Package 'umx'

August 28, 2015

Version 0.50

Date 2015-08-27

Title Helper Functions for Structural Equation Modelling in OpenMx

Maintainer Timothy C Bates <timothy.c.bates@gmail.com>

License GPL-3

URL http://github.com/tbates/umx

Description Helper functions for making, running, and reporting SEM models in OpenMx.
If you are just starting, try typing ?umx.

Depends R (>= 3.0.3), OpenMx (>= 2.2.0)

Imports formula.tools, utils, MASS, Matrix, methods, mvtnorm,
numDeriv, polycor, R2HTML, RCurl

Suggests knitr

BugReports http://github.com/tbates/umx/issues

LazyData true

NeedsCompilation no

Author Timothy C Bates [aut, cre]

Repository CRAN

Date/Publication 2015-08-28 01:11:25

R topics documented:

`coef.MxModel` *Get the coefficients of an MxModel*

Description

Returns the coefficients from an OpenMx RAM model

Usage

```
## S3 method for class 'MxModel'
coef(object, ...)
```

Arguments

object	an mxModel from which to get the AIC
...	Optional parameters

Value

- coefficients

References

-

See Also

Other Reporting functions: RMSEA.MxModel; RMSEA.summary.mxmodel; RMSEA; confint.MxModel; extractAIC.MxModel; logLik.MxModel; plot.MxModel, umxPlot; plot, plot.MxModel.ACE, umxPlotACE; residuals.MxModel; umxCI_boot; umxCI; umxCompare; umxExpCov; umxExpMeans; umxFitIndices; umxStandardizeModel; umxSummary.MxModel; umx_drop_ok

Examples

```
require(OpenMx)
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
m1 <- mxModel("One Factor", type = "RAM",
manifestVars = manifests, latentVars = latents,
mxPath(from = latents, to = manifests),
mxPath(from = manifests, arrows = 2),
mxPath(from = latents, arrows = 2, free = FALSE, values = 1.0),
mxData(cov(demoOneFactor), type = "cov", numObs = 500)
)
m1 = umxRun(m1, setLabels = TRUE, setValues = TRUE)
coef(m1)
# -2.615998
```

---

confint.MxModel *Get confidence intervals from an MxModel*

Description

Implements confidence interval function for OpenMx models. Note: Currently requested CIs are added to existing CIs, and all are run, even if they alrady exist in the output. This might change in the future.

Usage

```
## S3 method for class 'MxModel'
confint(object, parm = list("existing", c("vector", "of",
  "names"), "default = add all"), level = 0.95, run = FALSE,
  showErrorCodes = FALSE, ...)
```

Arguments

object	An mxModel, possibly already containing mxCIs that have been mxRun with intervals = TRUE))
parm	A specification of which parameters are to be given confidence intervals. Can be "existing", "all", or a vector of names.
level	The confidence level required (default = .95)
run	Whether to run the model (defaults to FALSE)
showErrorCodes	(default = FALSE)
...	Additional argument(s) for umxConfint.

Details

Unlike confint, if parm is missing, all CIs requested will be added to the model, but (because these can take time to run) by default only CIs already computed will be reported.

CIs will be run only if run is TRUE, allowing this function to be used to add CIs without automatically having to run them. If parm is empty, and run = FALSE, a message will alert you to add run = TRUE. Even a few CIs can take too long to make running the default.

Value

- mxModel

References

- http://www.github.com/tbates/umx

See Also

- confint

Other Reporting functions: RMSEA.MxModel; RMSEA.summary.mxmodel; RMSEA; coef.MxModel; extractAIC.MxModel; logLik.MxModel; plot.MxModel, umxPlot; plot, plot.MxModel.ACE, umxPlotACE; residuals.MxModel; umxCI_boot; umxCI; umxCompare; umxExpCov; umxExpMeans; umxFitIndices; umxStandardizeModel; umxSummary.MxModel; umx_drop_ok

Examples

```
require(OpenMx)
data(demoOneFactor)
latents = c("G")
manifests = names(demoOneFactor)
```

```
m1 <- mxModel("One Factor", type = "RAM",
manifestVars = manifests, latentVars = latents,
mxPath(from = latents, to = manifests),
mxPath(from = manifests, arrows = 2),
mxPath(from = latents, arrows = 2, free = FALSE, values = 1.0),
mxData(cov(demoOneFactor), type = "cov", numObs = 500)
)
m1 = umxRun(m1, setLabels = TRUE, setValues = TRUE)
m2 = confint(m1) # default: CIs added, but user prompted to set run = TRUE
m2 = confint(m2, run = TRUE) # CIs run and reported
# Add CIs for asymmetric paths in RAM model, report them, save m1 with this CI added
m1 = confint(m1, parm = "G_to_x1", run = TRUE)
# Add CIs for asymmetric paths in RAM model, report them, save m1 with mxCIs added
m1 = confint(m1, parm = "A", run = TRUE)
confint(m1, parm = "existing") # request existing CIs (none added yet...)
```

`dl_from_dropbox` *dl_from_dropbox*

Description

Download a file from Dropbox, given either the url, or the name and key

Usage

```
dl_from_dropbox(x, key = NULL)
```

Arguments

x	Either the file name, or full dropbox URL (see example below)
key	the code after s/ and before the file name in the dropbox url

Details

Improvements would include error handling...

Value

- NULL

References

- http://thebiobucket.blogspot.kr/2013/04/download-files-from-dropbox.html

See Also

Other Miscellaneous File Functions: umx_move_file; umx_open; umx_rename_file; umx, umx-package

Examples

```
## Not run:
dl_from_dropbox("https://dl.dropboxusercontent.com/s/7kauod48r9cfhwc/tinytwinData.rda")
dl_from_dropbox("tinytwinData.rda", key = "7kauod48r9cfhwc")

## End(Not run)
```

`extractAIC.MxModel` *extractAIC from MxModel*

Description

Returns the AIC for an OpenMx model helper function for `logLik.MxModel` (which enables AIC(model); logLik(model); BIC(model) Original Author: brandmaier

Usage

```
## S3 method for class 'MxModel'
extractAIC(fit, scale, k, ...)
```

Arguments

`fit`	an fitted `mxModel` from which to get the AIC
`scale`	not used
`k`	not used
`...`	any other parameters (not used)

Value

- AIC value

References

- http://openmx.psyc.virginia.edu/thread/931#comment-4858

See Also

- `AIC`, `umxCompare`, `logLik.MxModel`

Other Reporting functions: `RMSEA.MxModel`; `RMSEA.summary.mxmodel`; `RMSEA`; `coef.MxModel`; `confint.MxModel`; `logLik.MxModel`; `plot.MxModel`, `umxPlot`; `plot`, `plot.MxModel.ACE`, `umxPlotACE`; `residuals.MxModel`; `umxCI_boot`; `umxCI`; `umxCompare`; `umxExpCov`; `umxExpMeans`; `umxFitIndices`; `umxStandardizeModel`; `umxSummary.MxModel`; `umx_drop_ok`

Examples

```
require(OpenMx)
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
m1 <- mxModel("One Factor", type = "RAM",
manifestVars = manifests, latentVars = latents,
mxPath(from = latents, to = manifests),
mxPath(from = manifests, arrows = 2),
mxPath(from = latents, arrows = 2, free = FALSE, values = 1.0),
mxData(cov(demoOneFactor), type = "cov", numObs = 500)
)
m1 = umxRun(m1, setLabels = TRUE, setValues = TRUE)
extractAIC(m1)
# -2.615998
AIC(m1)
```

`logLik.MxModel` *logLik.MxModel*

Description

Returns the log likelihood for an OpenMx model. This helper also enables `AIC`(model); `BIC`(model).

Usage

```
## S3 method for class 'MxModel'
logLik(object, ...)
```

Arguments

`object`	the `mxModel` from which to get the log likelihood
`...`	Optional parameters

Details

hat-tip Andreas Brandmaier

Value

- the log likelihood

References

- http://openmx.psyc.virginia.edu/thread/931#comment-4858

See Also

- AIC, umxCompare

Other Reporting functions: RMSEA.MxModel; RMSEA.summary.mxmodel; RMSEA; coef.MxModel; confint.MxModel; extractAIC.MxModel; plot.MxModel, umxPlot; plot, plot.MxModel.ACE, umxPlotACE; residuals.MxModel; umxCI_boot; umxCI; umxCompare; umxExpCov; umxExpMeans; umxFitIndices; umxStandardizeModel; umxSummary.MxModel; umx_drop_ok

Examples

```
require(OpenMx)
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
m1 <- mxModel("One Factor", type = "RAM",
manifestVars = manifests, latentVars = latents,
mxPath(from = latents, to = manifests),
mxPath(from = manifests, arrows = 2),
mxPath(from = latents, arrows = 2, free = FALSE, values = 1.0),
mxData(cov(demoOneFactor), type = "cov", numObs = 500)
)
m1 = umxRun(m1, setLabels = TRUE, setValues = TRUE)
logLik(m1)
AIC(m1)
```

---

plot.MxModel *Create a figure from an MxModel*

Description

Create graphical path diagrams from your OpenMx models!

Usage

```
## S3 method for class 'MxModel'
plot(x = NA, std = TRUE, digits = 2,
  dotFilename = "name", pathLabels = c("none", "labels", "both"),
  showFixed = FALSE, showMeans = TRUE, showError = TRUE, ...)
```

Arguments

x	an mxModel from which to make a path diagram
std	Whether to standardize the model.
digits	The number of decimal places to add to the path coefficients
dotFilename	A file to write the path model to. if you leave it at the default "name", then the model's internal name will be used
pathLabels	Whether to show labels on the paths. both will show both the parameter and the label. ("both", "none" or "labels")

showFixed	Whether to show fixed paths (defaults to FALSE)
showMeans	Whether to show means
showError	Whether to show errors
...	Optional parameters

References

- http://www.github.com/tbates/umx

See Also

- umxLabel, umxRun, umxValues

Other Reporting functions: RMSEA.MxModel; RMSEA.summary.mxmodel; RMSEA; coef.MxModel; confint.MxModel; extractAIC.MxModel; logLik.MxModel; plot, plot.MxModel.ACE, umxPlotACE; residuals.MxModel; umxCI_boot; umxCI; umxCompare; umxExpCov; umxExpMeans; umxFitIndices; umxStandardizeModel; umxSummary.MxModel; umx_drop_ok

Examples

```
## Not run:
require(OpenMx)
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
m1 <- mxModel("One Factor", type = "RAM",
manifestVars = manifests, latentVars = latents,
mxPath(from = latents, to = manifests),
mxPath(from = manifests, arrows = 2),
mxPath(from = latents, arrows = 2, free = FALSE, values = 1.0),
mxData(cov(demoOneFactor), type = "cov", numObs = 500)
)
m1 = umxRun(m1, setLabels = TRUE, setValues = TRUE)
plot(m1)

## End(Not run)
```

---

qm *qm*

Description

Quickmatrix function

Usage

```
qm(..., rowMarker = "|")
```

Arguments

...	the components of your matrix
rowMarker	mark the end of each row

Value

- matrix

References

http://www.sumsar.net/blog/2014/03/a-hack-to-create-matrices-in-R-matlab-style

See Also

Other Miscellaneous Utility Functions: umx_check_OS; umx_find_object; umx_grep; umx_pb_note; umx, umx-package

Examples

```
# simple example
qm(0, 1 |
   2, NA)
## Not run:
# clever example
M1 = M2 = diag(2)
qm(M1,c(4,5) | c(1,2),M2 | t(1:3))

## End(Not run)
```

---

reliability *reliability*

Description

Compute and report Coefficient alpha

Usage

```
reliability(S)
```

Arguments

S	A square, symmetric, numeric covariance matrix

Value

-

References

- https://cran.r-project.org/package=Rcmdr

See Also

- cov

Other Miscellaneous Stats Functions: umxCov2cor; umx_cor; umx_means; umx, umx-package

Examples

```
# treat vehicle aspects as items of a test
reliability(cov(mtcars))
```

---

residuals.MxModel *Get residuals from an MxModel*

Description

Return the residuals from an OpenMx RAM model

Usage

```
## S3 method for class 'MxModel'
residuals(object, digits = 2, suppress = NULL, ...)
```

Arguments

object	An fitted mxModel from which to get residuals
digits	rounding (default = 2)
suppress	smallest deviation to print out (default = NULL = show all)
...	Optional parameters

Value

- matrix of residuals

References

- http://tbates.github.io, https://github.com/tbates/umx

See Also

Other Reporting functions: RMSEA.MxModel; RMSEA.summary.mxmodel; RMSEA; coef.MxModel; confint.MxModel; extractAIC.MxModel; logLik.MxModel; plot.MxModel, umxPlot; plot, plot.MxModel.ACE, umxPlotACE; umxCI_boot; umxCI; umxCompare; umxExpCov; umxExpMeans; umxFitIndices; umxStandardizeModel; umxSummary.MxModel; umx_drop_ok

Examples

```
require(OpenMx)
data(demoOneFactor)
latents  = c("g")
manifests = names(demoOneFactor)
m1 <- mxModel("One Factor", type = "RAM",
manifestVars = manifests, latentVars = latents,
mxPath(from = latents, to = manifests),
mxPath(from = manifests, arrows = 2),
mxPath(from = latents, arrows = 2, free = FALSE, values = 1.0),
mxData(cov(demoOneFactor), type = "cov", numObs = 500)
)
m1 = umxRun(m1, setLabels = TRUE, setValues = TRUE)
residuals(m1)
residuals(m1, digits = 3)
residuals(m1, digits = 3, suppress = .005)
# residuals are returned as an invisible object you can capture in a variable
a = residuals(m1); a
```

RMSEA *Generic RMSEA function*

Description

See RMSEA.MxModel to access the RMSEA of MxModels

Usage

```
RMSEA(x, ci.lower, ci.upper, digits)
```

Arguments

x	an object from which to get the RMSEA
ci.lower	the lower CI to compute
ci.upper	the upper CI to compute
digits	digits to show

Value

- RMSEA object containing value (and perhaps a CI)

References

- http://tbates.github.io, https://github.com/tbates/umx, http://openmx.psyc.virginia.edu

See Also

Other Reporting functions: RMSEA.MxModel; RMSEA.summary.mxmodel; coef.MxModel; confint.MxModel; extractAIC.MxModel; logLik.MxModel; plot.MxModel, umxPlot; plot, plot.MxModel.ACE, umxPlotACE; residuals.MxModel; umxCI_boot; umxCI; umxCompare; umxExpCov; umxExpMeans; umxFitIndices; umxStandardizeModel; umxSummary.MxModel; umx_drop_ok

---

`RMSEA.MxModel` *RMSEA function for MxModels*

Description

Compute the confidence interval on RMSEA

Usage

```
## S3 method for class 'MxModel'
RMSEA(x, ci.lower = 0.05, ci.upper = 0.95, digits = 3)
```

Arguments

`x`	an `mxModel` from which to get RMSEA
`ci.lower`	the lower CI to compute
`ci.upper`	the upper CI to compute
`digits`	digits to show (defaults to 3)

Value

- object containing the RMSEA and lower and upper bounds

References

- https://github.com/simsem/semTools/wiki/Functions, https://github.com/tbates/umx

See Also

Other Reporting functions: RMSEA.summary.mxmodel; RMSEA; coef.MxModel; confint.MxModel; extractAIC.MxModel; logLik.MxModel; plot.MxModel, umxPlot; plot, plot.MxModel.ACE, umxPlotACE; residuals.MxModel; umxCI_boot; umxCI; umxCompare; umxExpCov; umxExpMeans; umxFitIndices; umxStandardizeModel; umxSummary.MxModel; umx_drop_ok

Examples

```
require(OpenMx)
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
m1 <- umxRAM("One Factor", data = mxData(cov(demoOneFactor), type = "cov", numObs = 500),
umxPath(latents, to = manifests),
umxPath(var = manifests),
umxPath(var = latents, fixedAt = 1.0)
)
RMSEA(m1)
```

`RMSEA.summary.mxmodel` *RMSEA function for MxModels*

Description

Compute the confidence interval on RMSEA

Usage

```
## S3 method for class 'summary.mxmodel'
RMSEA(x, ci.lower = 0.05, ci.upper = 0.95,
  digits = 3)
```

Arguments

`x`	an `mxModel` summary from which to get RMSEA
`ci.lower`	the lower CI to compute
`ci.upper`	the upper CI to compute
`digits`	digits to show (defaults to 3)

Value

- object containing the RMSEA and lower and upper bounds

References

- https://github.com/simsem/semTools/wiki/Functions, https://github.com/tbates/umx

See Also

Other Reporting functions: `RMSEA.MxModel`; `RMSEA`; `coef.MxModel`; `confint.MxModel`; `extractAIC.MxModel`; `logLik.MxModel`; `plot.MxModel`, `umxPlot`; `plot`, `plot.MxModel.ACE`, `umxPlotACE`; `residuals.MxModel`; `umxCI_boot`; `umxCI`; `umxCompare`; `umxExpCov`; `umxExpMeans`; `umxFitIndices`; `umxStandardizeModel`; `umxSummary.MxModel`; `umx_drop_ok`

Examples

```
require(OpenMx)
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
m1 <- umxRAM("One Factor", data = mxData(cov(demoOneFactor), type = "cov", numObs = 500),
umxPath(latents, to = manifests),
umxPath(var = manifests),
umxPath(var = latents, fixedAt = 1.0)
)
RMSEA(m1)
```

---

umx *Helper Functions for Structural Equation Modelling in OpenMx*

Description

umx allows you to more easily build, run, modify, and report models using OpenMx with code. The core functions are linked below under **See Also**

Details

The functions are organized into families: Have a read of these below, click to explore.

