OpenMx Reference Manual

October 10, 2009

Version 0.2.1-922

Title The OpenMx Statistical Modeling Package

Author Steven Boker, Michael Neale, Hermine Maes, Paras Meta, Sarah Kenny, Timothy Bates, Ryne Estabrook, Jeffrey Spies, Timothy Brick, Michael Spiegel

Maintainer OpenMx Development Team <openmx-developers@list.mail.virginia.edu>

URL http://terpsichore.psyc.virginia.edu/wiki

Description The OpenMx Project intends to rewrite and extend the popular statistical package Mx to address the challenges facing a large range of modern statistical problems such as: the difficulty of measuring behavioral traits; the availability of technologies - such as magnetic resonance imaging, continuous physiological monitoring and microarrays - which generate extremely large amounts of data often with complex time-dependent patterning; and increased sophistication in the statistical models used to analyze the data.

License Apache License 2.0

Depends methods

Suggests snowfall

LazyLoad yes

LazyData yes

**mxAlgebra**

**Function To Create MxAlgebra Object**

**Description**

This function creates a new *MxAlgebra* object.

**Usage**

```r
mxAlgebra(expression, name = NA, dimnames = NA)
```
**Arguments**

- **expression**: An R expression of matrix operators and matrix functions.
- **name**: An optional character string indicating the name of the object.
- **dimnames**: List. The dimnames attribute for the algebra: a list of length 2 giving the row and column names respectively. An empty list is treated as NULL, and a list of length one as row names. The list can be named, and the list names will be used as names for the dimensions.

**Details**

The `mxAlgebra` function is used to create algebraic expressions that operate on one or more `MxMatrix` objects. To evaluate an `MxAlgebra` object, it must be placed in an `MxModel` object, along with all referenced `MxMatrix` objects and the `mxAlgebraObjective` function. The `mxAlgebraObjective` function must reference the `MxAlgebra` object to be evaluated by name.

The following operations are supported in `mxAlgebra`:

- `solve()`: Inversion
- `t()`: Transposition
- `+`: Addition
- `-`: Subtraction
- `%*%`: Matrix Multiplication
- `*`: Element or dot product
- `/`: Element division
- `%x%`: Kronecker product
- `%%`: Quadratic product
- `cbind()`: Horizontal adhesion
- `rbind()`: Vertical adhesion

The following functions are supported in `mxAlgebra`:

- `sum(diag())`: Trace
- `det()`: Determinant
- `sum()`: Sum
- `max()`: Maximum
- `min()`: Min
- `abs()`: Absolute value
- `sin()`: Sine
- `sinh()`: Hyperbolic sine
- `cos()`: Cosine
- `cosh()`: Hyperbolic cosine
- `tan()`: Tangent
- `tanh()`: Hyperbolic tangent
mxAlgebra

- **exp()**: Exponent
- **log()**: Natural Logarithm
- **sqrt()**: Square root
- **diag()**: Diagonal to vector or vector to diagonal, depending on input
- **c(t())**: Matrix to vector (by row; returns vector)
- **c()**: Matrix to vector (by column)
- **vech()**: Lower triangular matrix to vector (requires MCMCpack package)
- **cov2cor()**: Standardize matrix
- **Re(eigen(A, only.values=TRUE)$values)**: Real eigenvalues (for matrix A)
- **Im(eigen(A, only.values=TRUE)$values)**: Imaginary eigenvalues (for matrix A)
- **Re(eigen(A, )$values)**: Real eigenvectors (for matrix A)
- **Im(eigen(A, )$values)**: Imaginary eigenvectors (for matrix A)
- **colMeans()**: Mean of columns
- **var()**: Covariance of columns

**Value**

Returns a new MxAlgebra object.

**References**

The OpenMx User’s guide can be found at http://openmx.psyc.virginia.edu/documentation.

**See Also**

MxAlgebra for the S4 class created by mxAlgebra. MxMatrix and mxMatrix for objects which may be entered in the 'expression' argument and the function that creates them. More information about the OpenMx package may be found here.

**Examples**

```r
A <- mxMatrix("Full", nrow = 3, ncol = 3, values=2, name = "A")

# Simple example: algebra B simply evaluates to the matrix A
B <- mxAlgebra(A, name = "B")

# Compute A + B
C <- mxAlgebra(A + B, name = "C")

# Compute sin(C)
D <- mxAlgebra(sin(C), name = "D")

# Make a model and evaluate the mxAlgebra object 'D'
A <- mxMatrix("Full", nrow = 3, ncol = 3, values=2, name = "A")
model <- mxModel("AlgebraExample", A, B, C, D)
fit <- mxRun(model)
```
### MxAlgebra-class

**Description**

MxAlgebra is an S4 class. An MxAlgebra object is a named entity. New instances of this class can be created using the function `mxAlgebra`.

**Details**

The MxAlgebra class has the following slots:

<table>
<thead>
<tr>
<th>Slot</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the object</td>
</tr>
<tr>
<td>formula</td>
<td>The R expression to be evaluated</td>
</tr>
<tr>
<td>result</td>
<td>Either NULL or a 1x1 matrix with the computation result</td>
</tr>
</tbody>
</table>

The 'name' slot is the name of the MxAlgebra object. Use of MxAlgebra objects in the `mxConstraint` function or an OpenMx objective function requires reference by name.

The 'formula' slot is an expression containing one or more MxMatrix objects. These objects are operated on or related to one another using one or more operations detailed in the `mxAlgebra` help file.

The 'result' slot is used to hold the results of an optimization on the expression in the 'formula' slot. If this MxAlgebra has not been used as an argument in an objective function and subsequently included in an MxModel object and executed using the `mxRun` function, this slot will have a value of NULL. If this MxAlgebra has been used as an argument in an objective function and subsequently included in an MxModel object and executed using the `mxRun` function, then this slot will have a 1x1 matrix with the results of that optimization as its only value.

Slots may be referenced with the `@` symbol. See the documentation for `Classes` and the examples in the `mxAlgebra` document for more information.

### References

The OpenMx User’s guide can be found at http://openmx.psyc.virginia.edu/documentation.

### See Also

- `mxAlgebra` for creating MxAlgebra objects.
- `MxMatrix` and `mxMatrix` for objects which may be entered in the 'formula' slot and the function that creates them. More information about the OpenMx package may be found here.
mxAlgebraObjective  Function to Create MxAlgebraObjective Object

Description

This function creates a new MxAlgebraObjective object.

Usage

mxAlgebraObjective(algebra)

Arguments

algebra  A character string indicating the name of an MxAlgebra or MxMatrix object to use for optimization.

Details

Objective functions are functions for which free parameter values are chosen such that the value of the objective function is minimized. While the other objective functions in OpenMx are packaged with a function to be optimized (i.e., maximum likelihood), the mxAlgebraObjective function uses the referenced MxAlgebra or MxMatrix object as the function to be minimized.

If no free parameters are included in any part of the 'algebra’ argument, the mxAlgebraObjective works as an algebra evaluator. If the 'algebra' argument is given an MxMatrix object with no free parameters, the original object is returned. If the 'algebra' argument is given an MxAlgebra object whose referenced matrices and algebras have no free parameters, the mxAlgebraObjective function carries out the operations defined in the ‘expression’ argument of that MxAlgebra object.

If free parameters are included in the objects references in the 'algebra’ argument, the mxAlgebraObjective minimizes the algebra defined in the 'algebra’ argument. More than one free parameter may be included in the algebra, but the mxAlgebraObjective should be specified such that the objective function evaluates to a 1x1 matrix or scalar value. If the contents of the 'algebra’ argument evaluates to a non-scalar matrix, only the first entry of that matrix is minimized.

To evaluate, place MxAlgebraObjective objects, referenced MxAlgebra and MxMatrix objects, and optional MxBounds and MxConstraint objects in an MxModel object. This model may then be evaluated using the mxRun function. The results of the optimization can be found in the 'output’ slot of the resulting model, and may be obtained using the mxEval function.

Value

Returns a new MxAlgebraObjective object. MxAlgebraObjective objects should be included with models with referenced MxAlgebra and MxMatrix objects.

