32

WildcardGlobbing, 16, 33
WildcardGlobbingVector, 33, 34