All the functions have explanatory examples, so use the help, even if you think it won't help :-) Have a look, for example at umxRAM

Introductory working examples are below. You can run all demos with demo(umx) When I have a vignette, it will be: vignette("umx", package = "umx")

The development version of umx is github http://github.com/tbates/umx

There is a helpful blog at http://tbates.github.io

To install from github, you need: install.packages("devtools") library("devtools") install_github("tbates/umx") library("umx")

References

- http://www.github.com/tbates/umx

See Also

Other Advanced Helpers: umx_RAM_ordinal_objective

Other Miscellaneous Data Functions: umxHetCor; umxPadAndPruneForDefVars; umx_as_numeric; umx_cont_2_ordinal; umx_cov2raw; umx_lower2full; umx_make_bin_cont_pair_data; umx_merge_CIs; umx_read_lower; umx_residualize; umx_round; umx_scale_wide_twin_data; umx_scale; umx_swap_a_block

Other Miscellaneous File Functions: dl_from_dropbox; umx_move_file; umx_open; umx_rename_file

Other Miscellaneous Functions: umxEval; umxJiggle; umx_APA_pval; umx_add_variances; umx_apply; umx_check_model; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_check;

umx_default_option; umx_explode; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_cores; umx_get_optimizer; umx_has_CIs; umx_has_been_run; umx_has_means; umx_has_square_brackets; umx_is_MxMatrix; umx_is_MxModel; umx_is_RAM; umx_is_cov; umx_is_endogenous; umx_is_exogenous; umx_is_ordered; umx_msg; umx_names; umx_object_as_str; umx_paste_names; umx_print; umx_rename; umx_reorder; umx_rot; umx_set_cores; umx_set_optimizer; umx_string_to_algebra; umx_trim

Other Miscellaneous Stats Functions: reliability; umxCov2cor; umx_cor; umx_means

Other Miscellaneous Utility Functions: qm; umx_check_OS; umx_find_object; umx_grep; umx_pb_note

Other Model Building Functions: umxLabel; umxLatent; umxPath; umxRAM; umxReRun; umxRun; umxThresholdMatrix; umxValues; umx_fix_first_loadings; umx_fix_latents

Other Reporting Functions: umxStandardizeACE; umx_APA_CI; umx_APA_pval; umx_aggregate; umx_print; umx_show; umx_time

Other Twin Modeling Functions: umxACE; umxGxE_window

Other Twin Reporting Functions: umxSummary.MxModel.ACE, umxSummaryACE

Examples

```
require("OpenMx")
require("umx")
data(demoOneFactor)
myData = mxData(cov(demoOneFactor), type = "cov", numObs = nrow(demoOneFactor))
latents = c("G")
manifests = names(demoOneFactor)
m1 <- mxModel("One Factor", type = "RAM",
manifestVars = manifests,
latentVars  = latents,
mxPath(from = latents, to = manifests),
mxPath(from = manifests, arrows = 2),
mxPath(from = latents  , arrows = 2, free = FALSE, values = 1),
myData
)

omxGetParameters(m1) # nb: By default, paths have no labels, and starts of 0

# With \code{link{umxLabel}}, you can easily add informative and predictable labels to each free
# path (works with matrix style as well!) and use \code{link{umxValues}}, to set
# sensible guesses for start values...
m1 = umxLabel(m1)
m1 = umxValues(m1)

# nb: ?mxRAM simplifies model making in several ways. Check it out!

# Re-run omxGetParameters...
omxGetParameters(m1) # Wow! Now your model has informative labels, & better starts

# umxRun the model (calculates saturated models for raw data, & repeats
# if the model is not code green)
m1 = umxRun(m1, setLabels = TRUE, setValues = TRUE) # not needed given we've done this above.
# But you can see how umxRun enables 1-line setup and run
```

```
# Let's get some journal-ready fit information

umxSummary(m1)
umxSummary(m1, show = "std") #also display parameter estimates

# ==================
# = Model updating =
# ==================
# Can we set the loading of X5 on G to zero?
m2 = omxSetParameters(m1, labels = "G_to_x1", values = 0, free = FALSE, name = "no_g_on_X5")
m2 = mxRun(m2)
# Compare the two models
umxCompare(m1, m2)

# Use umxReRun to do the same thing in 1-line
m2 = umxReRun(m1, "G_to_x1", name = "no_effect_of_g_on_X5", comparison = TRUE)

# ================================
# = Get some Confidence intervals =
# ================================

confint(m1, run = TRUE) # lots more to learn about ?confint.MxModel

# And make a Figure it dot format!
# If you have installed GraphViz, the next command will open it for you to see!

# umxPlot(m1, std = TRUE)
# Run this instead if you don't have GraphViz
plot(m1, std = TRUE, dotFilename = NA)
```

---

umx-deprecated *Deprecated. May already stop() code and ask to be updated. May be dropped entirely in future.*

Description

umxSaturated should be replaced with mxRefModels

umx_grep_labels should be replaced with umx_grep

grepSPSS_labels should be replaced with umx_grep

umxStart should be replaced with umxValues

umxTryHard is deprecated: use umxRun instead

genEpi_Jiggle is deprecated: use umxJiggle instead

umxLabels Is deprecated: use umxLabel instead

umxLabels Is deprecated: use umxLabel instead

umxPath is deprecated: Use mxPath and umxLabel instead

umxReportFit is deprecated: use umxSummary instead

umxGetLabels is deprecated: use umxGetParameters instead

stringToMxAlgebra is deprecated: please use umx_string_to_algebra instead

genEpi_EvalQuote is deprecated: please use umxEval instead

umxReportCIs is deprecated: please use umxCI instead

hasSquareBrackets is deprecated: please use umx_has_square_brackets instead

xmuHasSquareBrackets is deprecated: please use umx_has_square_brackets instead

replace umxReportFit with umxSummary

Replace umxGraph_RAM with plot

Replace tryHard with umxRun

Replace genEpi_ReRun with umxReRun

Replace mxStart with umxValues

Replace umxLabeler with umxLabel

Replace standardizeRAM with umxStandardizeModel

Replace genEpi_equate with umxEquate

Replace genEpi_Path with umxPath

Replace genEpiCompare with umxCompare

Replace mxLatent with umxLatent

Change col.as.numeric is deprecated. Please replace with umx_as_numeric

Change cor.prob to umx_cor

Change umx_u_APA_pval to umx_APA_pval

Usage

```
umxSaturated(...)

grepSPSS_labels(...)

umxStart(...)

umxReportFit(...)

col.as.numeric(...)

cor.prob(...)

mxStart(...)

mxLatent(...)

umxReportFit(...)

umxReportCIs(...)
```

Arguments

...	the old function's parameters (now stripped out to avoid telling people how to do it the wrong way :-)

References

- http://tbates.github.io, https://github.com/tbates/umx, http://openmx.psyc.virginia.edu

---

umxACE *umxACE*

Description

Make a 2-group ACE model

Usage

```
umxACE(name = "ACE", selDVs, dzData, mzData, suffix = NULL, dzAr = 0.5,
  dzCr = 1, addStd = TRUE, addCI = TRUE, numObsDZ = NULL,
  numObsMZ = NULL, boundDiag = NULL, weightVar = NULL,
  equateMeans = TRUE, bVector = FALSE, hint = c("none", "left_censored"))
```

Arguments

name	The name of the model (defaults to"ACE")
selDVs	The variables to include from the data
dzData	The DZ dataframe
mzData	The MZ dataframe
suffix	The suffix for twin 1 and twin 2, often "_T" (defaults to NULL) With this, you can omit suffixes from names in SelDV, i.e., just "dep" not c("dep_T1", "dep_T2")
dzAr	The DZ genetic correlation (defaults to .5, set to .25 for dominance model)
dzCr	The DZ genetic correlation (defaults to 1, vary to examine assortative mating)
addStd	Whether to add the algebras to compute a std model (defaults to TRUE)
addCI	Whether to add intervals to compute CIs (defaults to TRUE)
numObsDZ	= Number of DZ twins: Set this if you input covariance data
numObsMZ	= Number of MZ twins: Set this if you input covariance data
boundDiag	= Whether to bound the diagonal of the a, c, and e matrices
weightVar	= If provided, a vector objective will be used to weight the data. (default = NULL)
equateMeans	Whether to equate the means across twins (defaults to TRUE)
bVector	Whether to compute row-wise likelihoods (defaults to FALSE)
hint	An analysis hint. Options include "none", (default) "left_censored". Default does nothing.

Value

- mxModel of subclass mxModel.ACE

References

- http://www.github.com/tbates/umx

See Also

Other Twin Modeling Functions: umxGxE_window; umx, umx-package

Examples

```
# Height, weight, and BMI data from Australian twins.
# The total sample has been subdivided into a young cohort, aged 18-30 years,
# and an older cohort aged 31 and above.
# Cohort 1 Zygosity is coded as follows:
# 1 == MZ females 2 == MZ males 3 == DZ females 4 == DZ males 5 == DZ opposite sex pairs
# tip: ?twinData to learn more about this data set
require(OpenMx)
require(umx)
data(twinData)
tmpTwin <- twinData
names(tmpTwin)
# "fam", "age", "zyg", "part", "wt1", "wt2", "ht1", "ht2", "htwt1", "htwt2", "bmi1", "bmi2"

# Set zygosity to a factor
labList = c("MZFF", "MZMM", "DZFF", "DZMM", "DZOS")
tmpTwin$zyg = factor(tmpTwin$zyg, levels = 1:5, labels = labList)

# Pick the variables
selDVs = c("bmi1", "bmi2") # nb: Can also give base name, (i.e., "bmi") AND set suffix.
# the function will then make the varnames for each twin using this:
# for example. "VarSuffix1" "VarSuffix2"
mzData <- tmpTwin[tmpTwin$zyg %in% "MZFF", selDVs]
dzData <- tmpTwin[tmpTwin$zyg %in% "DZFF", selDVs]
mzData <- mzData[1:200,] # just top 200 so example runs in a couple of secs
dzData <- dzData[1:200,]
m1 = umxACE(selDVs = selDVs, dzData = dzData, mzData = mzData)
m1 = umxRun(m1)
umxSummary(m1)
umxSummaryACE(m1)
## Not run:
plot(m1)

## End(Not run)
# ADE model (DZ correlation set to .25)
m2 = umxACE("ADE", selDVs = selDVs, dzData = dzData, mzData = mzData, dzCr = .25)
m2 = umxRun(m2)
mxCompare(m2, m1) # ADE is better
umxSummary(m2) # nb: though this is ADE, it's labeled ACE
```

```
# ===================
# = Ordinal example =
# ===================
require(OpenMx)
data(twinData)
tmpTwin <- twinData
names(tmpTwin)
# "fam", "age", "zyg", "part", "wt1", "wt2", "ht1", "ht2", "htwt1", "htwt2", "bmi1", "bmi2"

# Set zygosity to a factor
labList = c("MZFF", "MZMM", "DZFF", "DZMM", "DZOS")
tmpTwin$zyg = factor(tmpTwin$zyg, levels = 1:5, labels = labList)

# Cut bmi colum to form ordinal obesity variables
ordDVs = c("obese1", "obese2")
selDVs = c("obese")
obesityLevels = c('normal', 'overweight', 'obese')
cutPoints <- quantile(tmpTwin[, "bmi1"], probs = c(.5, .2), na.rm = TRUE)
tmpTwin$obese1 <- cut(tmpTwin$bmi1, breaks = c(-Inf, cutPoints, Inf), labels = obesityLevels)
tmpTwin$obese2 <- cut(tmpTwin$bmi2, breaks = c(-Inf, cutPoints, Inf), labels = obesityLevels)
# Make the ordinal variables into mxFactors (ensure ordered is TRUE, and require levels)
tmpTwin[, ordDVs] <- mxFactor(tmpTwin[, ordDVs], levels = obesityLevels)
mzData <- tmpTwin[tmpTwin$zyg %in% "MZFF", umx_paste_names(selDVs, "", 1:2)]
dzData <- tmpTwin[tmpTwin$zyg %in% "DZFF", umx_paste_names(selDVs, "", 1:2)]
mzData <- mzData[1:200,] # just top 200 so example runs in a couple of secs
dzData <- dzData[1:200,]
str(mzData)
m1 = umxACE(selDVs = selDVs, dzData = dzData, mzData = mzData, suffix = '')
m1 = mxRun(m1)
umxSummary(m1)
## Not run:
# plot(m1)

## End(Not run)

# ============================================
# = Bivariate continuous and ordinal example =
# ============================================
data(twinData)
tmpTwin <- twinData
selDVs = c("wt", "obese")
# Set zygosity to a factor
labList = c("MZFF", "MZMM", "DZFF", "DZMM", "DZOS")
tmpTwin$zyg = factor(tmpTwin$zyg, levels = 1:5, labels = labList)

# Cut bmi colum to form ordinal obesity variables
ordDVs = c("obese1", "obese2")
obesityLevels = c('normal', 'overweight', 'obese')
cutPoints <- quantile(tmpTwin[, "bmi1"], probs = c(.5, .2), na.rm = TRUE)
tmpTwin$obese1 <- cut(tmpTwin$bmi1, breaks = c(-Inf, cutPoints, Inf), labels = obesityLevels)
tmpTwin$obese2 <- cut(tmpTwin$bmi2, breaks = c(-Inf, cutPoints, Inf), labels = obesityLevels)
# Make the ordinal variables into mxFactors (ensure ordered is TRUE, and require levels)
```

```
tmpTwin[, ordDVs] <- mxFactor(tmpTwin[, ordDVs], levels = obesityLevels)
mzData <- tmpTwin[tmpTwin$zyg %in% "MZFF", umx_paste_names(selDVs, "", 1:2)]
dzData <- tmpTwin[tmpTwin$zyg %in% "DZFF", umx_paste_names(selDVs, "", 1:2)]
mzData <- mzData[1:200,] # just top 200 so example runs in a couple of secs
dzData <- dzData[1:200,]
str(mzData)
m1 = umxACE(selDVs = selDVs, dzData = dzData, mzData = mzData, suffix = '')
m1 = umxRun(m1)
umxSummary(m1)


# ======================================
# = Mixed continuous and binary example =
# ======================================
require(OpenMx)
data(twinData)
tmpTwin <- twinData
labList = c("MZFF", "MZMM", "DZFF", "DZMM", "DZOS")
tmpTwin$zyg = factor(tmpTwin$zyg, levels = 1:5, labels = labList)
# Cut to form category of 20% obese subjects
cutPoints <- quantile(tmpTwin[, "bmi1"], probs = .2, na.rm = TRUE)
obesityLevels = c('normal', 'obese')
tmpTwin$obese1 <- cut(tmpTwin$bmi1, breaks = c(-Inf, cutPoints, Inf), labels = obesityLevels)
tmpTwin$obese2 <- cut(tmpTwin$bmi2, breaks = c(-Inf, cutPoints, Inf), labels = obesityLevels)
# Make the ordinal variables into mxFactors (ensure ordered is TRUE, and require levels)
ordDVs = c("obese1", "obese2")
tmpTwin[, ordDVs] <- mxFactor(tmpTwin[, ordDVs], levels = obesityLevels)
selDVs = c("wt", "obese")
mzData <- tmpTwin[tmpTwin$zyg == "MZFF", umx_paste_names(selDVs, "", 1:2)]
dzData <- tmpTwin[tmpTwin$zyg == "DZFF", umx_paste_names(selDVs, "", 1:2)]
mzData <- mzData[1:200,] # just top 200 so example runs in a couple of secs
dzData <- dzData[1:200,]
str(mzData)
umx_paste_names(selDVs, "", 1:2)
m1 = umxACE(selDVs = selDVs, dzData = dzData, mzData = mzData, suffix = '')
m1 = umxRun(m1)
umxSummary(m1)

# ==================================
# Example with covariance data only =
# ==================================

require(OpenMx)
data(twinData)
tmpTwin <- twinData
labList = c("MZFF", "MZMM", "DZFF", "DZMM", "DZOS")
tmpTwin$zyg = factor(tmpTwin$zyg, levels = 1:5, labels = labList)
selDVs = c("wt1", "wt2")
dz = cov(tmpTwin[tmpTwin$zyg == "MZFF", selDVs], use = "complete")
mz = cov(tmpTwin[tmpTwin$zyg == "DZFF", selDVs], use = "complete")
m1 = umxACE(selDVs= selDVs, dzData=dz, mzData=mz, numObsDZ=nrow(dzData), numObsMZ=nrow(mzData))
m1 = mxRun(m1)
umxSummary(m1)
```

```
## Not run:
plot(m1)

## End(Not run)
```

---

umxAdd1 *umxAdd1*

Description

Add each of a set of paths you provide to the model, returning a table of theire effect on fit

Usage

```
umxAdd1(model, pathList1 = NULL, pathList2 = NULL, arrows = 2, maxP = 1)
```

Arguments

model	an mxModel to alter
pathList1	a list of variables to generate a set of paths
pathList2	an optional second list: IF set paths will be from pathList1 to members of this list
arrows	Make paths with one or two arrows
maxP	The threshold for returning values (defaults to p==1 - all values)

Value

a table of fit changes

References

- http://www.github.com/tbates/umx

See Also

Other Modify or Compare Models: parameters, umxGetParameters; umxDrop1; umxEquate; umxFixAll; umxMI; umxSetParameters; umxUnexplainedCausalNexus

Examples

```
## Not run:
model = umxAdd1(model)

## End(Not run)
```

umxCI *umxCI*

Description

umxCI adds mxCI() calls for all free parameters in a model, runs the CIs, and reports a neat summary.

Usage

```
umxCI(model = NULL, add = TRUE, run = c("no", "yes", "if necessary"),
  showErrorCodes = TRUE)
```

Arguments

model	The mxModel you wish to report mxCIs on
add	Whether or not to add mxCIs if none are found (defaults to TRUE)
run	Whether or not to compute the CIs. Valid values = "no" (default), "yes", "if necessary".
showErrorCodes	Whether to show errors (default == TRUE)

Details

This function also reports any problems computing a CI. The codes are standard OpenMx errors and warnings

- 1: The final iterate satisfies the optimality conditions to the accuracy requested, but the sequence of iterates has not yet converged. NPSOL was terminated because no further improvement could be made in the merit function (Mx status GREEN)
- 2: The linear constraints and bounds could not be satisfied. The problem has no feasible solution.
- 3: The nonlinear constraints and bounds could not be satisfied. The problem may have no feasible solution.
- 4: The major iteration limit was reached (Mx status BLUE).
- 6: The model does not satisfy the first-order optimality conditions to the required accuracy, and no improved point for the merit function could be found during the final linesearch (Mx status RED)
- 7: The function derivates returned by funcon or funobj appear to be incorrect.
- 9: An input parameter was invalid

If runCIs is FALSE, the function simply adds CIs to be computed and returns the model.

Value

- mxModel

References

- http://www.github.com/tbates/umx/

See Also

- mxCI, umxLabel, umxRun

Other Reporting functions: RMSEA.MxModel; RMSEA.summary.mxmodel; RMSEA; coef.MxModel; confint.MxModel; extractAIC.MxModel; logLik.MxModel; plot.MxModel, umxPlot; plot, plot.MxModel.ACE, umxPlotACE; residuals.MxModel; umxCI_boot; umxCompare; umxExpCov; umxExpMeans; umxFitIndices; umxStandardizeModel; umxSummary.MxModel; umx_drop_ok

Examples

```
require(OpenMx)
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
m1 <- mxModel("One Factor", type = "RAM",
manifestVars = manifests, latentVars = latents,
mxPath(from = latents, to = manifests),
mxPath(from = manifests, arrows = 2),
mxPath(from = latents, arrows = 2, free = FALSE, values = 1.0),
mxData(cov(demoOneFactor), type = "cov", numObs = 500)
)
m1 = umxRun(m1, setLabels = TRUE, setValues = TRUE)
m1$intervals # none yet list()
m1 = umxCI(m1)
m1$intervals # $G_to_x1
m1 = umxCI(m1, add = TRUE) # Add CIs for all free parameters, and return model
## Not run:
umxCI(model, run = "yes") # force update of CIs
# Don't force update of CIs, but if they were just added, then calculate them
umxCI(model, run = "if necessary")

## End(Not run)
```

---

umxCI_boot *umxCI_boot*

Description

Compute boot-strapped Confidence Intervals for parameters in an mxModel The function creates a sampling distribution for parameters by repeatedly drawing samples with replacement from your data and then computing the statistic for each redrawn sample.