References

The OpenMx User’s guide can be found at http://openmx.psyc.virginia.edu/documentation.
Examples

#Create a matrix 'A' with no free parameters
A <- mxMatrix('Full', nrow = 1, ncol = 1, values = c(0), name = 'A')

#Create an algebra 'B', which defines the expression A + A
B <- mxAlgebra(A + A, name = 'B')

#Define the objective function for algebra 'B'
objective <- mxAlgebraObjective('B')

#Place the algebra, its associated matrix and its objective function in a model
model <- mxModel(A, B, objective)

#Evaluate the algebra
modelRun <- mxRun(model)

#View the results
modelRun@output

mxBounds

Function To Create an mxBounds Object

Description

This function creates a new MxBounds object.

Usage

mxBounds(parameters, min = NA, max = NA)

Arguments

parameters A character vector indicating the names of the parameters on which to apply bounds.
min A numeric value for the lower bound. NA means use default value.
max A numeric value for the upper bound. NA means use default value.

Details

Creates a set of boundaries or limits for a parameter or set of parameters. Parameters may be any free parameter or parameters from an MxMatrix object. Parameters may be referenced either by name or by referring to their position in the 'spec' matrix of an MxMatrix object.

Minima and maxima may be specified as scalar numeric values.

Value

Returns a new MxBounds object. If used as an argument in an MxModel object, the parameters referenced in the 'parameters' argument must also be included prior to optimization.
References

The OpenMx User’s guide can be found at http://openmx.psyc.virginia.edu/documentation.

See Also

MxBounds for the S4 class created by mxBounds. MxMatrix and mxMatrix for free parameter specification. More information about the OpenMx package may be found here.

Examples

#Create lower and upper bounds for parameters 'A' and 'B'
bounds <- mxBounds(c('A', 'B'), 3, 5)

#Create a lower bound of zero for a set of variance parameters
varianceBounds <- mxBounds(c('Var1', 'Var2', 'Var3'), 0)

---

MxBounds-class  

MxBounds Class

Description

MxBounds is an S4 class. New instances of this class can be created using the function mxBounds.

Details

The MxBounds class has the following slots:

- **min** - The lower bound
- **max** - The upper bound
- **parameters** - The vector of parameter names

The 'min' and 'max' slots hold scalar numeric values for the lower and upper bounds on the list of parameters, respectively.

Parameters may be any free parameter or parameters from an MxMatrix object. Parameters may be referenced either by name or by referring to their position in the ‘spec’ matrix of an MxMatrix object. To affect an estimation or optimization, an MxBounds object must be included in an MxModel object with all referenced MxAlgebra and MxMatrix objects.

Slots may be referenced with the @ symbol. See the documentation for Classes and the examples in the mxBounds document for more information.

References

The OpenMx User’s guide can be found at http://openmx.psyc.virginia.edu/documentation.
mxConstraint

Function To Create an MxConstraint Object

Description

This function creates a new MxConstraint object.

Usage

mxConstraint(alg1, relation, alg2, name = NA)

Arguments

alg1 A character string indicating the name of an MxAlgebra or MxMatrix object, whose relationship to the object specified in the ‘alg2’ argument is constrained.

relation A character string indicating the relation between ‘alg1’ and ‘alg2’. Must be either "<", "=" or ">".

alg2 A character string indicating the name of an MxAlgebra or MxMatrix object, whose relationship to the object specified in the ‘alg1’ argument is constrained.

name An optional character string indicating the name of this object.

Details

The mxConstraint function defines relationships between two MxAlgebra or MxMatrix objects. They are used to affect the estimation of free parameters in the referenced objects. The relationships "<", "=" and ">" are supported. To affect an estimation or optimization, an MxConstraint object must be included in an MxModel object with all referenced MxAlgebra and MxMatrix objects.

The mxConstraint function may not be used to constrain free parameters, either by name or by their position in an MxMatrix or MxAlgebra object. Free parameters in the same MxModel may be constrained to equality by giving them the same name in their respective ‘spec’ matrices.

Value

Returns an MxConstraint object. If used as an argument in an MxModel object, the objects referenced in the ‘alg1’ and ‘alg2’ arguments must also be included prior to optimization.

References

The OpenMx User's guide can be found at http://openmx.psyc.virginia.edu/documentation.
See Also

MxConstraint for the S4 class created by mxConstraint. MxAlgebra and MxMatrix for objects which may be entered as arguments in the ‘alg’ and ‘alg2’ arguments, and mxAlgebra and mxMatrix for the functions that create them. More information about the OpenMx package may be found here.

Examples

#Create a constraint between MxMatrices 'A' and 'B'
constraint <- mxConstraint('A', '=', 'B', name = 'constraint')

# Constrain a matrix of free parameters 'K' to be equal to matrix 'limit'
K <- mxMatrix(type="Full", nrow=2, ncol=2, free=TRUE, name="K")
limit <- mxMatrix(type="Full", nrow=2, ncol=2, free=FALSE,name="limit", values=1:4)

model<- mxModel("con_test", K,limit,
mxConstraint("K", "="", "limit"),
mxAlgebra(min(K), name="minK"),
mxAlgebraObjective("minK")
)
fit <- mxRun(model)
fit@matrices$K@values
# [,1] [,2]
# [1,] 1 3
# [2,] 2 4
objects.
Slots may be referenced with the @ symbol. See the documentation for Classes and the examples in the mxConstraint document for more information.

References
The OpenMx User's guide can be found at http://openmx.psyc.virginia.edu/documentation.

See Also
mxConstraint for the function that creates MxConstraint objects. MxAlgebra and MxMatrix for objects which may be entered as arguments in the 'alg' and 'alg2' arguments, and mxAlgebra and mxMatrix for the functions that create them. More information about the OpenMx package may be found here.

mxData

Function To Create MxData Object

Description
This function creates a new MxData object.

Usage
mxData(observed, type, means = NA, numObs = NA)

Arguments

- observed: A matrix or data.frame which provides data to the MxData object.
- type: A character string defining the type of data in the ‘observed’ argument. Must be one of “raw”, “cov”, “cor”, or “sscp”.
- means: An optional vector of means for use when ‘type’ is “cov”, or “cor”.
- numObs: The number of observations in the data supplied in the ‘observed’ argument. Required unless ‘type’ equals “raw”.

Details
The mxData function creates MxData objects, which can be used as arguments in MxModel objects. The ‘observed’ argument may take either a data frame or a matrix, which is then described with the ‘type’ argument. Data types describe compatibility and usage with objective functions in MxModel objects. Four different data types are supported:

- raw: The contents of the ‘observed’ argument are treated as raw data. Missing values are permitted and must be designated as the system missing value. The ‘means’ and ‘numObs’ arguments cannot be specified, as the ‘means’ argument is not relevant and the ‘numObs’ argument is automatically populated with the number of rows in the data. Data of this type must use the mxFIMLObjective function as its objective function in MxModel objects, which deals with covariance estimation under full-information maximum likelihood.
cov  The contents of the ‘observed’ argument are treated as a covariance matrix. The ‘means’
argument is not required, but may be included for estimations involving means. The ‘num-
Obs’ argument is required, which should reflect the number of observations or rows in the
data described by the covariance matrix. Data of this type may use the mxMLObjective, or
mxRAMObjective functions, depending on the specified model.

cor  The contents of the ‘observed’ argument are treated as a correlation matrix. The ‘means’
argument is not required, but may be included for estimations involving means. The ‘num-
Obs’ argument is required, which should reflect the number of observations or rows in the
data described by the covariance matrix. Data of this type may use the mxMLObjective, or
mxRAMObjective functions, depending on the specified model.

sscp  The contents of the ‘observed’ argument are treated as a sums-of-squares and cross-products
matrix. The ‘means’ argument is not used. The ‘numObs’ argument is required, which should
reflect the number of observations or rows in the data described by the covariance matrix.
Data of this type may use the mxMLObjective, or mxRAMObjective functions, depending on
the specified model.

MxData objects may not be included in MxAlgebra objects or use the mxAlgebraObjective func-
tion. If these capabilities are desired, data should be appropriately input or transformed using the
mxMatrix and mxAlgebra functions.

While column names are stored in the ‘observed’ slot of MxData objects, these names are not
recognized as variable names in MxPath objects. Variable names must be specified using the ‘man-
ifestVars’ argument of the mxModel function prior to use in MxPath objects.

The mxData function does not currently place restrictions on the size, shape, or symmetry of matrices
input into the ‘observed’ argument. While it is possible to specify MxData objects as covariance,
correlation or sscp matrices that do not have the properties commonly associated with these matrices,
failure to correctly specify these matrices will likely lead to problems in model estimation.

Value

Returns a new MxData object.

References

The OpenMx User’s guide can be found at http://openmx.psyc.virginia.edu/documentation.

See Also

MxData for the S4 class created by mxData. matrix and data.frame for objects which may be
entered as arguments in the ‘observed’ slot. More information about the OpenMx package may be
found here.

Examples

#Create a covariance matrix
covMatrix <- matrix( c(0.77642931, 0.39590663,
                        0.39590663, 0.49115615),
                       nrow = 2, ncol = 2, byrow = TRUE)

#Create an MxData object including that covariance matrix
MxData-class

MxData Class

Description

MxData is an S4 class. An MxData object is a named entity. New instances of this class can be created using the function `mxData`. MxData is an S4 class union. An MxData object is either NULL or a MxNonNullData object.

Details

The MxNonNullData class has the following slots:

- **name**: The name of the object
- **observed**: Either a matrix or a data frame
- **vector**: A vector for means, or NA if missing
- **type**: Either 'raw', 'cov', 'cor', or 'sscp'
- **numObs**: The number of observations

The 'name' slot is the name of the MxData object.

The 'observed' slot is used to contain data, either as a matrix or as a data frame. Use of the data in this slot by other functions depends on the value of the 'type' slot. When 'type' is equal to 'cov', 'cor', or 'sscp', the data input into the 'matrix' slot should be a symmetric matrix or data frame.

The 'vector' slot is used to contain a vector of numeric values, which is used as a vector of means for MxData objects with 'type' equal to 'cov', 'cor', or 'sscp'. This slot may be used in estimation using the `mxMLObjective` function.

The 'type' slot may take one of four supported values:

- **raw**: The contents of the 'observed' slot are treated as raw data. Missing values are permitted and must be designated as the system missing value. The 'vector' and 'numObs' slots cannot be specified, as the 'vector' argument is not relevant and the 'numObs' argument is automatically populated with the number of rows in the data. Data of this type must use the `mxFIMLObjective` function as its objective function in MxModel objects, which deals with covariance estimation under full-information maximum likelihood.

- **cov**: The contents of the 'observed' slot are treated as a covariance matrix. The 'vector' argument is not required, but may be included for estimations involving means. The 'numObs' slot is required. Data of this type may use the `mxMLObjective`, or `mxRAMObjective` functions, depending on the specified model.

- **cor**: The contents of the 'observed' slot are treated as a correlation matrix. The 'vector' argument is not required, but may be included for estimations involving means. The 'numObs' slot
is required. Data of this type may use the mxMLObjective, or mxRAMObjective functions, depending on the specified model.

sscp The contents of the ‘observed’ slot are treated as a sums-of-squares and cross-products matrix. The ‘vector’ argument is not required, but may be included for estimations involving means. The ‘numObs’ slot is required. Data of this type may use the mxMLObjective, or mxRAMObjective functions, depending on the specified model.

The ‘numObs’ slot describes the number of observations in the data. If ‘type’ equals ‘raw’, then ‘numObs’ is automatically populated as the number of rows in the matrix or data frame in the ‘observed’ slot. If ‘type’ equals ‘cov’, ‘cor’, or ‘sscp’, then this slot must be input using the ‘numObs’ argument in the mxData function when the MxData argument is created.

MxData objects may not be included in MxAlgebra objects or use the mxAlgebraObjective function. If these capabilities are desired, data should be appropriately input or transformed using the mxMatrix and mxAlgebra functions.

While column names are stored in the ‘observed’ slot of MxData objects, these names are not recognized as variable names in MxPath objects. Variable names must be specified using the ‘manifestVars’ argument of the mxModel function prior to use in MxPath objects.

The mxData function does not currently place restrictions on the size, shape, or symmetry of matrices input into the ‘observed’ argument. While it is possible to specify MxData objects as covariance, correlation or sscp matrices that do not have the properties commonly associated with these matrices, failure to correctly specify these matrices will likely lead to problems in model estimation.

References

The OpenMx User’s guide can be found at http://openmx.psyc.virginia.edu/documentation.

See Also

mxData for creating MxData objects, matrix and data.frame for objects which may be entered as arguments in the ‘matrix’ slot. More information about the OpenMx package may be found here.

mxEval Function To Evaluate MxModel Values

Description

This function can be used to evaluate an arbitrary R expression that includes named entities from a MxModel object, or labels from a MxMatrix object.

Usage

mxEval(expression, model, compute = FALSE, show = FALSE)
**Arguments**

- **expression**: An arbitrary R expression.
- **model**: The model in which to evaluate the expression.
- **compute**: If TRUE then compute the value of algebra expressions.
- **show**: If TRUE then print the translated expression.

**Details**

The argument ‘expression’ is an arbitrary R expression. Any named entities that are used within the R expression are translated into their current value from the model. Any labels from the matrices within the model are translated into their current value from the model. Finally the expression is evaluated and the result is returned. To enable debugging, the ‘show’ argument has been provided. The most common mistake when using this function is to include named entities in the model that are identical to R function names. For example, if a model contains a named entity named ‘c’, then the following mxEval call will return an error: mxEval(c(A, B, C), model).

If ‘compute’ is FALSE, then MxAlgebra expressions returns their current value as they have been computed by the optimization call (using mxRun). If the ‘compute’ argument is TRUE, then MxAlgebra expressions will be calculated in R. Any references to an objective function that has not yet been calculated will return a 1 x 1 matrix with a value of NA.

**References**

The OpenMx User’s guide can be found at http://openmx.psyc.virginia.edu/documentation.

**Examples**

```r
matrixA <- mxMatrix("Full", nrow = 1, ncol = 1, 
        values = 1, name = "A")

algebraB <- mxAlgebra(A + A, name = "B")

model <- mxModel(matrixA, algebraB)
model <- mxRun(model)
start <- mxEval(-pi * A, model)

## Not run:
mxEval(plot(sin, start, B * pi), model)

# The statement above is equivalent to:

plot(sin, -pi, 2 * pi)

## End(Not run)
```
mxFIMLObjective  Function To Create MxFIMLObjective Object

Description

This function creates a new MxFIMLObjective object.

Usage

mxFIMLObjective(covariance, means, dimnames = NA, thresholds = NA)

Arguments

covariance  A character string indicating the name of the expected covariance algebra.
means       A character string indicating the name of the expected means algebra.
dimnames   An optional character vector to be assigned to the dimnames of the covariance and means algebras.
thresholds An optional character string indicating the name of the thresholds matrix.

Details

Objective functions are functions for which free parameter values are chosen such that the value of the objective function is minimized. The mxFIMLObjective function uses full-information maximum likelihood to provide maximum likelihood estimates of free parameters in the algebra defined by the 'covariance' and 'means' arguments. The 'covariance' argument takes an MxAlgebra object, which defines the expected covariance of an associated MxData object. The 'means' argument takes an MxAlgebra object, which defines the expected means of an associated MxData object. The 'dimnames' arguments takes an optional character vector. If this argument is not a single NA, then this vector be assigned to be the dimnames of the means vector, and the row and columns dimnames of the covariance matrix.

mxFIMLObjective evaluates with respect to an MxData object. The MxData object need not be referenced in the mxFIMLObjective function, but must be included in the MxModel object. mxFIMLObjective requires that the 'type' argument in the associated MxData object be equal to 'raw'. Missing values are permitted in the associated MxData object.

To evaluate, place MxFIMLObjective objects, the mxData object for which the expected covariance approximates, referenced MxAlgebra and MxMatrix objects, and optional MxBounds and MxConstraint objects in an MxModel object. This model may then be evaluated using the mxRun function. The results of the optimization can be found in the 'output' slot of the resulting model, and may be referenced using the Extract functionality.

Value

Returns a new MxFIMLObjective object. MxFIMLObjective objects should be included with models with referenced MxAlgebra, MxData and MxMatrix objects.
References

The OpenMx User’s guide can be found at http://openmx.psyc.virginia.edu/documentation.

Examples

```r
A <- mxMatrix(values = 0.5, nrow = 2, ncol = 1,
              free = TRUE, name = "A")
D <- mxMatrix(type = "Diag", values = c(0, 0.5),
              free = c(FALSE, TRUE), nrow = 2, name = "D")
M <- mxMatrix(type = "Zero", nrow = 1, ncol = 2, name = "M")
expectedCov <- mxAlgebra(A %*% t(A) + D, "expectedCov")
objective <- mxFIMLObjective("expectedCov", "M")
model <- mxModel(A, D, expectedCov, objective)
```

mxMatrix

Function To Create MxMatrix Object

Description

This functions creates a new MxMatrix object.