Usage

```
umxCI_boot(model, rawData = NULL, type = c("par.expected", "par.observed",
  "empirical"), std = TRUE, rep = 1000, conf = 95, dat = FALSE,
  digits = 3)
```

Arguments

model	is an optimized mxModel
rawData	is the raw data matrix used to estimate model
type	is the kind of bootstrap you want to run. "par.expected" and "par.observed" use parametric Monte Carlo bootstrapping based on your expected and observed covariance matrices, respectively. "empirical" uses empirical bootstrapping based on rawData.
std	specifies whether you want CIs for unstandardized or standardized parameters (default: std = TRUE)
rep	is the number of bootstrap samples to compute (default = 1000).
conf	is the confidence value (default = 95)
dat	specifies whether you want to store the bootstrapped data in the output (useful for multiple analyses, such as mediation analysis)
digits	rounding precision

Value

- expected covariance matrix

References

- http://openmx.psyc.virginia.edu/thread/2598 Original written by http://openmx.psyc.virginia.edu/users/bwiernik

See Also

- umxExpMeans, umxExpCov

Other Reporting functions: RMSEA.MxModel; RMSEA.summary.mxmodel; RMSEA; coef.MxModel; confint.MxModel; extractAIC.MxModel; logLik.MxModel; plot.MxModel, umxPlot; plot, plot.MxModel.ACE, umxPlotACE; residuals.MxModel; umxCI; umxCompare; umxExpCov; umxExpMeans; umxFitIndices; umxStandardizeModel; umxSummary.MxModel; umx_drop_ok

Examples

```
## Not run:
require(OpenMx)
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
m1 <- mxModel("One Factor", type = "RAM",
manifestVars = manifests, latentVars = latents,
mxPath(from = latents, to = manifests),
mxPath(from = manifests, arrows = 2),
mxPath(from = latents, arrows = 2, free = FALSE, values = 1.0),
mxData(cov(demoOneFactor), type = "cov", numObs = 500)
)
m1 = umxRun(m1, setLabels = TRUE, setValues = TRUE)
```

```
umxCI_boot(m1, type = "par.expected")

## End(Not run)
```

---

umxCompare *umxCompare*

Description

umxCompare compares two or more mxModels. If you leave comparison blank, it will just give fit info for the base model

Usage

```
umxCompare(base = NULL, comparison = NULL, all = TRUE, digits = 3,
  report = 1, file = "tmp.html")
```

Arguments

base	The base mxModel for comparison
comparison	The model (or list of models) which will be compared for fit with the base model (can be empty)
all	Whether to make all possible comparisons if there is more than one base model (defaults to T)
digits	rounding for p etc.
report	Optionally add sentences for inclusion inline in a paper (report= 2) and output to an html table which will open your default browser (report = 3). (This is handy for getting tables into Word, markdown, and other text systems!)
file	file to write html too if report=3 (defaults to "tmp.html")

References

- http://www.github.com/tbates/umx/

See Also

- mxCompare, umxSummary, umxRun,

Other Reporting functions: RMSEA.MxModel; RMSEA.summary.mxmodel; RMSEA; coef.MxModel; confint.MxModel; extractAIC.MxModel; logLik.MxModel; plot.MxModel, umxPlot; plot, plot.MxModel.ACE, umxPlotACE; residuals.MxModel; umxCI_boot; umxCI; umxExpCov; umxExpMeans; umxFitIndices; umxStandardizeModel; umxSummary.MxModel; umx_drop_ok

Examples

```
require(OpenMx)
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
m1 <- mxModel("One Factor", type = "RAM",
manifestVars = manifests, latentVars = latents,
mxPath(from = latents, to = manifests),
mxPath(from = manifests, arrows = 2),
mxPath(from = latents, arrows = 2, free = FALSE, values = 1.0),
mxData(cov(demoOneFactor), type = "cov", numObs = 500)
)
m1 = umxRun(m1, setLabels = TRUE, setValues = TRUE)
m2 = umxReRun(m1, update = "G_to_x2", name = "drop_path_2_x2")
umxCompare(m1, m2)
mxCompare(m1, m2) # what OpenMx gives by default
umxCompare(m1, m2, report = 2) # Add English-sentence descriptions
## Not run:
umxCompare(m1, m2, report = 3) # Open table in browser

## End(Not run)
m3 = umxReRun(m2, update = "G_to_x3", name = "drop_path_2_x2_and_3")
umxCompare(m1, c(m2, m3))
umxCompare(c(m1, m2), c(m2, m3), all = TRUE)
```

---

umxCov2cor *umxCov2cor*

Description

Version of cov2cor that forces upper and lower triangles to be identical (rather than nearly identical)

Usage

```
umxCov2cor(x)
```

Arguments

x	something that cov2cor can work on (matrix, df, etc.)

Value

- a correlation matrix

References

- http://www.github.com/tbates/umx

See Also

- umxLabel, umxRun, umxValues

Other Miscellaneous Stats Functions: reliability; umx_cor; umx_means; umx, umx-package

Examples

```
umxCov2cor(cov(mtcars))
```

---

umxCovData *umxCovData*

Description

convert a dataframe in an mxData object of type "cov"

Usage

```
umxCovData(df, columns = NA, use = c("complete.obs", "everything",
  "all.obs", "na.or.complete", "pairwise.complete.obs"))
```

Arguments

df	the dataframe to covert to a covariance matrix
columns	= manifests
use	= Default is "complete.obs"

Value

- mxModel

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

- umxHetCor

Examples

```
umxCovData(mtcars, c("mpg", "hp"))
```

umxDiagnose *mxDiagnostic*

Description

Diagnose problems in a model

Usage

```
umxDiagnose(model, tryHard = FALSE, diagonalizeExpCov = FALSE)
```

Arguments

model	an mxModel to diagnose
tryHard	whether I should try and fix it? (defaults to FALSE)
diagonalizeExpCov	Whether to diagonalize the ExpCov

Value

- helpful messages and perhaps a modified model

References

- http://tbates.github.io, https://github.com/tbates/umx

Examples

```
require(OpenMx)
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
myData = mxData(cov(demoOneFactor), type = "cov", numObs = 500)
m1 <- umxRAM("OneFactor", data = myData,
umxPath(latents, to = manifests),
umxPath(var = manifests),
umxPath(var = latents, fixedAt = 1.0)
)
m1 = mxRun(m1)
umxSummary(m1, show = "std")
umxDiagnose(m1)
```

umxDrop1 *umxDrop1*

Description

Drops each free parameter (selected via regex), returning an mxCompare table comparing the effects. A great way to quickly determine which of several parameters can be dropped without excessive cost

Usage

```
umxDrop1(model, regex = NULL, maxP = 1)
```

Arguments

model	An mxModel to drop parameters from
regex	A string to select parameters to drop. leave empty to try all. This is regular expression enabled. i.e., "^a_" will drop parameters beginning with "a_"
maxP	The threshold for returning values (defaults to p==1 - all values)

Value

a table of model comparisons

References

- http://www.github.com/tbates/umx

See Also

Other Modify or Compare Models: parameters, umxGetParameters; umxAdd1; umxEquate; umxFixAll; umxMI; umxSetParameters; umxUnexplainedCausalNexus

Examples

```
## Not run:
umxDrop1(fit3) # try dropping each free parameters (default)
# drop "a_r1c1" and "a_r1c2" and see which matters more.
umxDrop1(model, regex="a_r1c1|a_r1c2")

## End(Not run)
```

umxEFA *umxEFA*

Description

A helper for EFA that only requires you to enter your latents and manifests

Usage

```
umxEFA(name = "", latents, data, report = c("table", "line", "long"))
```

Arguments

name	the name for your new EFA model
latents	List of factors in your CFA
data	The dataframe of manifest columns you are modeling
report	What to report

Value

- mxModel

References

- http://github.com/tbates/umx

See Also

- umxLabel, umxRun, umxValues

Examples

```
## Not run:
umxEFA("test", latents = "g", data = mtcars[, c("mpg", "disp", "hp", "wt")])

## End(Not run)
```

umxEquate *umxEquate*

Description

Equate parameters by setting one or more labels (the slave set) equal to the labels in a master set. Setting two or more parameters to have the same umxLabel constrains them to take the same value.

Usage

```
umxEquate(model, master, slave, free = c(TRUE, FALSE, NA), verbose = TRUE,
  name = NULL)
```

Arguments

model	An mxModel within which to equate parameters
master	A list of "master" labels to which slave labels will be equated
slave	A list of slave labels which will be updated to match master labels, thus equating the parameters
free	Should parameter(s) initally be free? (default = TRUE)
verbose	Whether to give verbose feedback (default = TRUE)
name	name for the returned model (optional: Leave empty to leave name unchanged)

Details

note: In addition to using this method to equating parameters, you can also equate one parameter to another by setting its label to the "square bracket" address of the master, e.g. "a[r,c]".

Tip: To find labels of free parameters use umxGetParameters with free = T Tip: To find labels by name, use the regex parameter of umxGetParameters

Value

- mxModel

References

- http://www.github.com/tbates/umx

See Also

Other Modify or Compare Models: parameters, umxGetParameters; umxAdd1; umxDrop1; umxFixAll; umxMI; umxSetParameters; umxUnexplainedCausalNexus

Examples

```
require(OpenMx)
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
m1 <- mxModel("One Factor", type = "RAM",
manifestVars = manifests, latentVars = latents,
mxPath(from = latents, to = manifests),
mxPath(from = manifests, arrows = 2),
mxPath(from = latents, arrows = 2, free = FALSE, values = 1.0),
mxData(cov(demoOneFactor), type = "cov", numObs = 500)
)
m1 = umxRun(m1, setLabels = TRUE, setValues = TRUE)
m2 = umxEquate(m1, master = "G_to_x1", slave = "G_to_x2", name = "Equate x1 and x2 loadings")
m2 = mxRun(m2) # have to run the model again...
umxCompare(m1, m2) # not good :-)
```

`umxEval` *umxEval*

Description

Takes an expression as a string, and evaluates it as an expression in model, optionally computing the result. # TODO Currently broken...

Usage

```
umxEval(expstring, model, compute = FALSE, show = FALSE)
```

Arguments

`expstring`	an expression string, i.e, "a + b"
`model`	an `mxModel` to evaluate in
`compute`	Whether to compute the result or not (default = FALSE)
`show`	Whether to show??? (default = FALSE)

Value

- an openmx algebra (formula)

References

- http://www.github.com/tbates/umx

See Also

- umxLabel, umxRun, umxValues

Other Miscellaneous Functions: umxJiggle; umx_APA_pval; umx_add_variances; umx_apply; umx_check_model; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_check; umx_default_option; umx_explode; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_cores; umx_get_optimizer; umx_has_CIs; umx_has_been_run; umx_has_means; umx_has_square_brackets; umx_is_MxMatrix; umx_is_MxModel; umx_is_RAM; umx_is_cov; umx_is_endogenous; umx_is_exogenous; umx_is_ordered; umx_msg; umx_names; umx_object_as_str; umx_paste_names; umx_print; umx_rename; umx_reorder; umx_rot; umx_set_cores; umx_set_optimizer; umx_string_to_algebra; umx_trim; umx, umx-package

Examples

```
m1 = mxModel("fit",
mxMatrix("Full", nrow = 1, ncol = 1, free = TRUE, values = 1, name = "a"),
mxMatrix("Full", nrow = 1, ncol = 1, free = TRUE, values = 2, name = "b"),
mxAlgebra(a %*% b, name = "ab"),
mxConstraint(ab == 35, name = "maxHours"),
mxAlgebraObjective(algebra = "ab", numObs= NA, numStats = NA)
)
m1 = mxRun(m1)
mxEval(list(ab = ab), m1)
```

---

umxExpCov *umxExpCov*

Description

extract the expected covariance matrix from an mxModel

Usage

```
umxExpCov(model, latents = FALSE, manifests = TRUE, digits = NULL)
```

Arguments

model	an mxModel to get the covariance matrix from
latents	Whether to select the latent variables (defaults to TRUE)
manifests	Whether to select the manifest variables (defaults to TRUE)
digits	precision of reporting. Leave NULL to do no rounding.

Value

- expected covariance matrix

References

- http://openmx.psyc.virginia.edu/thread/2598 Original written by http://openmx.psyc.virginia.edu/users/bwiernik

See Also

- umxRun, umxCI_boot

Other Reporting functions: RMSEA.MxModel; RMSEA.summary.mxmodel; RMSEA; coef.MxModel; confint.MxModel; extractAIC.MxModel; logLik.MxModel; plot.MxModel, umxPlot; plot, plot.MxModel.ACE, umxPlotACE; residuals.MxModel; umxCI_boot; umxCI; umxCompare; umxExpMeans; umxFitIndices; umxStandardizeModel; umxSummary.MxModel; umx_drop_ok

Examples

```
require(OpenMx)
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
m1 <- mxModel("One Factor", type = "RAM",
manifestVars = manifests, latentVars = latents,
mxPath(from = latents, to = manifests),
mxPath(from = manifests, arrows = 2),
mxPath(from = latents, arrows = 2, free = FALSE, values = 1.0),
mxData(cov(demoOneFactor), type = "cov", numObs = 500)
)
m1 = umxRun(m1, setLabels = TRUE, setValues = TRUE)
umxExpCov(m1)
umxExpCov(m1, digits = 3)
```

---

umxExpMeans *umxExpMean*

Description

Extract the expected means matrix from an mxModel

Usage

```
umxExpMeans(model, manifests = TRUE, latents = NULL, digits = NULL)
```

Arguments

model	an mxModel to get the means from
manifests	Whether to select the manifest variables (defaults to TRUE)
latents	Whether to select the latent variables (defaults to TRUE)
digits	precision of reporting. Leave NULL to do no rounding.

Value

- expected means

References

- http://openmx.psyc.virginia.edu/thread/2598

See Also

Other Reporting functions: RMSEA.MxModel; RMSEA.summary.mxmodel; RMSEA; coef.MxModel; confint.MxModel; extractAIC.MxModel; logLik.MxModel; plot.MxModel, umxPlot; plot, plot.MxModel.ACE, umxPlotACE; residuals.MxModel; umxCI_boot; umxCI; umxCompare; umxExpCov; umxFitIndices; umxStandardizeModel; umxSummary.MxModel; umx_drop_ok

Examples

```
require(OpenMx)
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
m1 <- mxModel("One Factor", type = "RAM",
manifestVars = manifests, latentVars = latents,
mxPath(from = latents, to = manifests),
mxPath(from = manifests, arrows = 2),
mxPath(from = "one", to = manifests),
mxPath(from = latents, arrows = 2, free = FALSE, values = 1.0),
mxData(demoOneFactor, type = "raw")
)
m1 = umxRun(m1, setLabels = TRUE, setValues = TRUE)
umxExpMeans(model = m1)
umxExpMeans(m1, digits = 3)
```

---

umxFitIndices *umxFitIndices*

Description

A list of fit indices

Usage

```
umxFitIndices(model, indepfit)
```

Arguments

model	the mxModel you want fit indices for
indepfit	an (optional) saturated mxModel

Value

NULL

References

- http://www.github.com/tbates/umx

See Also

Other Reporting functions: RMSEA.MxModel; RMSEA.summary.mxmodel; RMSEA; coef.MxModel; confint.MxModel; extractAIC.MxModel; logLik.MxModel; plot.MxModel, umxPlot; plot, plot.MxModel.ACE, umxPlotACE; residuals.MxModel; umxCI_boot; umxCI; umxCompare; umxExpCov; umxExpMeans; umxStandardizeModel; umxSummary.MxModel; umx_drop_ok

Examples

```
require(OpenMx)
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
m1 <- mxModel("One Factor", type = "RAM",
manifestVars = manifests, latentVars = latents,
mxPath(from = latents, to = manifests),
mxPath(from = manifests, arrows = 2),
mxPath(from = latents, arrows = 2, free = FALSE, values = 1.0),
mxData(cov(demoOneFactor), type = "cov", numObs = 500)
)
m1 = umxRun(m1, setLabels = TRUE, setValues = TRUE)
## Not run:
umxFitIndices(m1)

## End(Not run)
```

---

umxFixAll *umxFixAll*

Description

Fix all free parameters in a model using omxGetParameters()

Usage

```
umxFixAll(model, name = "_fixed", run = FALSE, verbose = FALSE)
```

Arguments

model	an mxModel within which to fix free parameters
name	optional new name for the model. if you begin with a _ it will be made a suffix
run	whether to fix and re-run the model, or just return it (defaults to FALSE)
verbose	whether to mention how many paths were fixed (default is FALSE)

Value

- the fixed mxModel

References

- http://tbates.github.io, https://github.com/tbates/umx

See Also

Other Modify or Compare Models: parameters, umxGetParameters; umxAdd1; umxDrop1; umxEquate; umxMI; umxSetParameters; umxUnexplainedCausalNexus

Examples

```
require(OpenMx)
data(demoOneFactor)
latents = c("G")
manifests = names(demoOneFactor)
m1 <- umxRAM("OneFactor", data = mxData(cov(demoOneFactor), type = "cov", numObs = 500),
umxPath(latents, to = manifests),
umxPath(var = manifests),
umxPath(var = latents, fixedAt = 1)
)
m1 = mxRun(m1)
m2 = umxFixAll(m1, run = TRUE, verbose = TRUE)
mxCompare(m1, m2)
```

---

umxGetParameters *umxGetParameters*

Description

Get the parameter labels from a model. Like omxGetParameters, but supercharged with regular expressions for more power and ease!

Usage

```
umxGetParameters(inputTarget, regex = NA, free = NA, verbose = FALSE)

parameters(inputTarget, regex = NA, free = NA, verbose = FALSE)
```

Arguments

inputTarget	An object to get parameters from: could be a RAM mxModel
regex	A regular expression to filter the labels defaults to NA - just returns all labels)
free	A Boolean determining whether to return only free parameters.
verbose	How much feedback to give

References

- http://www.github.com/tbates/umx

See Also

Other Modify or Compare Models: umxAdd1; umxDrop1; umxEquate; umxFixAll; umxMI; umxSetParameters; umxUnexplainedCausalNexus

Examples

```
require(OpenMx)
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
m1 <- mxModel("One Factor", type = "RAM",
manifestVars = manifests, latentVars = latents,
mxPath(from = latents, to = manifests),
mxPath(from = manifests, arrows = 2),
mxPath(from = latents, arrows = 2, free = FALSE, values = 1.0),
mxData(cov(demoOneFactor), type = "cov", numObs = 500)
)
m1 = umxRun(m1)
umxGetParameters(m1)
m1 = umxRun(m1, setLabels = TRUE)
umxGetParameters(m1)
umxGetParameters(m1, free = TRUE) # only the free parameter
umxGetParameters(m1, free = FALSE) # only parameters which are fixed
## Not run:
# Complex regex patterns
umxGetParameters(m2, regex = "S_r_[0-9]c_6", free = TRUE) # Column 6 of matrix "as"

## End(Not run)
```

---

umxGxE_window *umxGxE_window*

Description

Makes a model to do a GxE analysis using Local SEM (Hildebrandt, Wilhelm & Robitzsch, 2009, p96) Local SEM GxE relies on weighting the moderator to allow conducting repeated regular ACE analyses targeted at sucessive regions of the moderator. In this sense, you can think of it as non-parametric GxE

Usage

```
umxGxE_window(selDVs = NULL, moderator = NULL, mzData = mzData,
  dzData = dzData, weightCov = FALSE, target = NULL, width = 1,
  plotWindow = FALSE, return = c("estimates", "last_model"))
```

Arguments

selDVs	The dependent variables for T1 and T2, e.g. c("bmi_T1", "bmi_T2")
moderator	The name of the moderator variable in the dataset e.g. "age", "SES" etc.
mzData	Dataframe containing the DV and moderator for MZ twins
dzData	Dataframe containing the DV and moderator for DZ twins
weightCov	Whether to use cov.wt matrices or FIML default = FALSE, i.e., FIML
target	A user-selected list of moderator values to test (default = NULL = explore the full range)
width	An option to widen or narrow the window from its default (of 1)
plotWindow	whether to plot what the window looks like
return	whether to return the last model (useful for specifiedTargets) or the list of estimates (default = "estimates")

Value

- Table of estimates of ACE along the moderator

References

- Hildebrandt, A., Wilhelm, O, & Robitzsch, A. (2009) Complementary and competing factor analytic approaches for the investigation of measurement invariance. *Review of Psychology*, **16**, 87–107.