Usage

```r
mxMatrix(type = "Full", nrow = NA, ncol = NA,
          free = FALSE, values = NA, labels = NA, lbound = NA,
          ubound = NA, byrow = getOption('mxByrow'), dimnames = NA, name = NA)
```

Arguments


nrow the desired number of rows. One or both of ‘nrow’ and ‘ncol’ is required when ‘values’, ‘free’, ‘labels’, ‘lbound’, and ‘ubound’ arguments are not matrices, depending on the matrix type.

ncol the desired number of columns. One or both of ‘nrow’ and ‘ncol’ is required when ‘values’, ‘free’, ‘labels’, ‘lbound’, and ‘ubound’ arguments are not matrices, depending on the matrix type.

free a vector or matrix of logicals for free parameter specification. A single ‘TRUE’ or ‘FALSE’ will set all allowable variables to free or fixed, respectively.
values
a vector or matrix of numeric starting values. By default, all values are set to zero.

labels
a vector or matrix of characters for variable label specification.

lbounds
a vector or matrix of numeric lower bounds. Default bounds are specified with an NA.

ubounds
a vector or matrix of numeric upper bounds. Default bounds are specified with an NA.

byrow
logical. If ‘FALSE’ (default), the ‘values’, ‘free’, ‘labels’, ‘lbounds’, and ‘ubounds’ matrices are populated by column rather than by row.

dimnames
list. The dimnames attribute for the matrix: a list of length 2 giving the row and column names respectively. An empty list is treated as NULL, and a list of length one as row names. The list can be named, and the list names will be used as names for the dimensions.

name
an optional character string indicating the name of the MxMatrix object created by the mxModel function.

Details
The mxMatrix function creates MxMatrix objects, which consist of a pair of matrices and a ‘type’ argument. The ‘values’ matrix is made up of numeric elements whose usage and capabilities in other functions are defined by the ‘free’ matrix. If an element is specified as a fixed parameter in the ‘free’ matrix, then the element in the ‘values’ matrix is treated as a constant value and cannot be altered or updated by an objective function when included in an mxRun function. If an element is specified as a free parameter in the ‘free’ matrix, the element in the ‘value’ matrix is considered a starting value and can be changed by an objective function when included in an mxRun function. Free parameters are specified with a character string, non-zero numeric value, or ‘NA’; fixed parameters are specified with a numeric zero.

Objects created by the mxMatrix function are of a specific ‘type’, which specifies the number and location of parameters in the ‘labels’ matrix and the starting values in the ‘values’ matrix. Input ‘values’, ‘free’, and ‘labels’ matrices must be of appropriate shape and have appropriate values for the matrix type requested. Nine types of matrices are supported:

- **‘Diag’** matrices must be square, and only elements on the principle diagonal may be specified as free parameters or take non-zero values. All other elements are fixed and have a value of 0.
- **‘Full’** matrices may be either rectangular or square, and all elements in the matrix may be freely estimated. This type is the default for the mxMatrix() function.
- **‘Iden’** matrices must be square, and consist of no free parameters. Matrices of this type have a value of 1 for all entries on the principle diagonal.
- **‘Lower’** matrices must be square, with a value of 0 for all entries in the upper triangle and no free parameters in the upper triangle.
- **‘Sdiag’** matrices must be square, with a value of 0 for all entries in the upper triangle and along the diagonal. No free parameters in the upper triangle or along the diagonal.
- **‘Symm’** matrices are symmetric matrices (see ‘Symm’) with 1’s along the main diagonal.
- **‘Stand’** matrices may be either rectangular or square, and contain no free parameters. All elements in matrices of this type have a value of 1 for all entries on the principle diagonal.
- **‘Unit’** matrices may be either rectangular or square, and contain no free parameters. All elements in matrices of this type have a value of 0 for all entries on the principle diagonal.
- **‘Zero’** matrices may be either rectangular or square, and contain no free parameters. All elements in matrices of this type have a value of 0 for all entries on the principle diagonal.

When ‘type’ is either ‘Lower’, ‘Sdiag’, ‘Symm’, or ‘Stand’, and the arguments to ‘free’, ‘values’, ‘labels’, ‘lbounds’, or ‘ubounds’ are vectors with enough elements to populate exactly one half of the matrix, then mxMatrix() populates the lower triangle of the matrix (and transposes the lower triangle if the matrix is symmetric).
**Value**

Returns a new `MxMatrix` object, which consists of a ‘values’ matrix of numeric starting values, a ‘free’ matrix describing free parameter specification, a ‘labels’ matrix of labels for the variable names, and ‘lbound’ and ‘ubound’ matrices of the lower and upper parameter bounds. This `MxMatrix` object can be used as an argument in the `mxAlgebra`, `mxBounds`, `mxConstraint` and `mxModel` functions.

**References**

The OpenMx User’s guide can be found at http://openmx.psyc.virginia.edu/documentation.

**See Also**

`MxMatrix` for the S4 class created by `mxMatrix`. More information about the OpenMx package may be found here.

**Examples**

```r
# Create a 3 x 3 identity matrix
idenMatrix <- mxMatrix(type = "Iden", nrow = 3,
                        ncol = 3, name = "I")

# Create a full 4 x 2 matrix from existing
# value matrix with all free parameters
vals <- matrix(1:8, nrow = 4)
fullMatrix <- mxMatrix(type = "Full", values = vals,
                        free = TRUE, name = "foo")

# Create a 3 x 3 symmetric matrix with free off-
# diagonal parameters and starting values
symmMatrix <- mxMatrix(type = "Symm", nrow = 3, ncol = 3,
                        free = c(FALSE, TRUE, TRUE, FALSE, TRUE, FALSE),
                        values = c(1, .8, .8, 1, .8, 1),
                        labels = c(NA, "free1", "free2", NA, "free3", NA),
                        name = "bar")
```

---

**Description**

MxMatrix is an S4 class. An MxMatrix object is a **named entity**. New instances of this class can be created using the function `mxMatrix`. MxMatrix objects may be used as arguments in other functions from the OpenMx library, including `mxAlgebra`, `mxConstraint`, and `mxModel`. 
Details

The MxMatrix class has the following slots:

- name - the name of the object
- free - the free matrix
- values - the values matrix
- labels - the labels matrix

The 'name' slot is the name of the MxMatrix object. Use of MxMatrix objects in an mxAlgebra or mxConstraint function requires reference by name.

The 'free' slot takes a matrix which describes the location of free and fixed parameters. A variable is a free parameter if-and-only-if the corresponding value in the 'free' matrix is 'TRUE'. Free parameters are elements of an MxMatrix object whose values may be changed by an objective function when that MxMatrix object is included in an MxModel object and evaluated using the mxRun function.

The 'values' slot takes a matrix of numeric values. If an element is specified as a fixed parameter in the 'free' matrix, then the element in the 'values' matrix is treated as a constant value and cannot be altered or updated by an objective function when included in an mxRun function. If an element is specified as a free parameter in the 'free' matrix, the element in the 'value' matrix is considered a starting value and can be changed by an objective function when included in an mxRun function.

The 'labels' slot takes a matrix which describes the labels of free and fixed parameters. Fixed parameters with identical labels must have identical values. Free parameters with identical labels impose an equality constraint. The same label cannot be applied to a free parameter and a fixed parameter. A free parameter with the label 'NA' implies a unique free parameter, that cannot be constrained to equal any other free parameter.

References

The OpenMx User’s guide can be found at http://openmx.psyc.virginia.edu/documentation.

See Also

mxMatrix for creating MxMatrix objects. More information about the OpenMx package may be found here.

mxMLOObjective Function To Create MxMLObjective Object

Description

This function creates a new MxMLObjective object.

Usage

mxMLObjective(covariance, means = NA, dimnames = NA, thresholds = NA)
mxMLObjective

Arguments

covariance  A character string indicating the name of the expected covariance algebra.
means  An optional character string indicating the name of the expected means algebra.
dimnames  An optional character vector to be assigned to the dimnames of the covariance and means algebras.
thresholds  An optional character string indicating the name of the thresholds matrix.

Details

Objective functions are functions for which free parameter values are chosen such that the value of the objective function is minimized. The mxMLObjective function uses full-information maximum likelihood to provide maximum likelihood estimates of free parameters in the algebra defined by the 'covariance' argument given the covariance of an MxData object. The 'covariance' argument takes an MxAlgebra object, which defines the expected covariance of an associated MxData object. The 'dimnames' arguments takes an optional character vector. If this argument is not a single NA, then this vector be assigned to be the dimnames of the means vector, and the row and columns dimnames of the covariance matrix.

mxMLObjective evaluates with respect to an MxData object. The MxData object need not be referenced in the mxMLObjective function, but must be included in the MxModel object. mxMLObjective requires that the 'type' argument in the associated MxData object be equal to 'cov', 'cov', or 'sscp'. The 'covariance' argument of this function evaluates with respect to the 'matrix' argument of the associated MxData object, while the 'means' argument of this function evaluates with respect to the 'vector' argument of the associated MxData object. The 'means' and 'vector' arguments are optional in both functions. If the 'means' argument is not specified (NA), the optional 'vector' argument of the MxData object is ignored. If the 'means' argument is specified, the associated MxData object should specify a 'means' argument of equivalent dimension as the 'means' algebra.

To evaluate, place MxMLObjective objects, the mxData object for which the expected covariance approximates, referenced MxAlgebra and MxMatrix objects, and optional MxBound and MxConstraint objects in an MxModel object. This model may then be evaluated using the mxRun function. The results of the optimization can be found in the 'output' slot of the resulting model, or using the mxEval function.

Value

Returns a new MxMLObjective object. MxMLObjective objects should be included with models with referenced MxAlgebra, MxData and MxMatrix objects.

References

The OpenMx User's guide can be found at http://openmx.psyc.virginia.edu/documentation.

Examples

vars <- c('x', 'y')

A <- mxMatrix(values = 0.5, nrow = 2, ncol = 1,
              free = TRUE, name = "A")
mxModel

Function To Create MxModel Object

Description

This function creates a new MxModel object.

Usage

mxModel(model = NA, ..., manifestVars = NA, latentVars = NA,
         remove = FALSE, independent = NA, type = NA, name = NA)

Arguments

model

This argument is either an MxModel object or a string. If 'model' is an MxModel object, then all elements of that model are placed in the resulting MxModel object. If 'model' is a string, then a new model is created with the string as its name. If 'model' is either unspecified or 'model' is a named entity, data source, or MxPath object, then a new model is created.

... An arbitrary number of named entities, data sources, or MxPath objects. These will all be added or removed from the model as specified in the 'model' argument, based on the 'remove' argument.

manifestVars A list of manifest variables to be included in the model.

latentVars A list of latent variables to be included in the model.

remove logical. If TRUE, elements listed in this statement are removed from the original model. If FALSE, elements listed in this statement are added to the original model.

independent logical. If TRUE then the model is independent.

type character vector. The name of the model type to assign to this model.

name An optional character vector indicating the name of the object.
mxModel

Details

The mxModel function is used to create MxModel objects. Objects created by this function may be new, or may be modified versions of existing MxModel objects. To create a new MxModel object, omit or specify a character string in the 'model' argument. To create a modified version of an existing MxModel object, include this model in the 'model' argument.

Other named-entities may be added as arguments to the mxModel function, which are then added to or removed from the model specified in the 'model' argument. MxAlgebra objects, MxBounds objects, MxConstraint objects, MxData objects, MxMatrix objects, MxModel objects and objective functions may all be added in this way. MxModel objects that are included as arguments will be considered sub-models of the output model, and may be estimated separately or jointly depending on shared parameters and the 'independent' flag discussed below. Only one MxData object and one objective function may be included per model, but there are no restrictions on the number of other named-entities included in an mxModel statement.

All other arguments must be named (i.e. 'latentVars = names'), or they will be interpreted as elements of the ellipsis list. The 'manifestVars' and 'latentVars' arguments specify the names of the manifest and latent variables, respectively, for use with the mxPath function. The 'remove' argument may be used when mxModel is used to create a modified version of an existing MxMatrix object. When 'remove' is set to TRUE, the listed objects are removed from the model specified in the 'model' argument. When 'remove' is set to FALSE, the listed objects are added to the model specified in the 'model' argument.

Model independence may be specified with the 'independent' argument. If a model is independent ('independent = TRUE'), then the parameters of this model are not shared with any other model. An independent model may be estimated with no dependency on any other model. If a model is not independent ('independent = FALSE'), then this model shares parameters with one or more other models such that these models must be jointly estimated. These dependent models must be entered as arguments in another model, so that they are simultaneously optimized.

The model type is determined by a character vector supplied to the 'type' argument. The type of a model is a dynamic property, ie. it is allowed to change during the lifetime of the model. To see a list of available types, use the mxTypes command. When a new model is created and no type is specified, the type specified by options("mxDefaultType") is used.

To be estimated, MxModel objects must include objective functions as arguments (mxAlgebraObjective, mxFIMLObjective, mxMLObjective or mxRAMObjective) and executed using the mxRun function. When MxData objects are included in models, the 'type' argument of these objects may require or exclude certain objective functions, or set an objective function as default.

Named entities in MxModel objects may be viewed and referenced by name using double brackets (model["matrixname"]). Slots may be referenced with the @ symbol (model@data). See the documentation for Classes and the examples in this document for more information.

Value

Returns a new MxModel object. MxModel objects must include an objective function to be used as arguments in mxRun functions.

References

The OpenMx User’s guide can be found at http://openmx.psyc.virginia.edu/documentation.
See Also

MxModel for the S4 class created by mxMatrix. More information about the OpenMx package may be found here.

Examples

# Create an empty model, and place it in an object.
model <- mxModel()

# Create a model named 'firstdraft' with one matrix
model <- mxModel('firstdraft',
  mxMatrix('Full', nrow = 3, ncol = 3, name = "A")
)

# Add other matrices to model 'firstdraft', and rename that model 'finaldraft'
model <- mxModel(model, 
  mxMatrix('Symm', nrow = 3, ncol = 3, name = "S"),
  mxMatrix('Iden', nrow = 3, name = "F"),
  name = "finaldraft")

# Add data to the model from an existing data frame in object 'data'
data <- data.frame()
model <- mxModel(model, mxData(data, type = 'raw'))

# View the matrix named "A" in MxModel object 'model'
model[['A']]

# View the data associated with MxModel object 'model'
model@data

MxModel-class

MxModel Class

Description

MxModel is an S4 class. An MxModel object is a named entity. New instances of this class can be created using the function mxModel.

Details

The MxModel class has the following slots:

- name - The name of the object
- matrices - A list of MxMatrix objects
- algebras - A list of MxAlgebra objects
- submodels - A list of MxModel objects
- constraints - A list of MxConstraint objects
- bounds - A list of MxBounds objects
MxModel-class

latentVars - A list of latent variables
manifestVars - A list of manifest variables
data - A MxData object
objective - Either NULL or a MxObjective object
independent - TRUE if-and-only-if the model is independent
options - A list of optimizer options
output - A list with optimization results

The ‘name’ slot is the name of the MxModel object.

The ‘matrices’ slot contains a list of the MxMatrix objects included in the model. These objects are listed by name. Two objects may not share the same name. If a new MxMatrix is added to an MxModel object with the same name as an MxMatrix object in that model, the added version replaces the previous version. There is no imposed limit on the number of MxMatrix objects that may be added here.

The ‘algebras’ slot contains a list of the MxAlgebra objects included in the model. These objects are listed by name. Two objects may not share the same name. If a new MxAlgebra is added to an MxModel object with the same name as an MxAlgebra object in that model, the added version replaces the previous version. All MxMatrix objects referenced in the included MxAlgebra objects must be included in the ‘matrices’ slot prior to estimation. There is no imposed limit on the number of MxAlgebra objects that may be added here.

The ‘submodels’ slot contains references to all of the MxModel objects included as submodels of this MxModel object. Models held as arguments in other models are considered to be submodels. These objects are listed by name. Two objects may not share the same name. If a new submodel is added to an MxModel object with the same name as an existing submodel, the added version replaces the previous version. When a model containing other models is executed using mxRun, all included submodels are executed as well. If the submodels are dependent on one another, they are treated as one larger model for purposes of estimation.