Briley, D.A., Harden, K.P., Bates, T.C., Tucker-Drob, E.M. (2015). Nonparametric Estimates of Gene x Environment Interaction Using Local Structural Equation Modeling. *Behavior Genetics*.

See Also

Other Twin Modeling Functions: umxACE; umx, umx-package

Examples

```
library(OpenMx);
# ==============================
# = 1. Open and clean the data =
# ==============================
# umxGxE_window takes a dataframe consisting of a moderator and two DV columns: one for each twin
mod = "age"         # The name of the moderator column in the dataset
selDVs = c("bmi1", "bmi2") # The DV for twin 1 and twin 2
data(twinData) # Dataset of Australian twins, built into OpenMx
# The twinData consist of two cohorts. First we label them
# TODO: Q for OpenMx team: can I add a cohort column to this dataset?
twinData$cohort = 1; twinData$cohort[twinData$zyg %in% 6:10] = 2
twinData$zyg[twinData$cohort == 2] = twinData$zyg[twinData$cohort == 2]-5
# And set a plain-English label
labList = c("MZFF", "MZMM", "DZFF", "DZMM", "DZOS")
twinData$ZYG = factor(twinData$zyg, levels = 1:5, labels = labList)
# The model also assumes two groups: MZ and DZ. Moderator can't be missing
```

```
# Delete missing moderator rows
twinData = twinData[!is.na(twinData[mod]),]
mzData = subset(twinData, ZYG == "MZFF", c(selDVs, mod))
dzData = subset(twinData, ZYG == "DZFF", c(selDVs, mod))

# ========================
# = 2. Run the analyses! =
# ========================
# Run and plot for specified windows (in this case just 1927)
umxGxE_window(selDVs = selDVs, moderator = mod, mzData = mzData, dzData = dzData,
target = 40, plotWindow = TRUE)

## Not run:
# Run with FIML (default) uses all information
umxGxE_window(selDVs = selDVs, moderator = mod, mzData = mzData, dzData = dzData);

# Run creating weighted covariance matrices (excludes missing data)
umxGxE_window(selDVs = selDVs, moderator = mod, mzData = mzData, dzData = dzData,
weightCov = TRUE);

## End(Not run)
```

---

umxHetCor *umxHetCor*

Description

umxHetCor Helper to return just the correlations from John Fox's polycor::hetcor function

Usage

```
umxHetCor(data, ML = FALSE, use = "complete.obs",
  treatAllAsFactor = FALSE, verbose = FALSE)
```

Arguments

data	A data.frame of columns for which to compute heterochoric correlations
ML	Whether to use Maximum likelihood computation of correlations (default = FALSE)
use	How to handle missing data: "complete.obs", "pairwise.complete.obs" Default is complete.obs
treatAllAsFactor	Whether to treat all columns as factors, whether they are or not.
verbose	How much to tell the user about what was done.

Value

- A matrix of correlations

References

-

See Also

- hetcor

Other Miscellaneous Data Functions: umxPadAndPruneForDefVars; umx_as_numeric; umx_cont_2_ordinal; umx_cov2raw; umx_lower2full; umx_make_bin_cont_pair_data; umx_merge_CIs; umx_read_lower; umx_residualize; umx_round; umx_scale_wide_twin_data; umx_scale; umx_swap_a_block; umx, umx-package

Examples

```
umxHetCor(mtcars[,c("mpg", "am")])
umxHetCor(mtcars[,c("mpg", "am")], treatAllAsFactor = FALSE, verbose = TRUE)
```

---

umxJiggle *umxJiggle*

Description

umxJiggle takes values in a matrix and jiggles them

Usage

```
umxJiggle(matrixIn, mean = 0, sd = 0.1, dontTouch = 0)
```

Arguments

matrixIn	an mxMatrix to jiggle the values of
mean	the mean value to add to each value
sd	the sd of the jiggle noise
dontTouch	A value, which, if found, will be left as-is (defaults to 0)

Value

- mxMatrix

References

- http://www.github.com/tbates/umx

See Also

Other Miscellaneous Functions: umxEval; umx_APA_pval; umx_add_variances; umx_apply; umx_check_model; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_check; umx_default_option; umx_explode; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_cores; umx_get_optimizer; umx_has_CIs; umx_has_been_run; umx_has_means; umx_has_square_brackets; umx_is_MxMatrix; umx_is_MxModel; umx_is_RAM; umx_is_cov; umx_is_endogenous; umx_is_exogenous; umx_is_ordered; umx_msg; umx_names; umx_object_as_str; umx_paste_names; umx_print; umx_rename; umx_reorder; umx_rot; umx_set_cores; umx_set_optimizer; umx_string_to_algebra; umx_trim; umx, umx-package

Examples

```
## Not run:
mat1 = umxJiggle(mat1)

## End(Not run)
```

---

umxLabel *umxLabel*

Description

umxLabel adds labels to things, be it an: mxModel (RAM or matrix based), an mxPath, or an mxMatrix This is a core function in umx: Adding labels to paths opens the door to umxEquate, as well as omxSetParameters

Usage

```
umxLabel(obj, suffix = "", baseName = NA, setfree = FALSE, drop = 0,
  labelFixedCells = TRUE, jiggle = NA, boundDiag = NA, verbose = FALSE,
  overRideExisting = FALSE)
```

Arguments

obj	An mxModel (RAM or matrix based), mxPath, or mxMatrix
suffix	String to append to each label (might be used to distinguish, say male and female submodels in a model)
baseName	String to prepend to labels. Defaults to NA ("")
setfree	Whether to label only the free paths (defaults to FALSE)
drop	The value to fix "drop" paths to (defaults to 0)
labelFixedCells	= TRUE
jiggle	How much to jiggle values in a matrix or list of path values
boundDiag	Whether to bound the diagonal of a matrix
verbose	How much feedback to give the user (default = FALSE)
overRideExisting	= FALSE

Value

- mxModel

References

- http://www.github.com/tbates/umx

See Also

Other Model Building Functions: umxLatent; umxPath; umxRAM; umxReRun; umxRun; umxThresholdMatrix; umxValues; umx_fix_first_loadings; umx_fix_latents; umx, umx-package

Examples

```
require(OpenMx)
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
m1 <- mxModel("One Factor", type = "RAM",
manifestVars = manifests, latentVars = latents,
mxPath(from = latents, to = manifests),
mxPath(from = manifests, arrows = 2),
mxPath(from = latents, arrows = 2, free = FALSE, values = 1),
mxData(cov(demoOneFactor), type = "cov", numObs = 500)
)
umxGetParameters(m1) # Default "matrix address" labels, i.e "One Factor.S[2,2]"
m1 = umxLabel(m1)
umxGetParameters(m1, free = TRUE) # Informative labels: "G_to_x1", "x4_with_x4", etc.
# Labeling a matrix
a = umxLabel(mxMatrix(name = "a", "Full", 3, 3, values = 1:9))
a$labels
# labels with "data." in the name are left alone
a = mxMatrix(name = "a", "Full", 1,3, labels = c("data.a", "test", NA))
umxLabel(a, verbose = TRUE)
umxLabel(a, verbose = TRUE, overRideExisting = FALSE)
umxLabel(a, verbose = TRUE, overRideExisting = TRUE)
umxLabel(a, verbose = TRUE, overRideExisting = TRUE)
```

---

umxLatent *umxLatent*

Description

Helper to ease the creation of latent variables including formative and reflective variables (see below) For formative variables, the manifests define (form) the latent. This function takes care of intercorrelating manifests for formatives, and fixing variances correctly

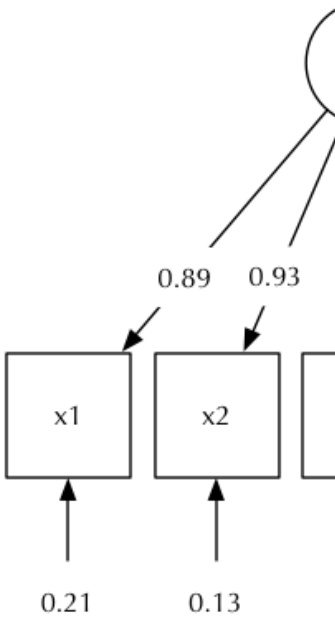
Usage

```
umxLatent(latent = NULL, formedBy = NULL, forms = NULL, data = NULL,
  type = NULL, name = NULL, labelSuffix = "", verbose = TRUE,
  endogenous = "deprecated")
```

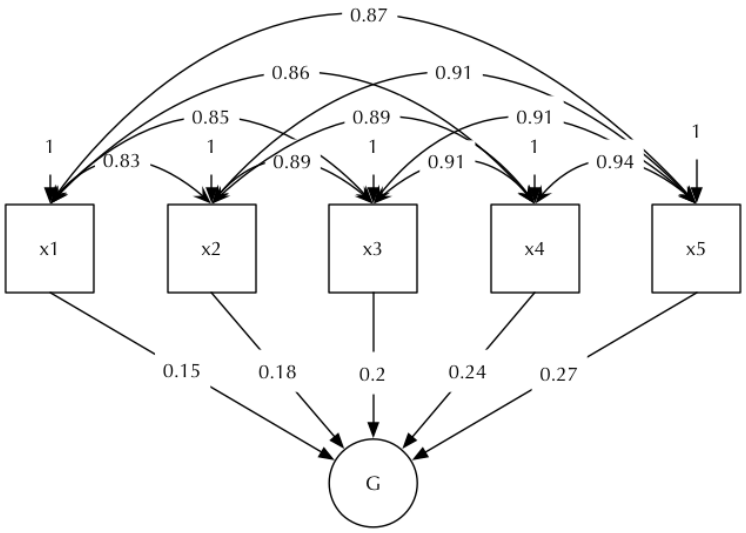
Arguments

latent	the name of the latent variable (string)
formedBy	the list of variables forming this latent
forms	the list of variables which this latent forms (leave blank for formedBy)
data	the dataframe being used in this model
type	= \"exogenous|endogenous\"
name	A name for the path NULL
labelSuffix	a suffix string to append to each label
verbose	Default is TRUE as this function does quite a lot
endogenous	This is now deprecated. use type= \"exogenous|endogenous\"

Details



The following figures show how a reflective and a formative variable look as path diagrams:



formative

Value

- path list

References

- http://www.github.com/tbates/umx

See Also

Other Model Building Functions: umxLabel; umxPath; umxRAM; umxReRun; umxRun; umxThresholdMatrix; umxValues; umx_fix_first_loadings; umx_fix_latents; umx, umx-package

Examples

```
## Not run:
library(OpenMx)
library(umx)
data(demoOneFactor)
latents = c("G")
manifests = names(demoOneFactor) # x1-5
theData = cov(demoOneFactor)
m1 = mxModel("reflective", type = "RAM",
manifestVars = manifests,
latentVars   = latents,
# Factor loadings
umxLatent("G", forms = manifests, type = "exogenous", data = theData),
mxData(theData, type = "cov", numObs = nrow(demoOneFactor))
)
m1 = umxRun(m1, setValues = TRUE, setLabels = TRUE); umxSummary(m1, show="std")
plot(m1)

m2 = mxModel("formative", type = "RAM",
manifestVars = manifests,
latentVars   = latents,
# Factor loadings
umxLatent("G", formedBy = manifests, data = theData),
mxData(theData, type = "cov", numObs = nrow(demoOneFactor))
)
m2 = umxRun(m2, setValues = TRUE, setLabels = TRUE);
umxSummary(m2, show = "std")
plot(m2)

## End(Not run)
```

---

umxMI *umxMI*

Description

Report modifications which would improve fit. Notes: 1. Runs much fast with full = FALSE (but this doesn't allow the model to re-fit around the newly- freed parameter). 2. Compared to mxMI, this function returns top changes, and also suppresses the run message. 3. Finally, of course: see the requirements for (legitimate) post-hoc modeling in mxMI You are almost certainly doing better science when testing competing models rather than modifying a model to fit.

Usage

```
umxMI(model = NA, matrices = NA, full = TRUE, numInd = NA,
  typeToShow = "both", decreasing = TRUE)
```

Arguments

model	An mxModel for which to report modification indices
matrices	which matrices to test. The default (NA) will test A & S for RAM models
full	Change in fit allowing all parameters to move. If FALSE only the parameter under test can move.
numInd	How many modifications to report. Use -1 for all. Default (NA) will report all over 6.63 (p = .01)
typeToShow	Whether to shown additions or deletions (default = "both")
decreasing	How to sort (default = TRUE, decreasing)

References

- http://www.github.com/tbates/umx

See Also

- mxMI

Other Modify or Compare Models: parameters, umxGetParameters; umxAdd1; umxDrop1; umxEquate; umxFixAll; umxSetParameters; umxUnexplainedCausalNexus

Examples

```
## Not run:
require(OpenMx)
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
m1 <- mxModel("One Factor", type = "RAM",
manifestVars = manifests, latentVars = latents,
mxPath(from = latents, to = manifests),
mxPath(from = manifests, arrows = 2),
mxPath(from = latents, arrows = 2, free = FALSE, values = 1.0),
mxData(cov(demoOneFactor), type = "cov", numObs = 500)
)
m1 = umxRun(m1, setLabels = TRUE, setValues = TRUE)
umxMI(model)
umxMI(model, numInd=5, typeToShow="add") # valid options are "both|add|delete"

## End(Not run)
```

umxPadAndPruneForDefVars *umxPadAndPruneForDefVars*

Description

Replaces NAs in definition slots with the mean for that variable ONLY where all data are missing for that twin

Usage

```
umxPadAndPruneForDefVars(df, varNames, defNames, suffixes, highDefValue = 99,
  rm = c("drop_missing_def", "pad_with_mean"))
```

Arguments

df	the dataframe to process
varNames	list of names of the variables being analysed
defNames	list of covariates
suffixes	suffixes that map names on columns in df (i.e., c("T1", "T2"))
highDefValue	What to replace missing definition variables (covariates) with. Default = 99
rm	= how to handle missing values in the varNames. Default is "drop_missing_def", "pad_with_mean")

Value

- dataframes

References

- http://tbates.github.io, https://github.com/tbates/umx

See Also

Other Miscellaneous Data Functions: umxHetCor; umx_as_numeric; umx_cont_2_ordinal; umx_cov2raw; umx_lower2full; umx_make_bin_cont_pair_data; umx_merge_CIs; umx_read_lower; umx_residualize; umx_round; umx_scale_wide_twin_data; umx_scale; umx_swap_a_block; umx, umx-package

Examples

```
## Not run:
df = umxPadAndPruneForDefVars(df, "E", "age", c("_T1", "_T2"))

## End(Not run)
```

umxPath *umxPath: Flexible specification of sem paths*

Description

The goal of this function is to enable quck-to-write, quick-to-read, flexible path descriptions for RAM models in OpenMx.

It introduces 11 new verbs: **with**, **var**, **cov**, **unique.bivariate**, **Cholesky**, **means**, **v1m0**, **v.m.**, **fixedAt**, **freeAt**, **firstAt**.

The new key "with" means you no-longer need set arrows = 2 on covariances. So you can say:

`umxPath(A, with = B)` instead of `mxPath(from = A, to = B, arrows = 2)`.

Specify a variance for A with

`umxPath(var = A)`.

This is equivalent to `mxPath(from = A, to = A, arrows = 2)`.

To specify a mean, you just say

`umxPath(mean = A)`, which is equivalent to `mxPath(from = "one", to = A)`.

To fix a path at a value, instead of to `mxPath(from = A, to = A, arrows = 2, free = FALSE, values = 1)` you can say:

`umxPath(var = A, fixedAt = 1)` .

The common task of creating a variable with variance fixed at 1 and mean at 0, you can simply say:

`umxPath(v1m0 = A)`

To just add estimates of variance and means to a variable use : `umxPath(v.m. = A)`

umxPath exposes “unique.parameter” as a parameter so you don’t have remember how to fill in connect= in mxPath

All unique bivariate paths can be specified using unique.bivariate. So say:

`umxPath(c('A',"B","C"), unique.bivariate = TRUE)` to create paths A<->B, B<->C, and A<->C.

Finally, you can create *Cholesky* form paths (see `umxACE`)

`umxPath(c("A1", "A2"), to c("var1", "var2"), Cholesky = TURE)`

Setting up a latent trait, you can fix the loading of the first path with

`mxPath(A, to = c(B,C,D), fixFirst = TRUE)`

This is equivalent to `mxPath(from = A, to = c(B,C,D), free = c(F, T, T), values = c(1, .5, .4))`.

Finally, in the future I will implement the John Fox "sem" package style notation, i.e., "A -> B". If you want to add multiple paths that way, separate them with a semi-colon or a return (see examples below.)

Usage

```
umxPath(from = NULL, to = NULL, with = NULL, var = NULL, cov = NULL,
  unique.bivariate = NULL, formative = NULL, Cholesky = NULL,
  means = NULL, v1m0 = NULL, v.m. = NULL, fixedAt = NULL,
  freeAt = NULL, firstAt = NULL, connect = "single", arrows = 1,
  free = TRUE, values = NA, labels = NA, lbound = NA, ubound = NA)
```

Arguments

from	either a source variable e.g "A" or c("A","B"), OR a sem-style path description, e.g. "A-> B" or "C <> B"
to	one or more target variables for one-headed paths, e.g "A" or c("A","B")
with	same as "to = vars, arrows = 2". nb: from, to= and var= must be left empty (their default)
var	equivalent to setting "from = vars, arrows = 2". nb: from, to, and with must be left empty (their default)
cov	equivalent to setting "from = X, to = Y, arrows = 2". nb: from, to, and with must be left empty (their default)
unique.bivariate	equivalent to setting "connect = "unique.bivariate", arrows = 2". nb: from, to, and with must be left empty (their default)
formative	Paired with to, this will build a formative variable, from the formatives, allowing these to covary, and to the latent "to" variable, fixing its variance to zero.
Cholesky	Treat the **from** vars as latent and **to** as measured, and connect up as in an ACE model.
means	equivalent to "from = 'one', to = x. nb: from, to, with and var must be left empty (their default).
v1m0	variance of 1 and mean of zero in one call.
v.m.	variance and mean added, both free.
fixedAt	Equivalent to setting "free = FALSE, values = x" nb: free and values must be left empty (their default)
freeAt	Equivalent to setting "free = TRUE, values = x" nb: free and values must be left empty (their default)
firstAt	first value is fixed at this (values passed to free are ignored: warning if not a single TRUE)
connect	as in mxPath - nb: Only used when using from and to
arrows	as in mxPath - nb: Only used when using from and to
free	whether the value is free to be optimised
values	default value list
labels	labels for each path
lbound	lower bounds for each path value
ubound	upper bounds for each path value

Details

This function returns a standard mxPath, but gives new options for specifying the path. In addition to the normal from and to, it adds specialised parameters for variances (var), two headed paths (with) and means (mean). There are also verbs for fixing values: "fixedAt" and "fixFirst" Finally, it also allows sem-style "A->B" string specification.

Value

- mxPath

References

- http://tbates.github.io, https://github.com/tbates/umx, http://openmx.psyc.virginia.edu

See Also

- umxLabel, mxMatrix, umxStart

Other Model Building Functions: umxLabel; umxLatent; umxRAM; umxReRun; umxRun; umxThresholdMatrix; umxValues; umx_fix_first_loadings; umx_fix_latents; umx, umx-package

Examples

```
require(OpenMx)
# Some examples of paths with umxPath
umxPath("A", to = "B")
umxPath("A", to = "B", fixedAt = 1)
umxPath("A", to = LETTERS[2:4], firstAt = 1) # Same as "free = FALSE, values = 1"
umxPath("A", with = "B") # using with: same as "to = B, arrows = 2"
umxPath("A", with = "B", fixedAt = .5)
umxPath("A", with = "B", firstAt = 1)
umxPath("A", with = c("B","C"), fixedAt = 1)
umxPath(var = "A") # Give a variance to A
umxPath(var = "A", fixedAt = 1)
umxPath(var = LETTERS[1:5], fixedAt = 1)
umxPath(cov = c("A", "B")) # Covariance A <-> B
umxPath(means = c("A","B")) # Create a means model for A: from = "one", to = "A"
umxPath(means = c("A","B"), values = c(pi,exp(1)))
# A worked example
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
myData = mxData(cov(demoOneFactor), type = "cov", numObs = 500)
m1 <- umxRAM("One Factor", data = myData,
umxPath(latents, to = manifests),
# umxPath("G -> manifests"),
umxPath(var = manifests),
umxPath(var = latents, fixedAt = 1.0)
)
m1 = mxRun(m1)
umxSummary(m1, show = "std")
```

```
#umxPath("A <-> B") # same path as above using a string
#umxPath("A -> B") # one-headed arrow with string syntax
#umxPath("A <> B; A <-- B") # This is ok too
#umxPath("A -> B; B>C; C --> D") # two paths. white space and hyphens not needed
## manifests is a reserved word, as is latents.
## It allows the string syntax to use the manifestVars variable
#umxPath("A -> manifests")
```

---

umxPlotACE *umxPlotACE*

Description

Make a graphical display of an ACE model

Usage

```
umxPlotACE(x = NA, dotFilename = "name", digits = 2, showMeans = FALSE,
  std = TRUE, ...)
```

Arguments

x	mxModel to plot (created by umxACE in order to inherit the MxModel.ACE class)
dotFilename	the name of the file that is created (use "name" to create the file using the model's name parameter)
digits	How many decimals to include in path loadings (default is 2)
showMeans	Whether to show means paths (default is FALSE)
std	Whether to standardize the model (default is TRUE)
...	Additional (optional) parameters

Value

- optionally return the dot code

References

- http://openmx.psyc.virginia.edu

See Also

Other Reporting functions: RMSEA.MxModel; RMSEA.summary.mxmodel; RMSEA; coef.MxModel; confint.MxModel; extractAIC.MxModel; logLik.MxModel; plot.MxModel, umxPlot; residuals.MxModel; umxCI_boot; umxCI; umxCompare; umxExpCov; umxExpMeans; umxFitIndices; umxStandardizeModel; umxSummary.MxModel; umx_drop_ok

Examples

```
require(OpenMx)
data(twinData)
labList = c("MZFF", "MZMM", "DZFF", "DZMM", "DZOS")
twinData$ZYG = factor(twinData$zyg, levels = 1:5, labels = labList)
selDVs = c("bmi1","bmi2")
mzData <- subset(twinData, ZYG == "MZFF", selDVs)
dzData <- subset(twinData, ZYG == "DZFF", selDVs)
m1 = umxACE(selDVs = selDVs, dzData = dzData, mzData = mzData)
m1 = mxRun(m1)
## Not run:
plot(m1)
umxPlotACE(m1, dotFilename = "override_model_name")
plot(m1, std = FALSE) # don't standardize

## End(Not run)
```

---

umxRAM *umxRAM*

Description

Making it as simple as possible to create a RAM model, without doing invisible things to the user.

Usage

```
umxRAM(name, data = NULL, ..., run = TRUE, exog.variances = FALSE,
  endog.variances = FALSE, latentVars = NULL, setValues = TRUE,
  independent = NA, remove_unused_manifests = TRUE, fix = "deprecated")
```

Arguments

name	A friendly name for the model
data	the data for the model. Can be an mxData or a data.frame
...	A list of mxPath, umxPath, or mxThreshold objects
run	Whether to mxRun the model (defaults to TRUE: the estimated model will be returned)
exog.variances	If TRUE, free variance parameters are added for exogenous variables that lack them (the default is FALSE).
endog.variances	If TRUE, free error-variance parameters are added for any endogenous variables that lack them (default is FALSE).
latentVars	Latents you want in your model (defaults to NULL, in which case any variable not in the data is assumed to be a latent variable)
setValues	Whether to try and guess good start values (Defults to TRUE, set them)
independent	Whether the model is independent (default = NA)

remove_unused_manifests
: Whether to remove variables in the data to which no path makes reference (defaults to TRUE)

fix
: deprecated. use umxPath(fixedAt = etc.

Details

Like mxModel, you list the theoretical causal paths. Unlike mxModel:

1. type defaults to "RAM"
2. You don't need to list manifestVars (they are detected from path usage)
3. You don't need to list latentVars (detected as anything in paths but not in mxData)
4. You add data like you do in lm, with **data =**
5. with umxPath you can use powerful verbs like **var =**

Comparison with other software

Some software has massive behind-the-scenes defaulting and path addition. I've played with some similar features (like auto-creating error and exogenous variances using endog.variances = TRUE and exog.variances = TRUE). Also identification helpers like fix = "latents" and fix = "firstLoadings"

To be honest, these are not only more trouble than they are worth, they encourage errors and poor modelling. I suggest user learn the handful of umxPath short cuts and stay clean and explicit!

Value

- mxModel

References

- http://tbates.github.io, https://github.com/tbates/umx

See Also

Other Model Building Functions: umxLabel; umxLatent; umxPath; umxReRun; umxRun; umxThresholdMatrix; umxValues; umx_fix_first_loadings; umx_fix_latents; umx, umx-package

Examples

```
# umxRAM is like ggplot2::qplot(), you give the data in a data =  parameter
# A common error is to include data in the main list,
# a bit like saying lm(y~x + df) instead of lm(y~x, data=dd)...
# nb: unlike mxModel, umxRAM needs data at build time.

# 1. For convenience, list up the manifests you will be using
selVars = c("mpg", "wt", "disp")

# 2. Create an mxData object
myCov = mxData(cov(mtcars[,selVars]), type = "cov", numObs = nrow(mtcars) )

# 3. Create the model (see ?umxPath for more nifty options)
```

```
m1 = umxRAM("tim", data = myCov,
umxPath(c("wt", "disp"), to = "mpg"),
umxPath(cov = c("wt", "disp")),
umxPath(var = c("wt", "disp", "mpg"))
)

## Not run:
# 5. Print a nice summary
umxSummary(m1, show = "std")

# 6. Draw a nice path diagram (needs Graphviz)
plot(m1)

## End(Not run)
```

---

umxReduce *umxReduce*

Description

Reduce a model - this is a work in progress

Usage

```
umxReduce(m1, report = 3, baseFileName = "tmp")
```

Arguments

m1	an mxModel to reduce
report	how to report the results table. 3 = html file
baseFileName	file to use when report = 3 (defaults to "tmp.html", I add the html)

Value

-

References

- http://tbates.github.io, https://github.com/tbates/umx, http://openmx.psyc.virginia.edu

See Also

- umxLabel, umxRun, umxStart

Examples

```
## Not run:
model = umxReduce(model)

## End(Not run)
```

---

umxReRun *umxReRun*

Description

umxReRun Is a convenience function to re-run an mxModel, optionally adding, setting, or dropping parameters. The main value for umxReRun is compactness. So this one-liner drops a path labelled "Cs", and returns the updated model:

Usage

```
umxReRun(lastFit, update = NULL, regex = FALSE, free = FALSE, value = 0,
  freeToStart = NA, name = NULL, verbose = FALSE, intervals = FALSE,
  comparison = FALSE, dropList = "deprecated")
```

Arguments

lastFit	The mxModel you wish to update and run.
update	What to update before re-running. Can be a list of labels, a regular expression (set regex = TRUE) or an object such as mxCI etc.
regex	Whether or not update is a regular expression (defaults to FALSE)
free	The state to set "free" to for the parameters whose labels you specify (defaults to free = FALSE, i.e., fixed)
value	The value to set the parameters whose labels you specify too (defaults to 0)
freeToStart	Whether to update parameters based on their current free-state. free = c(TRUE, FALSE, NA), (defaults to NA - i.e, not checked)
name	The name for the new model
verbose	How much feedback to give
intervals	Whether to run confidence intervals (see mxRun)
comparison	Whether to run umxCompare() after umxRun
dropList	A list of strings. If not NA, then the labels listed here will be dropped (or set to the value and free state you specify)

Details

```
fit2 = umxReRun(fit1, update = "Cs", name = "newModelName", comparison = TRUE)
```

A powerful feature is regular expression. These let you drop collections of paths by matching patterns fit2 = umxReRun(fit1, update = "C[sr]", regex = TRUE, name = "drop_Cs_andCr", comparison = TRUE)

If you're just starting out, you might find it easier to be more explicit. Like this:

fit2 = omxSetParameters(fit1, labels = "Cs", values = 0, free = FALSE, name = "newModelName")

fit2 = mxRun(fit2)

Value

- mxModel

References

- http://github.com/tbates/umx

See Also

Other Model Building Functions: umxLabel; umxLatent; umxPath; umxRAM; umxRun; umxThresholdMatrix; umxValues; umx_fix_first_loadings; umx_fix_latents; umx, umx-package

Examples

```
require(OpenMx)
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
m1 <- mxModel("One Factor", type = "RAM",
manifestVars = manifests, latentVars = latents,
mxPath(from = latents, to = manifests),
mxPath(from = manifests, arrows = 2),
mxPath(from = latents, arrows = 2, free = FALSE, values = 1.0),
mxData(cov(demoOneFactor), type = "cov", numObs = 500)
)
m1 = umxRun(m1, setLabels = TRUE, setValues = TRUE)
m2 = umxReRun(m1, update = "G_to_x1", name = "drop_X1")
umxSummary(m2); umxCompare(m1, m2)
# 1-line version including comparison
m2 = umxReRun(m1, update = "G_to_x1", name = "drop_X1", comparison = TRUE)
m2 = umxReRun(m1, update = "^G_to_x[3-5]", regex = TRUE, name = "no_G_to_x3_5", comp = TRUE)
m2 = umxReRun(m1, update = "G_to_x1", value = .2, name = "fix_G_x1", comp = TRUE)
m3 = umxReRun(m2, update = "G_to_x1", free = TRUE, name = "free_G_x1_again")
umxCompare(m3, m2)
```

umxRun *umxRun*

Description

umxRun is a version of mxRun which can run also set start values, labels, and run multiple times It can also calculate the saturated and independence likelihoods necessary for most fit indices.

Usage

```
umxRun(model, n = 1, calc_SE = TRUE, calc_sat = TRUE, setValues = FALSE,
  setLabels = FALSE, intervals = FALSE, comparison = NULL,
  setStarts = NULL)
```

Arguments

model	The mxModel you wish to run.
n	The maximum number of times you want to run the model trying to get a code green run (defaults to 1)
calc_SE	Whether to calculate standard errors (not used when n = 1) for the summary (they are not very accurate, so if you use mxCI or umxCI, you can turn this off)
calc_sat	Whether to calculate the saturated and independence models (for raw mxData mxModels) (defaults to TRUE - why would you want anything else?)
setValues	Whether to set the starting values of free parameters (defaults to F)
setLabels	Whether to set the labels (defaults to F)
intervals	Whether to run mxCI confindence intervals (defaults to F)
comparison	Whether to run umxCompare() after umxRun
setStarts	Deprecated way to setValues

Value

- mxModel

References

- http://www.github.com/tbates/umx

See Also

Other Model Building Functions: umxLabel; umxLatent; umxPath; umxRAM; umxReRun; umxThresholdMatrix; umxValues; umx_fix_first_loadings; umx_fix_latents; umx, umx-package

Examples

```
require(OpenMx)
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
m1 <- mxModel("One Factor", type = "RAM",
manifestVars = manifests, latentVars = latents,
mxPath(from = latents, to = manifests),
mxPath(from = manifests, arrows = 2),
mxPath(from = latents, arrows = 2, free = FALSE, values = 1.0),
mxData(cov(demoOneFactor), type = "cov", numObs = 500)
)
m1 = umxRun(m1) # just run: will create saturated model if needed
m1 = umxRun(m1, setValues = TRUE, setLabels = TRUE) # set start values and label all parameters
umxSummary(m1, show = "std")
m1 = mxModel(m1, mxCI("G_to_x1")) # add one CI
m1 = mxRun(m1, intervals = TRUE)
residuals(m1, run = TRUE) # get CIs on all free parameters
confint(m1, run = TRUE) # get CIs on all free parameters
m1 = umxRun(m1, n = 10) # re-run up to 10 times if not green on first run
```

---

umxSetParameters *umxSetParameters*

Description

umxSetParameters currently just a wrapper to omxSetParameters to ease user discovery. this also underlies to update, allowing homology with update() for lm models by freeing or fixing labeled parameters. It also set starts for parameters which now have identical labels

Usage

```
umxSetParameters(model, labels, free = NULL, values = NULL,
  newlabels = NULL, lbound = NULL, ubound = NULL, indep = FALSE,
  strict = TRUE, name = NULL)
```

Arguments

model	an mxModel to WITH
labels	= labels to find
free	= new value for free
values	= new values
newlabels	= newlabels
lbound	= value for lbound
ubound	= value for ubound
indep	= whether to look in indep models
strict	whether to complain if labels not found
name	= new name for the returned model

Value

- mxModel

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

- umxLabel

Other Modify or Compare Models: parameters, umxGetParameters; umxAdd1; umxDrop1; umxEquate; umxFixAll; umxMI; umxUnexplainedCausalNexus

Examples

```
require(OpenMx)
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
m1 <- umxRAM("One Factor", data = mxData(demoOneFactor, type = "raw"),
umxPath(from = latents, to = manifests),
umxPath(v.m. = manifests),
umxPath(v1m0 = latents)
)
parameters(m1, free=TRUE)
m2 = umxSetParameters(m1, "G_to_x1", newlabels= "G_to_x2")
```

---

umxStandardizeACE *umxStandardizeACE*

Description

standardize an ACE model

Usage

```
umxStandardizeACE(fit)
```

Arguments

fit	an mxModel to standardize

Value

- standardized ACE mxModel

References

- http://tbates.github.io, https://github.com/tbates/umx, http://openmx.psyc.virginia.edu

See Also

Other Reporting Functions: umx_APA_CI; umx_APA_pval; umx_aggregate; umx_print; umx_show; umx_time; umx, umx-package

Examples

```
## Not run:
fit = umxStandardizeACE(fit)

## End(Not run)
```

---

umxStandardizeModel *umxStandardizeModel*

Description

umxStandardizeModel takes a RAM-style model, and returns standardized version.

Usage

```
umxStandardizeModel(model, return = "parameters", Amatrix = NA,
  Smatrix = NA, Mmatrix = NA)
```

Arguments

model	The mxModel you wish to standardise
return	What to return. Valid options: "parameters", "matrices", or "model"
Amatrix	Optionally tell the function what the name of the asymmetric matrix is (defaults to RAM standard A)
Smatrix	Optionally tell the function what the name of the symmetric matrix is (defaults to RAM standard S)
Mmatrix	Optionally tell the function what the name of the means matrix is (defaults to RAM standard M)

Value

- a mxModel or else parameters or matrices if you request those

References

- http://github.com/tbates/umx

See Also

Other Reporting functions: RMSEA.MxModel; RMSEA.summary.mxmodel; RMSEA; coef.MxModel; confint.MxModel; extractAIC.MxModel; logLik.MxModel; plot.MxModel, umxPlot; plot, plot.MxModel.ACE, umxPlotACE; residuals.MxModel; umxCI_boot; umxCI; umxCompare; umxExpCov; umxExpMeans; umxFitIndices; umxSummary.MxModel; umx_drop_ok

Examples

```
require(OpenMx)
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
m1 <- mxModel("One Factor", type = "RAM",
manifestVars = manifests, latentVars = latents,
mxPath(from = latents, to = manifests),
mxPath(from = manifests, arrows = 2),
mxPath(from = latents, arrows = 2, free = FALSE, values = 1.0),
mxData(cov(demoOneFactor), type = "cov", numObs = 500)
)
m1 = umxRun(m1, setLabels = TRUE, setValues = TRUE)
m1 = umxStandardizeModel(m1, return = "model")
summary(m1)
```

---

umxSummary *umxSummary.default*

Description

Report the fit of a OpenMx model or specialized model class (such as ACE, CP etc.) in a compact form suitable for a journal. Because this same function is needed in umx, it gets defined twice currently.

Usage

```
umxSummary(model, ...)
```

Arguments

model	The mxModel whose fit will be reported
...	Other parameters to control model summary

Details

You can view documentation on the twin model subclass umxSummary.MxModel.ACE,

umxSummary.MxModel *umxSummary.MxModel*

Description

Report the fit of a model in a compact form suitable for a journal. Emits a "warning" when model fit is worse than accepted criterion (TLI >= .95 and RMSEA <= .06; (Hu & Bentler, 1999; Yu, 2002).