The ‘constraints’ slot contains a list of the MxConstraint objects included in the model. These objects are listed by name. Two objects may not share the same name. If a new MxConstraint is added to an MxModel object with the same name as an existing MxConstraint object in that model, the added version replaces the previous version. All MxMatrix objects referenced in the included MxConstraint objects must be included in the ‘matrices’ slot prior to estimation. There is no imposed limit on the number of MxAlgebra objects that may be added here.

The ‘bounds’ slot contains a list of the MxBounds objects included in the model. These objects are listed by name. Two objects may not share the same name. If a new MxBounds is added to an MxModel object with the same name as an existing MxBounds object in that model, the added version replaces the previous version. All MxMatrix objects referenced in the included MxBounds objects must be included in the ‘matrices’ slot prior to estimation. There is no imposed limit on the number of MxAlgebra objects that may be added here.

The ‘latentVars’ slot contains a list of latent variable names, which may be referenced by MxPath objects. This slot defaults to 'NA', and is only used when the mxPath function is used.

The ‘manifestVars’ slot contains a list of latent variable names, which may be referenced by MxPath objects. This slot defaults to 'NA', and is only used when the mxPath function is used.

The ‘data’ slot contains an MxData object. This slot must be filled prior to execution when an objective function referencing data is used. Only one MxData object may be included per model,
but submodels may have their own data in their own ‘data’ slots. If an MxData object is added to
an MxModel which already contains an MxData object, the new object replaces the existing one.

The ‘objective’ slot contains an objective function. This slot must be filled prior to using the mxRun
function for model execution and optimization. MxAlgebra, MxData, and MxMatrix objects re-
quired by the included objective function must be included in the appropriate slot of the MxModel
prior to using mxRun.

The ‘independent’ slot contains a logical value indicating whether or not the model is independent.
If a model is independent (independent=TRUE), then the parameters of this model are not shared
with any other model. An independent model may be estimated with no dependency on any other
model. If a model is not independent (independent=FALSE), then this model shares parameters
with one or more other models such that these models must be jointly estimated. These dependent
models must be entered as submodels of another MxModel objects, so that they are simultaneously
optimized.

The ‘options’ slot contains a list of options for the optimizer. The name of each entry in the list is
the option name to be passed to the optimizer. The values in this list are the values of the optimizer
options. The standard interface for updating options is through the mxOption function.

The ‘output’ slot contains a list of output added to the model by the mxRun function. Output
includes parameter estimates, optimization information, model fit, and other information as dictated
by the objective function. If a model has not been optimized using the mxRun function, the ‘output’
slot will be ’NULL’.

Named entities in MxModel objects may be viewed and referenced by name using double brackets
(model["matrixname"])). Slots may be referenced with the @ symbol (model@data). See the
documentation for Classes and the examples in mxModel for more information.

References

The OpenMx User’s guide can be found at http://openmx.psyc.virginia.edu/documentation.

See Also

mxModel for creating MxModel objects. More information about the OpenMx package may be
found here.

mxOption

Set or Clear an Optimizer Option

Description

The function sets or clears an option that is specific to the optimizer in the back-end.

Usage

mxOption(model, key, value, reset = FALSE)
Arguments

- **model**: An `MxModel` object
- **key**: The name of the option.
- **value**: The value of the option.
- **reset**: If TRUE then reset all options to their defaults.

Details

Sets an option that is specific to the particular optimizer used in the back-end. The name of the option is the 'key' argument. Use `value = NULL` to remove an existing option. Before the model is submitted to the back-end, all keys and values are converted into strings using the `as.character` function. To reset all options to their default values, use `reset = TRUE`. If `reset = TRUE`, then 'key' and 'value' are ignored. Use `getOption('mxOptimizerOptions')` to see the default optimizer options.

Value

Returns the model with the optimizer option either set or cleared.

References

The OpenMx User’s guide can be found at http://openmx.psyc.virginia.edu/documentation.

Examples

```r
model <- mxModel()

model <- mxOption(model, "Function Precision", 1e-5)
model <- mxOption(model, "Hessian", "No")
model <- mxOption(model, "Function Precision", NULL)
```

mxPath  

*Function To Create List of Paths*

Description

This function creates a list of paths.

Usage

```r
mxPath(from, to = NA, all = FALSE, arrows = 1,
       free = TRUE, values = NA, labels = NA,
       lbound = NA, ubound = NA)
```
Arguments

- **from**: character vector. These are the sources of the new paths.
- **to**: character vector. These are the sinks of the new paths.
- **all**: boolean value. If TRUE, then connect all sources to all sinks.
- **arrows**: numeric value. Must be either 1 for single-headed or 2 for double-headed arrows.
- **free**: boolean vector. Indicates whether paths are free or fixed.
- **values**: numeric vector. The starting values of the parameters.
- **labels**: character vector. The names of the paths.
- **lbound**: numeric vector. The lower bounds of free parameters.
- **ubound**: numeric vector. The upper bounds of free parameters.

Details

The `mxPath` function creates MxPath objects, which are lists of paths describing the relationships between variables in a model using the RAM modeling approach (McArdle and MacDonald, 1984). Variables are referenced by name, which are included in the ‘manifestVar’ and ‘latentVar’ arguments of the `mxModel` function.

Paths are specified as going from one variable or set of variables to another variable or set of variables using the ‘from’ and ‘to’ arguments, respectively. Sets of variables may be input as a vector of variable names. If the ‘all’ argument is set to FALSE, then paths are created going from each entry in the ‘from’ vector to the corresponding position in the ‘to’ vector. If the ‘to’ and ‘from’ vectors are of different lengths when the ‘all’ argument is set to FALSE, the shorter vector is repeated to make the vectors of equal length. If the ‘all’ argument is set to TRUE, all possible paths from the vector of ‘from’ variables to the vector of ‘to’ variables are created.

The ‘free’ argument specifies whether the paths created by the `mxPath` function are free or fixed parameters. This argument may take either TRUE for free parameters, FALSE for fixed parameters, or a vector of TRUEs and FALSEs to be applied in order to the created paths.

The ‘arrows’ argument specifies the type of paths created. A value of 1 indicates a one-headed arrow representing regression. This path represents a regression of the ‘to’ variable on the ‘from’ variable, such that the arrow points to the ‘to’ variable in a path diagram. A value of 2 indicates a two-headed arrow, representing a covariance or variance. If multiple paths are created in the same `mxPath` function, then the ‘arrows’ argument may take a vector of 1s and 2s to be applied to the set of created paths.

The ‘values’ is a numeric vectors containing the starting values of the created paths. ‘values’ gives a starting value for estimation. The ‘labels’ argument specifies the names of the resulting MxPath object. The ‘lbound’ and ‘ubound’ arguments specify lower and upper bounds for the created paths.

Value

Returns a list of paths.
mxRAMObjective

References


The OpenMx User’s guide can be found at http://openmx.psyc.virginia.edu/documentation.

Examples

```r
myManifest <- sprintf("%02d", c(1:100))
myLatent <- c("G1", "G2", "G3", "G4", "G5")
model <- mxModel(type = "RAM", manifestVars = myManifest, latentVars = myLatent)

singles <- list()
for (i in 1:5) {
  j <- i*20
  singles <- c(singles, mxPath(from = myLatent[i],
                        to = myManifest[(j - 19) : j],
                        arrows = 1,
                        free = c(FALSE, rep(TRUE, 19)),
                        values = c(1, rep(0.75, 19))))
}
model <- mxModel(model, singles)

doubles <- mxPath(from = myLatent, all = TRUE, arrows = 2,
                  free = TRUE, values = 1)
model <- mxModel(model, doubles)
```

mxRAMObjective  
*Function To Create MxRAMObjective Object*

Description

This function creates a new MxRAMObjective object.

Usage

```r
mxRAMObjective(A, S, F, M = NA, thresholds = NA)
```

Arguments

- **A**  
  A character string indicating the name of the ’A’ matrix.
- **S**  
  A character string indicating the name of the ’S’ matrix.
- **F**  
  A character string indicating the name of the ’F’ matrix.
- **M**  
  An optional character string indicating the name of the ’M’ matrix.
- **thresholds**  
  An optional character string indicating the name of the thresholds matrix.
mxRAMObjective

Details

Objective functions are functions for which free parameter values are chosen such that the value of the objective function is minimized. The mxRAMObjective provides maximum likelihood estimates of free parameters in a model of the covariance of a given MxData object. This model is defined by reticular action modeling (McArdle and McDonald, 1984). The 'A', 'S', and 'F' arguments must refer to MxMatrix objects with the associated properties of the A, S, and F matrices in the RAM modeling approach.

The 'A' argument refers to the A or asymmetric matrix in the RAM approach. This matrix consists of all of the asymmetric paths (one-headed arrows) in the model. A free parameter in any row and column describes a regression of the variable represented by that row regressed on the variable represented in that column.

The 'S' argument refers to the S or symmetric matrix in the RAM approach, and as such must be square. This matrix consists of all of the symmetric paths (two-headed arrows) in the model. A free parameter in any row and column describes a covariance between the variable represented by that row and the variable represented by that column. Variances are covariances between any variable at itself, which occur on the diagonal of the specified matrix.

The 'F' argument refers to the F or filter matrix in the RAM approach. If no latent variables are included in the model (i.e., the A and S matrices are of both of the same dimension as the data matrix), then the 'F' should refer to an identity matrix. If latent variables are included (i.e., the A and S matrices are not of the same dimension as the data matrix), then the 'F' argument should consist of a horizontal adhesion of an identity matrix and a matrix of zeros.

The 'M' argument refers to the M or means matrix in the RAM approach. It is a 1 x n matrix, where n is the number of manifest variables + the number of latent variables. The M matrix must be specified if either the mxData type is “cov” or “cor” and a means vector is provided, or if the mxData type is “raw”. Otherwise the M matrix is ignored.

The MxMatrix objects included as arguments may be of any type, but should have the properties described above. The mxRAMObjective will not return an error for incorrect specification, but incorrect specification will likely lead to estimation problems or errors in the mxRun function.

mxRAMObjective evaluates with respect to an MxData object. The MxData object need not be referenced in the mxRAMObjective function, but must be included in the MxModel object. mxRAMObjective requires that the 'type' argument in the associated MxData object be equal to 'cov', 'cor' or 'sscp'.

To evaluate, place MxRAMObjective objects, the mxData object for which the expected covariance approximates, referenced MxAlgebra and MxMatrix objects, and optional MxBounds and MxConstraint objects in an MxModel object. This model may then be evaluated using the mxRun function. The results of the optimization can be found in the 'output' slot of the resulting model, and may be obtained using the mxEval function.

Value

Returns a new MxRAMObjective object. MxRAMObjective objects should be included with models with referenced MxAlgebra, MxData and MxMatrix objects.

References

Examples

```r
matrixA <- mxMatrix("Full", values=c(0,0.2,0,0), name="A", nrow=2, ncol=2)
matrixS <- mxMatrix("Full", values=c(0.8,0,0,0.8), name="S", nrow=2, ncol=2, free=TRUE)
matrixF <- mxMatrix("Full", values=c(1,0,0,1), name="F", nrow=2, ncol=2)

# Create a RAM objective with default A, S, F matrix names
objective <- mxRAMObjective("A", "S", "F")

model <- mxModel(matrixA, matrixS, matrixF, objective)
```

---

### mxRObjective

#### Function to Create MxRObjective Object

**Description**

This function creates a new MxRObjective object.

**Usage**

```r
mxRObjective(objfun)
```

**Arguments**

- `objfun`: A function that accepts two arguments.

**Details**

The `objfun` argument must be a function that accepts two arguments. The first argument is the `mxModel` that should be evaluated, and the second argument is some persistent state information that can be stored between one iteration of optimization to the next iteration. It is valid for the function to simply ignore the second argument.

The function must return either a single numeric value, or a list of exactly two elements. If the function returns a list, the first argument must be a single numeric value and the second element will be the new persistent state information to be passed into this function at the next iteration. The single numeric value will be used by the optimizer to perform optimization. The initial default value for the persistent state information is `NA`.

**Value**

Returns a new MxRObjective object.
References

The OpenMx User’s guide can be found at http://openmx.psyc.virginia.edu/documentation.

Examples

\[ A <- \text{mxMatrix}(\text{nrow} = 2, \text{ncol} = 2, \text{values} = c(1:4), \text{free} = \text{TRUE}, \text{name} = 'A') \]

\[
\text{squared} <- \text{function}(x) \{ x ^ 2 \}
\]

\[
\text{objFunction} <- \text{function}(\text{model}, \text{state}) \{
    \text{values} <- \text{model}[['A']]@values
    \text{return}(\text{squared(values}[1,1] - 4) + \text{squared(values}[1,2] - 3) +
    \text{squared(values}[2,1] - 2) + \text{squared(values}[2,2] - 1))
\}
\]

\[
\text{objective} <- \text{mxROobjective}(\text{objFunction})
\]

\[
\text{model} <- \text{mxModel}(\text{'model'}, A, \text{objective})
\]

\[ \text{mxRun} \quad \text{Send a Model to the Optimizer} \]

Description

This function begins optimization on the top-level model.

Usage

\[ \text{mxRun}(\text{model}) \]

Arguments

\text{model} \quad \text{An MxModel object to be optimized.}

Details

The mxRun function is used to optimize free parameters in MxModel objects based on an objective function. MxModel objects included in the mxRun function must include an appropriate objective function.

Free parameters are estimated or updated based on the objective function. These estimated values, along with estimation information and model fit, can be found in the 'output' slot of MxModel objects after mxRun has been used.

If a model is dependent on or shares parameters with another model, both models must be included as arguments in another MxModel object. This top-level MxModel object must include objective functions in both submodels, as well as an additional objective function describing how the results of the first two should be combined.
mxTypes

Value

Returns an MxModel object with free parameters updated to their final values. The return value contains an "output" slot with the results of optimization.

References

The OpenMx User's guide can be found at http://openmx.psyc.virginia.edu/documentation.

Examples

#Create a model that includes data, matrices A, S and F, and an objective function
## Not run:
data <- mxData(mydata, type="cov", numObs = 100)
objective <- mxRAMObjective('A', 'S', 'F')
model <- mxModel("mymodel", A, S, F, data, objective)

#Use mxRun to optimize the free parameters in the matrices A and S
model <- mxRun(model)

#print the output
model@output
## End(Not run)
mxVersion  
*Returns Current Version String*

**Description**

This function returns a string with the current version number of OpenMx.

**Usage**

mxVersion()

**References**

The OpenMx User’s guide can be found at http://openmx.psyc.virginia.edu/documentation.

**Examples**

mxVersion()
omxCheckCloseEnough

Approximate Equality Testing Function

Description

This function tests whether two numeric vectors or matrixes are approximately equal to one another, within a specified threshold.

Usage

omxCheckCloseEnough(a, b, epsilon = 10^(-15))

Arguments

- **a**: A numeric vector or matrix.
- **b**: A numeric vector or matrix.
- **epsilon**: A non-negative tolerance threshold.

Details

Arguments ‘a’ and ‘b’ must be of the same type, i.e., they must be either vectors of equal dimension or matrices of equal dimension. The two arguments are compared element-wise for approximate equality. If the absolute value of the difference of any two values is greater than the threshold, then an error will be thrown. If ‘a’ and ‘b’ are approximately equal to each other, by default the function will print a statement informing the user the test has passed. To turn off these print statements use options("mxPrintUnitTests" = FALSE).

References

The OpenMx User’s guide can be found at http://openmx.psyc.virginia.edu/documentation.

See Also

- omxCheckWithinPercentError for approximate equality testing within a percentage.
- omxCheckEquals for exact equality testing.
- omxCheckSetEquals for set equality testing.
- omxCheckTrue for boolean equality testing.

Examples

omxCheckCloseEnough(c(1, 2, 3), c(1.1, 1.9, 3.0), epsilon = 0.5)

omxCheckCloseEnough(matrix(3, 3, 3), matrix(4, 3, 3), epsilon = 2)

# Throws an error
omxCheckEquals

**Exact Equality Testing Function**

**Description**

This function tests whether two objects are equal.

**Usage**

omxCheckEquals(a, b)

**Arguments**

- **a**: The first value to compare.
- **b**: The second value to compare.

**Details**

Performs the `==` operator on the two arguments. If the two arguments are not equal, then an error will be thrown. If ‘a’ and ‘b’ are equal to each other, by default the function will print a statement informing the user the test has passed. To turn off these print statements use `options("mxPrintUnitTests" = FALSE)`.