Usage

```
## S3 method for class 'MxModel'
umxSummary(model, refModels = NULL, report = "line",
  showEstimates = c("none", "raw", "std", "both", "list of column names"),
  digits = 2, RMSEA_CI = FALSE, matrixAddresses = FALSE,
  filter = c("ALL", "NS", "SIG"), SE = TRUE, ...)
```

Arguments

model	The mxModel whose fit will be reported
refModels	Saturated models if needed for fit indices (see example below: Only needed for raw data. nb also, umxRun takes care of this for you)
report	The format for the output line or table (default is "line")
showEstimates	What estimates to show. Options are c("none", "raw", "std", "both", "list of column names"). Default is "none" (just shows the fit indices)
digits	How many decimal places to report to (default = 2)
RMSEA_CI	Whether to compute the CI on RMSEA (Defaults to FALSE)
matrixAddresses	Whether to show "matrix address" columns (Default = FALSE)
filter	whether to show significant paths (SIG) or NS paths (NS) or all paths (ALL)
SE	Whether to compute SEs... defaults to TRUE. In rare cases, you might need to turn off to avoid errors.
...	Other parameters to control model summary

Details

notes on CIs and Identification Note, the conventional standard errors reported by OpenMx are used to produce the CIs you see in umxSummary These are used to derive confidence intervals based on the formula 95

Sometimes they appear NA. This often indicates a model which is not identified (seehttp://davidakenny.net/cm/identify.htm). This can include empirical under-identification - for instance two factors that are essentially identical in structure.

A signature of this would be paths estimated at or close to zero. Fixing one or two of these to zero may fix the standard error calculation, and alleviate the need to estimate likelihood-based or bootstrap CIs

If factor loadings can flip sign and provide identical fit, this creates another form of under-identification and can break confidence interval estimation, but I think Fixing a factor loading to 1 and estimating factor variances can help here

Value

- parameterTable returned invisibly, if estimates requested

References

- Hu, L., & Bentler, P. M. (1999). Cutoff criteria for fit indexes in covariance structure analysis: Coventional criteria versus new alternatives. Structural Equation Modeling, 6, 1-55.

- Yu, C.Y. (2002). Evaluating cutoff criteria of model fit indices for latent variable models with binary and continuous outcomes. University of California, Los Angeles, Los Angeles. Retrieved from http://www.statmodel.com/download/Yudissertation.pdf http://www.github.com/tbates/umx

See Also

- mxCI, umxCI_boot, umxRun

Other Reporting functions: RMSEA.MxModel; RMSEA.summary.mxmodel; RMSEA; coef.MxModel; confint.MxModel; extractAIC.MxModel; logLik.MxModel; plot.MxModel, umxPlot; plot, plot.MxModel.ACE, umxPlotACE; residuals.MxModel; umxCI_boot; umxCI; umxCompare; umxExpCov; umxExpMeans; umxFitIndices; umxStandardizeModel; umx_drop_ok

Examples

```
require(OpenMx)
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
m1 <- umxRAM("One Factor",
data = mxData(cov(demoOneFactor), type = "cov", numObs = 500),
umxPath(latents, to = manifests),
umxPath(var = manifests),
umxPath(var = latents, fixedAt = 1)
)
umxSummary(m1, show = "std")
umxSummary(m1, show = "raw")
m1 <- mxModel(m1,
  mxData(demoOneFactor, type = "raw"),
  umxPath(mean = manifests),
  umxPath(mean = latents, fixedAt = 0)
)
m1 <- mxRun(m1)
umxSummary(m1, show = "std")
# umxSummary(m1, report = "table") # not yet implemented
```

umxSummaryACE *umxSummaryACE*

Description

Summarise a Cholesky model, as returned by umxACE

Usage

```
umxSummaryACE(model, digits = 2, dotFilename = NULL, returnStd = FALSE,
  extended = FALSE, showRg = FALSE, showStd = TRUE, comparison = NULL,
  CIs = TRUE, zero.print = ".", report = 1, ...)
```

Arguments

model	an mxModel to summarize
digits	rounding (default = 2)
dotFilename	The name of the dot file to write: NA = none; "name" = use the name of the model
returnStd	Whether to return the standardized form of the model (default = F)
extended	how much to report (F)
showRg	= whether to show the genetic correlations (F)
showStd	= whether to show the standardized model (T)
comparison	you can run mxCompare on a comparison model (NULL)
CIs	Whether to show Confidence intervals if they exist (T)
zero.print	How to show zeros (".")
report	If 3, then open an html table of the results
...	Other parameters to control model summary

Value

- optional mxModel

References

- http://tbates.github.io, https://github.com/tbates/umx

See Also

- umxACE

Other Twin Reporting Functions: umx, umx-package

Examples

```
require(OpenMx)
data(twinData)
labList = c("MZFF", "MZMM", "DZFF", "DZMM", "DZOS")
twinData$ZYG = factor(twinData$zyg, levels = 1:5, labels = labList)
selDVs = c("bmi1", "bmi2")
mzData <- subset(twinData, ZYG == "MZFF", selDVs)
dzData <- subset(twinData, ZYG == "DZFF", selDVs)
m1 = umxACE(selDVs = selDVs, dzData = dzData, mzData = mzData)
m1 = umxRun(m1)
umxSummaryACE(m1)
## Not run:
umxSummaryACE(m1, dotFilename = NA);
umxSummaryACE(m1, dotFilename = "name", showStd = TRUE)
stdFit = umxSummaryACE(m1, returnStd = TRUE);

## End(Not run)
```

---

umxThresholdMatrix *umxThresholdMatrix*

Description

High-level helper for ordinal modeling. Creates, labels, and sets smart-starts for this complex matrix. Big time saver!

Usage

```
umxThresholdMatrix(df, suffixes = NA, threshMatName = "threshMat",
  method = c("auto", "Mehta", "allFree"), l_u_bound = c(NA, NA),
  deviationBased = TRUE, droplevels = FALSE, verbose = FALSE,
  hint = c("none", "left_censored"))
```

Arguments

df	the data being modelled (to allow access to the factor levels and quantiles within these for each variable)
suffixes	e.g. c("T1", "T2") - Use for data with repeated observations in a row (i.e., twin data) (defaults to NA)
threshMatName	name of the matrix which is returned. Defaults to "threshMat" - best not to change it.
method	How to set the thresholds: auto (the default), Mehta, which fixes the first two (auto chooses this for ordinal) or "allFree" (auto chooses this for binary)
l_u_bound	c(NA, NA) by default, you can use this to bound the thresholds. Careful you don't set bounds too close if you do.
deviationBased	Whether to build a helper matrix to keep the thresholds in order (defaults to = TRUE)

droplevels	Whether to drop levels with no observed data (defaults to FALSE)
verbose	(defaults to FALSE))
hint	currently used for "left_censored" data (defaults to "none"))

Details

When modeling ordinal data (sex, low-med-hi, depressed/normal, not at all, rarely, often, always), a useful conceptual strategy to handle expectations is to build a standard-normal model (i.e., a latent model with zero-means, and unit (1.0) variances), and then to threshold this normal distribution to generate the observed data. Thus an observation of "depressed" is modeled as a high score on the latent normally distributed trait, with thresholds set so that only scores above this threshold (1-minus the number of categories).

For **deviation methods**, it returns a list of lowerOnes_for_thresh, deviations_for_thresh & thresholdsAlgebra (named threshMatName)

For **direct**, it returns a thresholdsMatrix (named threshMatName)

Value

- thresholds matrix

References

- http://tbates.github.io, https://github.com/tbates/umx

See Also

Other Model Building Functions: umxLabel; umxLatent; umxPath; umxRAM; umxReRun; umxRun; umxValues; umx_fix_first_loadings; umx_fix_latents; umx, umx-package

Examples

```
x = data.frame(ordered(rbinom(100,1,.5))); names(x)<-c("x")
umxThresholdMatrix(x)
x = cut(rnorm(100), breaks = c(-Inf,.2,.5, .7, Inf)); levels(x) = 1:5
x = data.frame(ordered(x)); names(x)<-c("x")
umxThresholdMatrix(x)

require(OpenMx)
data(twinData)
labList = c("MZFF", "MZMM", "DZFF", "DZMM", "DZOS")
twinData$zyg = factor(twinData$zyg, levels = 1:5, labels = labList)
# ==================
# = Binary example =
# ==================
# Cut to form category of 80 % obese subjects
cutPoints <- quantile(twinData[, "bmi1"], probs = .2, na.rm = TRUE)
obesityLevels = c('normal', 'obese')
twinData$obese1 <- cut(twinData$bmi1, breaks = c(-Inf, cutPoints, Inf), labels = obesityLevels)
twinData$obese2 <- cut(twinData$bmi2, breaks = c(-Inf, cutPoints, Inf), labels = obesityLevels)
# Step 2: Make the ordinal variables into mxFactors
```

```
# this ensures ordered= TRUE + requires user to confirm levels
selDVs = c("obese1", "obese2")
twinData[, selDVs] <- mxFactor(twinData[, selDVs], levels = obesityLevels)
mzData <- subset(twinData, zyg == "MZFF", selDVs)
str(mzData)
umxThresholdMatrix(mzData, suffixes = 1:2)
umxThresholdMatrix(mzData, suffixes = 1:2, verbose = FALSE) # suppress informative messages

# ====================================
# = Ordinal (n categories > 2) example =
# ====================================
# Cut to form three categories of weight
cutPoints <- quantile(twinData[, "bmi1"], probs = c(.4, .7), na.rm = TRUE)
obesityLevels = c('normal', 'overweight', 'obese')
twinData$obeseTri1 <- cut(twinData$bmi1, breaks = c(-Inf, cutPoints, Inf), labels = obesityLevels)
twinData$obeseTri2 <- cut(twinData$bmi2, breaks = c(-Inf, cutPoints, Inf), labels = obesityLevels)
selDVs = c("obeseTri1", "obeseTri2")
twinData[, selDVs] <- mxFactor(twinData[, selDVs], levels = obesityLevels)
mzData <- subset(twinData, zyg == "MZFF", selDVs)
str(mzData)
umxThresholdMatrix(mzData, suffixes = 1:2)
umxThresholdMatrix(mzData, suffixes = 1:2, verbose = FALSE)

# ========================================================
# = Mix of all three kinds example (and a 4-level trait) =
# ========================================================

cutPoints <- quantile(twinData[, "bmi1"], probs = c(.25, .4, .7), na.rm = TRUE)
obesityLevels = c('underWeight', 'normal', 'overweight', 'obese')
twinData$obeseQuad1 <- cut(twinData$bmi1, breaks = c(-Inf, cutPoints, Inf), labels = obesityLevels)
twinData$obeseQuad2 <- cut(twinData$bmi2, breaks = c(-Inf, cutPoints, Inf), labels = obesityLevels)
selDVs = c("obeseQuad1", "obeseQuad2")
twinData[, selDVs] <- mxFactor(twinData[, selDVs], levels = obesityLevels)

selDVs = umx_paste_names(c("bmi", "obese", "obeseTri", "obeseQuad"), "", 1:2)
mzData <- subset(twinData, zyg == "MZFF", selDVs)
str(mzData)
umxThresholdMatrix(mzData, suffixes = 1:2, verbose = FALSE)

# ====================
# = "left_censored" =
# ====================

x = round(10*rnorm(1000, mean=-.2))
x[x<0] = 0
x = mxFactor(x, levels = sort(unique(x)))
x = data.frame(x)
umxThresholdMatrix(x, deviation = FALSE, hint = "left_censored")
```

umxUnexplainedCausalNexus *umxUnexplainedCausalNexus*

Description

umxUnexplainedCausalNexus report the effect of a change (delta) in a variable (from) on an output (to)

Usage

```
umxUnexplainedCausalNexus(from, delta, to, model)
```

Arguments

from	A variable in the model that you want to imput the effect of a change
delta	A the amount to simulate changing \"from\" by.
to	The dependent variable that you want to watch changing
model	The model containing from and to

References

- http://www.github.com/tbates/umx/

See Also

- umxRun, mxCompare

Other Modify or Compare Models: parameters, umxGetParameters; umxAdd1; umxDrop1; umxEquate; umxFixAll; umxMI; umxSetParameters

Examples

```
## Not run:
umxUnexplainedCausalNexus(from="yrsEd", delta = .5, to = "income35", model)

## End(Not run)
```

---

umxValues *umxValues*

Description

umxValues will set start values for the free parameters in RAM and Matrix mxModels, or even mxMatrices. It will try and be smart in guessing these from the values in your data, and the model type. If you give it a numeric input, it will use obj as the mean, return a list of length n, with sd = sd

Usage

```
umxValues(obj = NA, sd = NA, n = 1, onlyTouchZeros = FALSE)
```

Arguments

obj	The RAM or matrix mxModel, or mxMatrix that you want to set start values for.
sd	Optional Standard Deviation for start values
n	Optional Mean for start values
onlyTouchZeros	Don't start things that appear to have already been started (useful for speeding umxReRun)

Value

- mxModel with updated start values

References

- http://www.github.com/tbates/umx

See Also

- Core functions:

Other Model Building Functions: umxLabel; umxLatent; umxPath; umxRAM; umxReRun; umxRun; umxThresholdMatrix; umx_fix_first_loadings; umx_fix_latents; umx, umx-package

Examples

```
require(OpenMx)
data(demoOneFactor)
latents = c("G")
manifests = names(demoOneFactor)
m1 <- mxModel("One Factor", type = "RAM",
manifestVars = manifests, latentVars = latents,
mxPath(from = latents, to = manifests),
mxPath(from = manifests, arrows = 2),
mxPath(from = latents, arrows = 2, free = FALSE, values = 1.0),
mxData(cov(demoOneFactor), type = "cov", numObs = 500)
)
mxEval(S, m1) # default variances are 0
m1 = umxValues(m1)
mxEval(S, m1) # plausible variances
umx_print(mxEval(S,m1), 3, zero.print = ".") # plausible variances
umxValues(14, sd = 1, n = 10) # Return vector of length 10, with mean 14 and sd 1
# todo: handle complex guided matrix value starts...
```

umx_add_variances *umx_add_variances*

Description

Convenience function to save the user specifying mxPaths adding variance to each variable

Usage

```
umx_add_variances(model, add.to, values = NULL, free = NULL)
```

Arguments

model	an mxModel to add variances to
add.to	= List of variables to create variance for
values	= List of values (default = NULL)
free	= List of variables to create variance for (default = NULL)

Value

- mxModel

References

- http://tbates.github.io, https://github.com/tbates/umx, http://openmx.psyc.virginia.edu

See Also

Other Miscellaneous Functions: umxEval; umxJiggle; umx_APA_pval; umx_apply; umx_check_model; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_check; umx_default_option; umx_explode; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_cores; umx_get_optimizer; umx_has_CIs; umx_has_been_run; umx_has_means; umx_has_square_brackets; umx_is_MxMatrix; umx_is_MxModel; umx_is_RAM; umx_is_cov; umx_is_endogenous; umx_is_exogenous; umx_is_ordered; umx_msg; umx_names; umx_object_as_str; umx_paste_names; umx_print; umx_rename; umx_reorder; umx_rot; umx_set_cores; umx_set_optimizer; umx_string_to_algebra; umx_trim; umx, umx-package

Examples

```
require(OpenMx)
data(demoOneFactor)
m1 <- mxModel("One Factor", type = "RAM",
 manifestVars = names(demoOneFactor),
 latentVars = "g",
mxPath(from = "g", to = names(demoOneFactor), values= .1),
mxData(cov(demoOneFactor), type = "cov", numObs = 500)
)
```

```
umx_show(m1, matrices = "S") # variables lack variance :-(
m1 = umx_add_variances(m1, add.to = names(demoOneFactor))
m1 = umx_add_variances(m1, add.to = "g", FALSE, 1)
umx_show(m1, matrices = "S")
# Note: latent g has been treated like the manifests...
# umxFixLatents() will take care of this for you...
m1 = umxRun(m1, setLabels = TRUE, setValues = TRUE)
umxSummary(m1)
```

---

umx_aggregate *umx_aggregate*

Description

umx_aggregate Aggregate based on a formula, using a function. Has some handy base functions

Usage

```
umx_aggregate(formula = DV ~ condition, data, what = c("mean_sd", "n"))
```

Arguments

formula	the aggregation formula. e.g., DV ~ condition
data	the dataframe to aggregate with
what	function to use. Defaults to a built-in "smart" mean (sd)

Value

- table

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

- aggregate

Other Reporting Functions: umxStandardizeACE; umx_APA_CI; umx_APA_pval; umx_print; umx_show; umx_time; umx, umx-package

Examples

```
aggregate(mpg ~ cyl, FUN = mean, na.rm = TRUE, data = mtcars)
umx_aggregate(mpg ~ cyl, data = mtcars)
umx_aggregate(cbind(mpg, qsec) ~ cyl, data = mtcars)
t(umx_aggregate(cbind(mpg, qsec) ~ cyl, data = mtcars))
## Not run:
umx_aggregate(cbind(moodAvg, mood) ~ condition, data = study1)

## End(Not run)
```

umx_APA_CI *umx_APA_CI*

Description

Given an lm, will return a nicely-formated effect including 95% CI in square brackets, for one of the effects (specified by name in se). e.g.:

umx_APA_CI(m1, "wt") β = -5.344 [-6.486, -4.203], p< 0.001

Given b and se will return a CI based on 1.96 times the se.

Usage

```
umx_APA_CI(b, se, digits = 3)
```

Arguments

b	Either a model (lm), or a beta-value
se	If b is a model, then name of the parameter of interest, else the SE (standard-error)
digits	How many digits to use in rounding values

Value

- string

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

Other Reporting Functions: umxStandardizeACE; umx_APA_pval; umx_aggregate; umx_print; umx_show; umx_time; umx, umx-package

Examples

```
umx_APA_CI(lm(mpg ~ wt, mtcars), "wt")
umx_APA_CI(.4, .3)
```

umx_APA_pval *umx_APA_pval*

Description

round a p value so you get < .001 instead of .000000002 or 1.00E-09

Usage

```
umx_APA_pval(p, min = 0.001, rounding = 3, addComparison = NA)
```

Arguments

p	The p-value to round
min	Values below min reported as "< min"
rounding	Number of decimal to which to round
addComparison	Whether to add '=' '<' etc. (NA adds when needed)

Value

- formatted p-value

See Also

- round

Other Miscellaneous Functions: umxEval; umxJiggle; umx_add_variances; umx_apply; umx_check_model; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_check; umx_default_option; umx_explode; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_cores; umx_get_optimizer; umx_has_CIs; umx_has_been_run; umx_has_means; umx_has_square_brackets; umx_is_MxMatrix; umx_is_MxModel; umx_is_RAM; umx_is_cov; umx_is_endogenous; umx_is_exogenous; umx_is_ordered; umx_msg; umx_names; umx_object_as_str; umx_paste_names; umx_print; umx_rename; umx_reorder; umx_rot; umx_set_cores; umx_set_optimizer; umx_string_to_algebra; umx_trim; umx, umx-package

Other Reporting Functions: umxStandardizeACE; umx_APA_CI; umx_aggregate; umx_print; umx_show; umx_time; umx, umx-package

Examples

```
umx_APA_pval(.052347)
umx_APA_pval(1.23E-3)
umx_APA_pval(1.23E-4)
umx_APA_pval(c(1.23E-3, .5))
umx_APA_pval(c(1.23E-3, .5), addComparison = TRUE)
```

umx_apply *umx_apply*

Description

Tries to make apply more readable. Other functions to think of include cumsum, rowSums, colMeans, etc.

Usage

```
umx_apply(FUN, of, by = "columns", ...)
```

Arguments

FUN	The function to apply
of	The dataframe to work with
by	What to apply the function to: columns or rows (default = "columns")
...	optional arguments to FUN, i.e., na.rm = T

Value

- mxModel

References

- http://tbates.github.io, https://github.com/tbates/umx, http://openmx.psyc.virginia.edu

See Also

Other Miscellaneous Functions: umxEval; umxJiggle; umx_APA_pval; umx_add_variances; umx_check_model; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_check; umx_default_option; umx_explode; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_cores; umx_get_optimizer; umx_has_CIs; umx_has_been_run; umx_has_means; umx_has_square_brackets; umx_is_MxMatrix; umx_is_MxModel; umx_is_RAM; umx_is_cov; umx_is_endogenous; umx_is_exogenous; umx_is_ordered; umx_msg; umx_names; umx_object_as_str; umx_paste_names; umx_print; umx_rename; umx_reorder; umx_rot; umx_set_cores; umx_set_optimizer; umx_string_to_algebra; umx_trim; umx, umx-package

Examples

```
umx_apply(mean, mtcars, by = "columns")
umx_apply(mean, of = mtcars, by = "columns")
umx_apply(mean, by = "rows", of = mtcars[1:3,], na.rm = TRUE)
```

umx_as_numeric *umx_as_numeric*

Description

Convert each column of a dataframe to numeric

Usage

```
umx_as_numeric(df, force = FALSE)
```

Arguments

df	a data.frame to convert
force	whether to force conversion to numeric for non-numeric columns (defaults to FALSE)

Value

- data.frame

References

- http://www.github.com/tbates/umx

See Also

Other Miscellaneous Data Functions: umxHetCor; umxPadAndPruneForDefVars; umx_cont_2_ordinal; umx_cov2raw; umx_lower2full; umx_make_bin_cont_pair_data; umx_merge_CIs; umx_read_lower; umx_residualize; umx_round; umx_scale_wide_twin_data; umx_scale; umx_swap_a_block; umx, umx-package

Examples

```
df = mtcars
# make one variable non-numeric
df$mpg = c(letters,letters[1:6]); str(df)
df = umx_as_numeric(df)
```

umx_check *umx_check*

Description

Check that a test evaluates to TRUE. If not, stop, warn, or message the user

Usage

```
umx_check(boolean.test, action = c("stop", "warning", "message"),
  message = "check failed")
```

Arguments

boolean.test	test evaluating to TRUE or FALSE
action	One of "stop" (the default), "warning", or "message"
message	what to tell the user when boolean.test is FALSE

Value

- boolean

References

- http://www.github.com/tbates/umx

See Also

Other Miscellaneous Functions: umxEval; umxJiggle; umx_APA_pval; umx_add_variances; umx_apply; umx_check_model; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_default_option; umx_explode; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_cores; umx_get_optimizer; umx_has_CIs; umx_has_been_run; umx_has_means; umx_has_square_brackets; umx_is_MxMatrix; umx_is_MxModel; umx_is_RAM; umx_is_cov; umx_is_endogenous; umx_is_exogenous; umx_is_ordered; umx_msg; umx_names; umx_object_as_str; umx_paste_names; umx_print; umx_rename; umx_reorder; umx_rot; umx_set_cores; umx_set_optimizer; umx_string_to_algebra; umx_trim; umx, umx-package

Examples

```
umx_check(length(1:3)==3, "stop", "item must have length == 3")
```

umx_check_model *umx_check_model*

Description

Check an OpenMx model

Usage

```
umx_check_model(obj, type = NULL, hasData = NULL, beenRun = NULL,
  hasMeans = NULL, checkSubmodels = FALSE)
```

Arguments

obj	an object to check
type	what type the model must be, i.e., "RAM", "LISREL", etc. (defaults to not checking NULL)
hasData	whether the model should have data or not (defaults to not checking NULL)
beenRun	whether the model has been run or not (defaults to not checking NULL)
hasMeans	whether the model should have a means model or not (defaults to not checking NULL)
checkSubmodels	whether to check submodels (not implemented yet) (default = FALSE)

Value

- boolean

References

- http://www.github.com/tbates/umx

See Also

- umx, umxRun, umxValues

Other Miscellaneous Functions: umxEval; umxJiggle; umx_APA_pval; umx_add_variances; umx_apply; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_check; umx_default_option; umx_explode; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_cores; umx_get_optimizer; umx_has_CIs; umx_has_been_run; umx_has_means; umx_has_square_brackets; umx_is_MxMatrix; umx_is_MxModel; umx_is_RAM; umx_is_cov; umx_is_endogenous; umx_is_exogenous; umx_is_ordered; umx_msg; umx_names; umx_object_as_str; umx_paste_names; umx_print; umx_rename; umx_reorder; umx_rot; umx_set_cores; umx_set_optimizer; umx_string_to_algebra; umx_trim; umx, umx-package

Examples

```
require(OpenMx)
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
m1 <- mxModel("One Factor", type = "RAM",
manifestVars = manifests, latentVars = latents,
mxPath(from = latents, to = manifests),
mxPath(from = manifests, arrows = 2),
mxPath(from = latents, arrows = 2, free = FALSE, values = 1.0),
mxData(cov(demoOneFactor), type = "cov", numObs = 500)
)
umx_check_model(m1)
umx_check_model(m1, type = "RAM") # equivalent to umx_is_RAM()
umx_check_model(m1, hasData = TRUE)
## Not run:
umx_check_model(m1, hasMeans = TRUE)
umx_check_model(m1, beenRun = FALSE)

## End(Not run)
```

---

umx_check_multi_core *umx_check_multi_core*

Description

Shows how many cores you are using, and runs a test script so user can check CPU usage

Usage

```
umx_check_multi_core(pathToDemos = "~/bin/OpenMx/inst/models/nightly/",
  demoScript = "3LatentMultiRegWithContinuousModerator-c.R")
```

Arguments

pathToDemos	where to look for demo scripts (Default "~/bin/OpenMx/inst/models/nightly/")
demoScript	which demo script to use (Default ""3LatentMultiRegWithContinuousModerator-c.R"")

Value

- NULL

References

- http://tbates.github.io, https://github.com/tbates/umx

See Also

Other Miscellaneous Functions: umxEval; umxJiggle; umx_APA_pval; umx_add_variances; umx_apply; umx_check_model; umx_checkpoint, umx_set_checkpoint; umx_check; umx_default_option; umx_explode; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_cores; umx_get_optimizer; umx_has_CIs; umx_has_been_run; umx_has_means; umx_has_square_brackets; umx_is_MxMatrix; umx_is_MxModel; umx_is_RAM; umx_is_cov; umx_is_endogenous; umx_is_exogenous; umx_is_ordered; umx_msg; umx_names; umx_object_as_str; umx_paste_names; umx_print; umx_rename; umx_reorder; umx_rot; umx_set_cores; umx_set_optimizer; umx_string_to_algebra; umx_trim; umx, umx-package

Examples

```
## Not run:
model = umx_check_multi_core()

## End(Not run)
```

---

umx_check_names *umx_check_names*

Description

Check if a list of names are in the names() of a dataframe (or the of a matrix)

Usage

```
umx_check_names(namesNeeded, data, die = TRUE, no_others = FALSE)
```

Arguments

namesNeeded	list of variable names to find
data	data.frame (or matrix) to search in for names
die	whether to die if the check fails (defaults to TRUE)
no_others	Whether to test that the data contain no columns in addition to those in namesNeeded (defaults to FALSE)

Value

- boolean

References

- http://www.github.com/tbates/umx

See Also

Other Miscellaneous Building Functions: umx_cov_diag

Examples

```
require(OpenMx)
data(demoOneFactor) # "x1" "x2" "x3" "x4" "x5"
umx_check_names(c("x1", "x2"), demoOneFactor)
umx_check_names(c("x1", "x2"), as.matrix(demoOneFactor))
umx_check_names(c("x1", "x2"), cov(demoOneFactor[,c("x1","x2")]))
umx_check_names(c("z1", "x2"), data = demoOneFactor, die = FALSE)
umx_check_names(c("x1", "x2"), data = demoOneFactor, die = FALSE, no_others = TRUE)
umx_check_names(c("x1","x2","x3","x4","x5"), data = demoOneFactor, die = FALSE, no_others = TRUE)
## Not run:
umx_check_names(c("bad_var_name", "x2"), data = demoOneFactor, die = TRUE)

## End(Not run)
```

---

umx_check_OS *umx_check_OS*

Description

Check what OS we are running on (current default is OS X). Returns a boolean. Optionally warn or die on failure of the test

Usage

```
umx_check_OS(target = c("OSX", "SunOS", "Linux", "Windows"),
  action = c("ignore", "warn", "die"))
```

Arguments

target	Which OS(s) you wish to check for (default = "OSX")
action	What to do on failure of the test: nothing (default), warn or die

Value

- TRUE if on the specified OS (else FALSE)

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

Other Miscellaneous Utility Functions: qm; umx_find_object; umx_grep; umx_pb_note; umx, umx-package

Examples

```
umx_check_OS()
```

umx_cont_2_ordinal *umx_cont_2_ordinal*

Description

Recode a variable into n-quantiles (default = deciles (10 levels)). It returns an mxFactor, with the levels labeled with the max value in each quantile (i.e., open on the left-side).

Usage

```
umx_cont_2_ordinal(var, nlevels = 10, type = c("mxFactor", "ordered",
  "unordered"), verbose = FALSE)
```

Arguments

var	a variable to recode as ordinal
nlevels	how many bins or levels (at most) to use (default = 10 i,e deciles)
type	what to return (Default is "mxFactor") options include "ordered" and "unordered")
verbose	report the min, max, and decile cuts used (default = FALSE)

Details

Note: Redundant bins are merged. i.e., if the same score identifies all deciles up to the fourth, then these will be merged into one level.

Value

- recoded variable as an mxFactor

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

Other Miscellaneous Data Functions: umxHetCor; umxPadAndPruneForDefVars; umx_as_numeric; umx_cov2raw; umx_lower2full; umx_make_bin_cont_pair_data; umx_merge_CIs; umx_read_lower; umx_residualize; umx_round; umx_scale_wide_twin_data; umx_scale; umx_swap_a_block; umx, umx-package

Examples

```
x = umx_cont_2_ordinal(rnorm(10000), verbose = TRUE)
x = umx_cont_2_ordinal(rep(0:10, 10), verbose = TRUE)
levels(x)
str(umx_cont_2_ordinal(rnorm(10000), nlevels = 4, verbose = TRUE))
```

umx_cor *umx_cor*

Description

Report correlations and their p-values

Usage

```
umx_cor(X, df = nrow(X) - 2, use = "pairwise.complete.obs", digits = 3)
```

Arguments

X	a matrix or dataframe
df	the degrees of freedom for the test
use	how to handle missing data
digits	rounding of answers

Value

- matrix of correlations and p-values

References

- http://www.github.com/tbates/umx

See Also

- umxLabel, umxRun, umxValues

Other Miscellaneous Stats Functions: reliability; umxCov2cor; umx_means; umx, umx-package

Examples

```
umx_cor(myFADataRaw[1:8,])
```

umx_cov2raw *Turn a cov matrix into raw data*

Description

Turns a covariance matrix into comparable raw data :-)

Usage

```
umx_cov2raw(myCovariance, n, means = 0)
```

Arguments

myCovariance	a covariance matrix
n	how many rows of data to return
means	the means of the raw data (defaults to 0)

Value

- data.frame

References

- http://tbates.github.io, https://github.com/tbates/umx

See Also

- cov2cor

Other Miscellaneous Data Functions: umxHetCor; umxPadAndPruneForDefVars; umx_as_numeric; umx_cont_2_ordinal; umx_lower2full; umx_make_bin_cont_pair_data; umx_merge_CIs; umx_read_lower; umx_residualize; umx_round; umx_scale_wide_twin_data; umx_scale; umx_swap_a_block; umx, umx-package

Examples

```
covData <- matrix(nrow=6, ncol=6, byrow=TRUE, dimnames=list(paste0("v", 1:6), paste0("v", 1:6)),
data = c(0.9223099, 0.1862938, 0.4374359, 0.8959973, 0.9928430, 0.5320662,
           0.1862938, 0.2889364, 0.3927790, 0.3321639, 0.3371594, 0.4476898,
           0.4374359, 0.3927790, 1.0069552, 0.6918755, 0.7482155, 0.9013952,
           0.8959973, 0.3321639, 0.6918755, 1.8059956, 1.6142005, 0.8040448,
           0.9928430, 0.3371594, 0.7482155, 1.6142005, 1.9223567, 0.8777786,
           0.5320662, 0.4476898, 0.9013952, 0.8040448, 0.8777786, 1.3997558))
myData = umx_cov2raw(covData, n = 100, means = 1:6)
```

umx_cov_diag *umx_cov_diag*

Description

Helper to get variances from a df that might contain some non-numeric columns. Values at non-numeric columns are set to the value passed in as ordVar.

Usage

```
umx_cov_diag(df, ordVar = 1, format = c("diag", "Full", "Lower"),
  use = c("complete.obs", "pairwise.complete.obs", "everything", "all.obs",
  "na.or.complete"))
```

Arguments

df	a dataframe of raw data from which to get variances.
ordVar	The value to return at any ordinal columns (defaults to 1)
format	to return: options are c("diag", "Full", "Lower"). Defaults to diag: a vector of variances
use	passed to cov - defaults to "complete.obs" (other options are in the function)

Value

- mxModel

References

- http://tbates.github.io, https://github.com/tbates/umx, http://openmx.psyc.virginia.edu

See Also

- umxLabel, umxRun, umxStart

Other Miscellaneous Building Functions: umx_check_names

Examples

```
tmp = mtcars[,1:4]
tmp$cyl = ordered(mtcars$cyl) # ordered factor
tmp$hp  = ordered(mtcars$hp)  # binary factor
umx_cov_diag(tmp, ordVar = 1, use = "pair")
tmp2 = tmp[, c(1,3)]
umx_cov_diag(tmp2)
umx_cov_diag(tmp2, format = "Full")
```

umx_default_option *umx_default_option*

Description

Handle parameter options given as a default list in a function. This is just a version of x = match.arg(x) which allows items not in the list.

Usage

```
umx_default_option(x, option_list, check = TRUE)
```

Arguments

x	the value chosen (may be a selection, or the default list of options)
option_list	TODO fix this documentation
check	Whether to check that single items are in the list. Set false to accept abbreviations (defaults to TRUE)

Value

- the option

References

- http://www.github.com/tbates/umx

See Also

- match.arg

Other Miscellaneous Functions: umxEval; umxJiggle; umx_APA_pval; umx_add_variances; umx_apply; umx_check_model; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_check; umx_explode; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_cores; umx_get_optimizer; umx_has_CIs; umx_has_been_run; umx_has_means; umx_has_square_brackets; umx_is_MxMatrix; umx_is_MxModel; umx_is_RAM; umx_is_cov; umx_is_endogenous; umx_is_exogenous; umx_is_ordered; umx_msg; umx_names; umx_object_as_str; umx_paste_names; umx_print; umx_rename; umx_reorder; umx_rot; umx_set_cores; umx_set_optimizer; umx_string_to_algebra; umx_trim; umx, umx-package

Examples

```
## Not run:
option_list = c("default", "par.observed", "empirical")
umx_default_option("par.observed", option_list)
umx_default_option("bad", option_list)
umx_default_option("allow me", option_list, check = FALSE)
umx_default_option(option_list, option_list)
```

```
option_list = c(NULL, "par.observed", "empirical")
umx_default_option(option_list, option_list) # fails with NULL!!!!!
option_list = c(NA, "par.observed", "empirical")
umx_default_option(option_list, option_list) # use NA instead
option_list = c(TRUE, FALSE, NA)
umx_default_option(option_list, option_list) # works with non character

## End(Not run)
```

---

umx_drop_ok *umx_drop_ok*

Description

Print a meaningful sentence about a model comparison. If you use this, please email me and ask to have it merged with umxCompare() :-)

Usage

```
umx_drop_ok(model1, model2, text = "parameter")
```

Arguments

model1	the base codemxModel
model2	the nested codemxModel
text	name of the thing being tested, i.e., "Extraversion" or "variances"

Value

-

References

- http://tbates.github.io, https://github.com/tbates/umx

See Also

Other Reporting functions: RMSEA.MxModel; RMSEA.summary.mxmodel; RMSEA; coef.MxModel; confint.MxModel; extractAIC.MxModel; logLik.MxModel; plot.MxModel, umxPlot; plot, plot.MxModel.ACE, umxPlotACE; residuals.MxModel; umxCI_boot; umxCI; umxCompare; umxExpCov; umxExpMeans; umxFitIndices; umxStandardizeModel; umxSummary.MxModel

Examples

```
## Not run:
model = umx_drop_ok(model)

## End(Not run)
```

umx_explode *umx_explode - like php's explode function*

Description

Takes a string and returns each character as an item in an array

Usage

```
umx_explode(delimiter = character(), string)
```

Arguments

delimiter	what to break the string on. Default is empty string ""
string	an character string, e.g. "dog"

Value

- a collection of characters, e.g. c("d", "o", "g")

References

- http://tbates.github.io, https://github.com/tbates/umx, http://www.php.net/

See Also

Other Miscellaneous Functions: umxEval; umxJiggle; umx_APA_pval; umx_add_variances; umx_apply; umx_check_model; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_check; umx_default_option; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_cores; umx_get_optimizer; umx_has_CIs; umx_has_been_run; umx_has_means; umx_has_square_brackets; umx_is_MxMatrix; umx_is_MxModel; umx_is_RAM; umx_is_cov; umx_is_endogenous; umx_is_exogenous; umx_is_ordered; umx_msg; umx_names; umx_object_as_str; umx_paste_names; umx_print; umx_rename; umx_reorder; umx_rot; umx_set_cores; umx_set_optimizer; umx_string_to_algebra; umx_trim; umx, umx-package

Examples

```
umx_explode("", "dog") # "d" "o" "g"
umx_explode(" ", "cats and dogs") # [1] "cats" "and"  "dogs"
```

umx_fake_data *umx_fake_data*

Description

This function takes as argument an existing dataset, which must be either a matrix or a data frame. Each column of the dataset must consist either of numeric variables or ordered factors. When one or more ordered factors are included, then a heterogeneous correlation matrix is computed using John Fox's polycor package. Pairwise complete observations are used for all covariances, and the exact pattern of missing data present in the input is placed in the output, provided a new sample size is not requested. Warnings from the polycor::hetcor function are suppressed.

Usage

```
umx_fake_data(dataset, digits = 2, n = NA, use.names = TRUE,
  use.levels = TRUE, use.miss = TRUE, mvt.method = "eigen",
  het.ML = FALSE, het.suppress = TRUE)
```

Arguments

dataset	The original dataset you want to make a simulacrum of
digits	= 2
n	= NA
use.names	= T
use.levels	= T
use.miss	= T
mvt.method	= "eigen"
het.ML	= F
het.suppress	= T

Details

Author: Ryne Estabrook Created: 17 Aug 2010

Value

- new dataframe

See Also

- umxLabel, umxRun, umxValues

Examples

```
fakeCars = umx_fake_data(mtcars)
```

umx_find_object *umx_find_object*

Description

Find objects a certain class, whose name matches a search string. The string (pattern) is grep-enabled, so you can match wild-cards

Usage

```
umx_find_object(pattern = ".*", requiredClass = "MxModel")
```

Arguments

pattern	the pattern that matching objects must contain
requiredClass	the class of object that will be matched

Value

- a list of objects matching the class and name

References

-

See Also

- grep, umxRun, umxValues

Other Miscellaneous Utility Functions: qm; umx_check_OS; umx_grep; umx_pb_note; umx, umx-package

Examples

```
## Not run:
umx_find_object("^m[0-9]") # mxModels beginning "m1" etc.

## End(Not run)
```

umx_fix_first_loadings *umx_fix_first_loadings*

Description

Fix the loading of the first path from each latent at selected value (default = 1).