**References**

The OpenMx User’s guide can be found at http://openmx.psyc.virginia.edu/documentation.

**See Also**

- omxCheckCloseEnough for approximate equality testing within an epsilon.
- omxCheckSetEquals for set equality testing.
- omxCheckTrue for boolean equality testing.

**Examples**

```r
omxCheckEquals(c(1, 2, 3), c(1, 2, 3))

omxCheckEquals(FALSE, FALSE)

# Throws an error
try(omxCheckEquals(c(1, 2, 3), c(2, 1, 3)))
```
omxCheckSetEquals  Set Equality Testing Function

Description

This function tests whether two vectors contain the same elements.

Usage

omxCheckSetEquals(a, b)

Arguments

a  The first vector to compare.
b  The second vector to compare.

Details

Performs the 'setequal' function on the two arguments. If the two arguments do not contain the same elements, then an error will be thrown. If ‘a’ and ‘b’ contain the same elements, by default the function will print a statement informing the user the test has passed. To turn off these print statements use options("mxPrintUnitTests" = FALSE).

References

The OpenMx User’s guide can be found at http://openmx.psyc.virginia.edu/documentation.

See Also

omxCheckCloseEnough for approximate equality testing within an epsilon.
CheckWithinPercentError for approximate equality testing within a percentage.
omxCheckEquals for exact equality testing.
omxCheckTrue for boolean equality testing.

Examples

omxCheckSetEquals(c(1, 1, 2, 2, 3), c(3, 2, 1))
omxCheckSetEquals(matrix(1, 1, 1), matrix(1, 3, 3))

# Throws an error
try(omxCheckSetEquals(c(1, 2, 3, 4), c(2, 1, 3)))
omxCheckTrue  

Boolean Equality Testing Function

Description

This function tests whether an object is equal to TRUE.

Usage

omxCheckTrue(a)

Arguments

a

The value to test.

Details

Checks element-wise whether an object is equal to TRUE. If any of the elements are false, then an error will be thrown. If ‘a’ is TRUE, by default the function will print a statement informing the user the test has passed. To turn off these print statements use options("mxPrintUnitTests" = FALSE).

References

The OpenMx User’s guide can be found at http://openmx.psyc.virginia.edu/documentation.

See Also

omxCheckCloseEnough for approximate equality testing within an epsilon.
CheckWithinPercentError for approximate equality testing within a percentage.
omxCheckEquals for exact equality testing.
omxCheckSetEquals for set equality testing.

Examples

omxCheckTrue(1 + 1 == 2)
omxCheckTrue(matrix(TRUE, 3, 3))
# Throws an error
try(omxCheckTrue(FALSE))
omxCheckWithinPercentError

Approximate Percent Equality Testing Function

**Description**

This function tests whether two numeric vectors or matrixes are approximately equal to one another, within a specified percentage.

**Usage**

```r
omxCheckWithinPercentError(a, b, percent = 0.1)
```

**Arguments**

- `a`: A numeric vector or matrix.
- `b`: A numeric vector or matrix.
- `percent`: A non-negative percentage.

**Details**

Arguments ‘a’ and ‘b’ must be of the same type, i.e. they must be either vectors of equal dimension or matrices of equal dimension. The two arguments are compared element-wise for approximate equality. If the absolute value of the difference of any two values is greater than the percentage difference of ‘a’, then an error will be thrown. If ‘a’ and ‘b’ are approximately equal to each other, by default the function will print a statement informing the user the test has passed. To turn off these print statements use `options("mxPrintUnitTests" = FALSE)`.

**References**

The OpenMx User’s guide can be found at http://openmx.psyc.virginia.edu/documentation.

**See Also**

- `omxCheckCloseEnough` for approximate equality testing within an epsilon.
- `omxCheckEquals` for exact equality testing.
- `omxCheckSetEquals` for set equality testing.
- `omxCheckTrue` for boolean equality testing.

**Examples**

```r
omxCheckWithinPercentError(c(1, 2, 3), c(1.1, 1.9, 3.0), percent = 50)
omxCheckWithinPercentError(matrix(3, 3, 3), matrix(4, 3, 3), percent = 150)
# Throws an error
```
try(omxCheckWithinPercentError(c(1, 2, 3), c(1.1, 1.9, 3.0), percent = 0.01))

---

**OpenMx**

*OpenMx: Package for Matrix Algebra Optimization*

### Description

OpenMx is a package for structural equation modeling, matrix algebra optimization and other statistical estimation problems.

### Details

OpenMx is a package for algebra optimization and statistical estimation problems using matrix algebra. The OpenMx library defines a set of S4 classes and functions used to create them. The majority of these classes are used as arguments in models, which may include data, matrices, algebras, bounds and constraints. These models are then paired with objective functions, either existing (maximum likelihood, FIML) or user-defined with included algebra functions. These models can then be optimized, resulting in parameter estimation, algebra evaluation, and output for additional models.

Objects used or created by OpenMx may be of the following classes: `MxAlgebra`, `MxBounds`, `MxConstraint`, `MxData`, `MxMatrix`, `MxModel`, and `MxPath`. Objects of these classes may be created by the following OpenMx functions: `mxAlgebra`, `mxBounds`, `mxConstraint`, `mxData`, `mxMatrix`, `mxModel`, and `mxPath`. The functions `mxAlgebraObjective`, `mxFIMLObjective`, `mxMLObjective` and `mxRAMObjective` create objective functions for model estimation. Models which include objective functions may be estimated using the `mxRun` function.

### References

The OpenMx User’s guide can be found at http://openmx.psyc.virginia.edu/documentation.
Index

as.character, 26
Classes, 5, 8, 10, 23, 25
data.frame, 12, 13
Extract, 16, 33
here, 3, 5, 7–10, 12, 13, 18, 20, 23, 26
matrix, 12, 13
MxAlgebra, 2, 3, 5, 6, 8–11, 13, 15, 16, 20–22, 24, 25, 30, 33, 39
MxAlgebra (MxAlgebra-class), 4
mxAlgebra, 2, 4, 5, 9–11, 13, 18, 19, 39
MxAlgebra-class, 4
mxAlgebraObjective, 5, 11, 13, 22, 39
MxBounds, 6, 7, 16, 20, 22, 25, 30, 39
MxBounds (MxBounds-class), 7
mxBounds, 6, 7, 8, 18, 39
MxBounds-class, 7
MxConstraint, 6, 8, 9, 16, 20, 22, 25, 30, 33, 39
MxConstraint (MxConstraint-class), 9
mxConstraint, 4, 8, 9, 10, 18, 19, 39
MxConstraint-class, 9
MxData, 10–12, 15, 16, 20–25, 29, 30, 33, 39
MxData (MxData-class), 12
mxData, 10, 12, 13, 16, 20, 30, 39
MxData-class, 12
mxEval, 6, 14, 20, 30
mxFIMLObjective, 11, 13, 15, 22, 39
MxMatrix, 2–10, 14, 16–18, 20–22, 24, 25, 29, 30, 33, 39
MxMatrix (MxMatrix-class), 19
mxMatrix, 3, 5, 7–11, 13, 16, 19, 20, 39
MxMatrix-class, 19
mxMLObjective, 11–13, 20, 22, 39
MxModel, 2, 4, 6–11, 14–16, 19–23, 25, 26, 29, 30, 32, 33, 39
MxModel (MxModel-class), 24
mxModel, 11, 13, 18, 19, 21, 24–27, 39
MxModel-class, 24
mxOption, 25, 26
MxPath, 11, 13, 25, 27, 28, 39
MxPath (mxPath), 27
mxPath, 22, 25, 27, 39
mxRAMObjective, 11, 13, 22, 28, 39
mxRObjective, 30
mxRun, 4, 6, 14, 16, 17, 19, 20, 22–25, 29, 30, 31, 39
mxTypes, 22, 32
mxVersion, 33

Named entities, 23, 25
called entity, 4, 9, 12, 19, 24
Named-entities (Named-entity), 33
called-entities, 22
named-entities (Named-entity), 33
Named-entity, 33
named-entity (Named-entity), 33
NULL, 12

omxCheckCloseEnough, 34, 35–38
omxCheckEquals, 34, 35, 36–38
omxCheckSetEquals, 34, 35, 36, 37, 38
omxCheckTrue, 34–36, 37, 38
omxCheckWithinPercentError, 34–37, 38
OpenMx, 4, 39