Usage

```
umx_fix_first_loadings(model, latents = NULL, at = 1)
```

Arguments

model	an mxModel to set
latents	(If NULL then all latentVars in model)
at	(Default = 1)

Value

- mxModel

References

- http://tbates.github.io, https://github.com/tbates/umx, http://openmx.psyc.virginia.edu

See Also

Other Model Building Functions: umxLabel; umxLatent; umxPath; umxRAM; umxReRun; umxRun; umxThresholdMatrix; umxValues; umx_fix_latents; umx, umx-package

Examples

```
require(OpenMx)
data(demoOneFactor)
m1 <- mxModel("One Factor", type = "RAM",
 manifestVars = names(demoOneFactor),
 latentVars = "g",
mxPath(from = "g", to = names(demoOneFactor)),
mxPath(from = names(demoOneFactor), arrows = 2),
mxData(cov(demoOneFactor), type = "cov", numObs = 500)
)
m1 = umx_fix_first_loadings(m1)
umx_show(m1) # variance of g is fixed at 1
```

umx_fix_latents *umx_fix_latents*

Description

Fix the variance of all, or selected, exogenous latents at selected values. This function adds a variance to the factor if it does not exist.

Usage

```
umx_fix_latents(model, latents = NULL, exogenous.only = TRUE, at = 1)
```

Arguments

model	an mxModel to set
latents	(If NULL then all latentVars)
exogenous.only	only touch exogenous latents (default = TRUE)
at	(Default = 1)

Value

- mxModel

References

- http://tbates.github.io, https://github.com/tbates/umx, http://openmx.psyc.virginia.edu

See Also

Other Model Building Functions: umxLabel; umxLatent; umxPath; umxRAM; umxReRun; umxRun; umxThresholdMatrix; umxValues; umx_fix_first_loadings; umx, umx-package

Examples

```
require(OpenMx)
data(demoOneFactor)
m1 <- mxModel("One Factor", type = "RAM",
 manifestVars = names(demoOneFactor),
 latentVars = "g",
mxPath(from = "g", to = names(demoOneFactor)),
mxPath(from = names(demoOneFactor), arrows = 2),
mxData(cov(demoOneFactor), type = "cov", numObs = 500)
)
umx_show(m1, matrices = "S") # variance of g is not set
m1 = umx_fix_latents(m1)
umx_show(m1, matrices = "S") # variance of g is fixed at 1
```

umx_get_bracket_addresses *get mat[r,c] style cell address from an mxMatrix*

Description

Sometimes you want these :-) This also allows you to change the matrix name: useful for using mxMatrix addresses in an mxAlgebra.

Usage

```
umx_get_bracket_addresses(mat, free = NA, newName = NA)
```

Arguments

mat	an mxMatrix to get address labels from
free	how to filter on free (default = NA: take all)
newName	= NA

Value

- a list of bracket style labels

References

- http://tbates.github.io, https://github.com/tbates/umx

See Also

Other Miscellaneous Functions: umxEval; umxJiggle; umx_APA_pval; umx_add_variances; umx_apply; umx_check_model; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_check; umx_default_option; umx_explode; umx_get_CI_as_APA_string; umx_get_checkpoint; umx_get_cores; umx_get_optimizer; umx_has_CIs; umx_has_been_run; umx_has_means; umx_has_square_brackets; umx_is_MxMatrix; umx_is_MxModel; umx_is_RAM; umx_is_cov; umx_is_endogenous; umx_is_exogenous; umx_is_ordered; umx_msg; umx_names; umx_object_as_str; umx_paste_names; umx_print; umx_rename; umx_reorder; umx_rot; umx_set_cores; umx_set_optimizer; umx_string_to_algebra; umx_trim; umx, umx-package

Examples

```
require(OpenMx)
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
m1 <- mxModel("One Factor", type = "RAM",
manifestVars = manifests, latentVars = latents,
mxPath(from = latents, to = manifests),
mxPath(from = manifests, arrows = 2),
```

```
mxPath(from = latents, arrows = 2, free = FALSE, values = 1.0),
mxData(cov(demoOneFactor), type = "cov", numObs = 500)
)
umx_get_bracket_addresses(m1$matrices$A, free= TRUE)
```

---

umx_get_checkpoint *umx_get_checkpoint*

Description

get the checkpoint status for a model or global options

Usage

```
umx_get_checkpoint(model = NULL)
```

Arguments

model	an optional model to get options from

Value

- NULL

References

- http://tbates.github.io, https://github.com/tbates/umx, http://openmx.psyc.virginia.edu

See Also

- umxLabel, umxRun, umxStart

Other Miscellaneous Functions: umxEval; umxJiggle; umx_APA_pval; umx_add_variances; umx_apply; umx_check_model; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_check; umx_default_option; umx_explode; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_cores; umx_get_optimizer; umx_has_CIs; umx_has_been_run; umx_has_means; umx_has_square_brackets; umx_is_MxMatrix; umx_is_MxModel; umx_is_RAM; umx_is_cov; umx_is_endogenous; umx_is_exogenous; umx_is_ordered; umx_msg; umx_names; umx_object_as_str; umx_paste_names; umx_print; umx_rename; umx_reorder; umx_rot; umx_set_cores; umx_set_optimizer; umx_string_to_algebra; umx_trim; umx, umx-package

Examples

```
umx_get_checkpoint() # current global default
require(OpenMx)
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
m1 <- umxRAM("One Factor", mxData(cov(demoOneFactor), type = "cov", numObs = 500),
umxPath(latents, to = manifests),
umxPath(var = manifests),
umxPath(var = latents, fixedAt = 1.0)
)
m1 = umx_set_checkpoint(interval = 2, model = m1)
umx_get_checkpoint(model = m1)
```

---

umx_get_CI_as_APA_string

umx_get_CI_as_APA_string

Description

Look up CIs for free parameters in a model, and return as APA-formatted text string

Usage

```
umx_get_CI_as_APA_string(model, cellLabel, prefix = "top.", suffix = "_std",
  digits = 2, verbose = FALSE)
```

Arguments

model	an mxModel to get CIs from
cellLabel	the label of the cell to interogate for a CI, e.g. "ai_r1c1"
prefix	This submodel to look in (i.e. "top.")
suffix	The suffix for algebras ("_std")
digits	= 2
verbose	= FALSE

Value

- the CI string, e.g. ".73 [-.2, .98]"

References

- http://tbates.github.io, https://github.com/tbates/umx

See Also

Other Miscellaneous Functions: umxEval; umxJiggle; umx_APA_pval; umx_add_variances; umx_apply; umx_check_model; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_check; umx_default_option; umx_explode; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_cores; umx_get_optimizer; umx_has_CIs; umx_has_been_run; umx_has_means; umx_has_square_brackets; umx_is_MxMatrix; umx_is_MxModel; umx_is_RAM; umx_is_cov; umx_is_endogenous; umx_is_exogenous; umx_is_ordered; umx_msg; umx_names; umx_object_as_str; umx_paste_names; umx_print; umx_rename; umx_reorder; umx_rot; umx_set_cores; umx_set_optimizer; umx_string_to_algebra; umx_trim; umx, umx-package

Examples

```
## Not run:
umx_get_CI_as_APA_string(fit_IP, cellLabel = "ai_r1c1", prefix = "top.", suffix = "_std")

## End(Not run)
```

---

umx_get_cores *umx_get_cores*

Description

Gets the number of cores (threads) used by OpenMx.

Usage

```
umx_get_cores(model = NULL)
```

Arguments

model	an (optional) model to get from. If left NULL, the global option is returned

Value

- number of cores

References

- http://tbates.github.io, https://github.com/tbates/umx, http://openmx.psyc.virginia.edu

See Also

Other Miscellaneous Functions: umxEval; umxJiggle; umx_APA_pval; umx_add_variances; umx_apply; umx_check_model; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_check; umx_default_option; umx_explode; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_optimizer; umx_has_CIs; umx_has_been_run; umx_has_means; umx_has_square_brackets; umx_is_MxMatrix; umx_is_MxModel; umx_is_RAM; umx_is_cov; umx_is_endogenous; umx_is_exogenous; umx_is_ordered; umx_msg; umx_names; umx_object_as_str; umx_paste_names; umx_print; umx_rename; umx_reorder; umx_rot; umx_set_cores; umx_set_optimizer; umx_string_to_algebra; umx_trim; umx, umx-package

Examples

```
library(OpenMx)
manifests = c("mpg", "disp", "gear")
m1 <- mxModel("ind", type = "RAM",
manifestVars = manifests,
mxPath(from = manifests, arrows = 2),
mxPath(from = "one", to = manifests),
mxData(mtcars[, manifests], type = "raw")
)
oldCores = umx_get_cores()   # get current default value
umx_set_cores(model = m1)    # set to default (max - 1)
umx_get_cores(model = m1)    # show new value
umx_set_cores() # set to default (max - 1)
umx_get_cores()              # show new value
umx_set_cores(oldCores)      # reset to old value
```

---

umx_get_optimizer *umx_get_optimizer*

Description

get the optimizer in OpenMx

Usage

```
umx_get_optimizer(model = NULL)
```

Arguments

model	(optional) model to get from. If left NULL, the global option is returned

Value

- the optimizer - a string

References

- http://tbates.github.io, https://github.com/tbates/umx

See Also

Other Miscellaneous Functions: umxEval; umxJiggle; umx_APA_pval; umx_add_variances; umx_apply; umx_check_model; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_check; umx_default_option; umx_explode; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_cores; umx_has_CIs; umx_has_been_run; umx_has_means; umx_has_square_brackets; umx_is_MxMatrix; umx_is_MxModel; umx_is_RAM; umx_is_cov; umx_is_endogenous; umx_is_exogenous; umx_is_ordered; umx_msg; umx_names; umx_object_as_str; umx_paste_names; umx_print; umx_rename; umx_reorder; umx_rot; umx_set_cores; umx_set_optimizer; umx_string_to_algebra; umx_trim; umx, umx-package

Examples

```
umx_get_optimizer() # current optimizer
```

---

umx_grep *umx_grep*

Description

Search for text. Will search names if given a data.frame, or strings if given a vector of strings. NOTE: Handy feature is that this can search the labels of data imported from SPSS

Usage

```
umx_grep(df, grepString, output = c("both", "label", "name"),
  ignore.case = TRUE, useNames = FALSE)
```

Arguments

df	The data.frame or string to search
grepString	the search string
output	the column name, the label, or both (default)
ignore.case	whether to be case sensitive or not (default TRUE = ignore case)
useNames	whether to search the names as well as the labels (for SPSS files with label metadata)

Value

- list of matched column names and/or labels

References

- http://www.github.com/tbates/umx

See Also

Other Miscellaneous Utility Functions: qm; umx_check_OS; umx_find_object; umx_pb_note; umx, umx-package

Examples

```
umx_grep(mtcars, "hp", output="both", ignore.case= TRUE)
umx_grep(c("hp", "ph"), "hp")
umx_grep(mtcars, "^h.*", output="both", ignore.case= TRUE)
## Not run:
umx_grep(spss_df, "labeltext", output = "label")
umx_grep(spss_df, "labeltext", output = "name")

## End(Not run)
```

---

umx_has_been_run *umx_has_been_run*

Description

check if an mxModel has been run or not

Usage

```
umx_has_been_run(model, stop = FALSE)
```

Arguments

model	The mxModel you want to check has been run
stop	Whether to stop if the model has not been run (defaults to FALSE)

Value

- boolean

References

- http://www.github.com/tbates/umx

See Also

- umxLabel, umxRun, umxValues

Other Miscellaneous Functions: umxEval; umxJiggle; umx_APA_pval; umx_add_variances; umx_apply; umx_check_model; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_check; umx_default_option; umx_explode; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_cores; umx_get_optimizer; umx_has_CIs; umx_has_means; umx_has_square_brackets; umx_is_MxMatrix; umx_is_MxModel; umx_is_RAM; umx_is_cov; umx_is_endogenous; umx_is_exogenous; umx_is_ordered; umx_msg; umx_names; umx_object_as_str; umx_paste_names;

umx_print; umx_rename; umx_reorder; umx_rot; umx_set_cores; umx_set_optimizer; umx_string_to_algebra; umx_trim; umx, umx-package

Examples

```
require(OpenMx)
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
m1 <- mxModel("One Factor", type = "RAM",
manifestVars = manifests, latentVars = latents,
mxPath(from = latents, to = manifests),
mxPath(from = manifests, arrows = 2),
mxPath(from = latents, arrows = 2, free = FALSE, values = 1.0),
mxData(cov(demoOneFactor), type = "cov", numObs = 500)
)
m1 = umxRun(m1, setLabels = TRUE, setValues = TRUE)
umx_has_been_run(m1)
```

---

umx_has_CIs *umx_has_CIs*

Description

A utility function to return a binary answer to the question "does this mxModel have confidence intervals?"

Usage

```
umx_has_CIs(model, check = c("both", "intervals", "output"))
```

Arguments

model	The mxModel to check for presence of CIs
check	What to check for: "intervals" requested, "output" present, or "both". Defaults to "both"

Value

- TRUE or FALSE

References

- http://www.github.com/tbates/umx/

See Also

- mxCI, umxCI, umxRun

Other Miscellaneous Functions: umxEval; umxJiggle; umx_APA_pval; umx_add_variances; umx_apply; umx_check_model; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_check; umx_default_option; umx_explode; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_cores; umx_get_optimizer; umx_has_been_run; umx_has_means; umx_has_square_brackets; umx_is_MxMatrix; umx_is_MxModel; umx_is_RAM; umx_is_cov; umx_is_endogenous; umx_is_exogenous; umx_is_ordered; umx_msg; umx_names; umx_object_as_str; umx_paste_names; umx_print; umx_rename; umx_reorder; umx_rot; umx_set_cores; umx_set_optimizer; umx_string_to_algebra; umx_trim; umx, umx-package

Examples

```
require(OpenMx)
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
m1 <- mxModel("One Factor", type = "RAM",
manifestVars = manifests, latentVars = latents,
mxPath(from = latents, to = manifests),
mxPath(from = manifests, arrows = 2),
mxPath(from = latents, arrows = 2, free = FALSE, values = 1.0),
mxData(cov(demoOneFactor), type = "cov", numObs = 500)
)
m1 = umxRun(m1, setLabels = TRUE, setValues = TRUE)
umx_has_CIs(m1) # FALSE: no CIs and no output
m1 = mxModel(m1, mxCI("G_to_x1"))
umx_has_CIs(m1, check = "intervals") # TRUE intervals set
umx_has_CIs(m1, check = "output")  # FALSE not yet run
m1 = mxRun(m1)
umx_has_CIs(m1, check = "output")  # Still FALSE: Set and Run
m1 = mxRun(m1, intervals = TRUE)
umx_has_CIs(m1, check = "output")  # TRUE: Set, and Run with intervals = T
```

---

umx_has_means *umx_has_means*

Description

A utility function to return a binary answer to the question "does this mxModel have a means model?"

Usage

```
umx_has_means(model)
```

Arguments

model	The mxModel to check for presence of means

Value

- TRUE or FALSE

References

- http://www.github.com/tbates/umx/

See Also

Other Miscellaneous Functions: umxEval; umxJiggle; umx_APA_pval; umx_add_variances; umx_apply; umx_check_model; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_check; umx_default_option; umx_explode; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_cores; umx_get_optimizer; umx_has_CIs; umx_has_been_run; umx_has_square_brackets; umx_is_MxMatrix; umx_is_MxModel; umx_is_RAM; umx_is_cov; umx_is_endogenous; umx_is_exogenous; umx_is_ordered; umx_msg; umx_names; umx_object_as_str; umx_paste_names; umx_print; umx_rename; umx_reorder; umx_rot; umx_set_cores; umx_set_optimizer; umx_string_to_algebra; umx_trim; umx, umx-package

Examples

```
require(OpenMx)
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
m1 <- mxModel("One Factor", type = "RAM",
manifestVars = manifests, latentVars = latents,
mxPath(from = latents, to = manifests),
mxPath(from = manifests, arrows = 2),
mxPath(from = latents, arrows = 2, free = FALSE, values = 1.0),
mxData(cov(demoOneFactor), type = "cov", numObs = 500)
)
umx_has_means(m1)
m1 <- mxModel(m1,
mxPath(from = "one", to = manifests),
mxData(demoOneFactor, type = "raw")
)
umx_has_means(m1)
m1 = umxRun(m1, setLabels = TRUE, setValues = TRUE)
umx_has_means(m1)
```

---

umx_has_square_brackets

umx_has_square_brackets

Description

Helper function, checking if a label has sqaure brackets

Usage

```
umx_has_square_brackets(input)
```

Arguments

input	The label to check for square brackets (string input)

Value

- boolean

References

- http://www.github.com/tbates/umx

See Also

- umxLabel, umxRun, umxValues

Other Miscellaneous Functions: umxEval; umxJiggle; umx_APA_pval; umx_add_variances; umx_apply; umx_check_model; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_check; umx_default_option; umx_explode; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_cores; umx_get_optimizer; umx_has_CIs; umx_has_been_run; umx_has_means; umx_is_MxMatrix; umx_is_MxModel; umx_is_RAM; umx_is_cov; umx_is_endogenous; umx_is_exogenous; umx_is_ordered; umx_msg; umx_names; umx_object_as_str; umx_paste_names; umx_print; umx_rename; umx_reorder; umx_rot; umx_set_cores; umx_set_optimizer; umx_string_to_algebra; umx_trim; umx, umx-package

Examples

```
umx_has_square_brackets("[hello]")
umx_has_square_brackets("goodbye")
```

---

umx_is_cov *umx_is_cov*

Description

test if a data frame or matrix is cov or cor data, or is likely to be raw...

Usage

```
umx_is_cov(data = NULL, boolean = FALSE, verbose = FALSE)
```

Arguments

data	dataframe to test
boolean	whether to return the type ("cov") or a boolean (default = string)
verbose	How much feedback to give (default = FALSE)

Value

- "raw", "cor", or "cov", or, if boolean= T, then T | F

References

- http://www.github.com/tbates/umx

See Also

- umxLabel, umxRun, umxValues

Other Miscellaneous Functions: umxEval; umxJiggle; umx_APA_pval; umx_add_variances; umx_apply; umx_check_model; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_check; umx_default_option; umx_explode; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_cores; umx_get_optimizer; umx_has_CIs; umx_has_been_run; umx_has_means; umx_has_square_brackets; umx_is_MxMatrix; umx_is_MxModel; umx_is_RAM; umx_is_endogenous; umx_is_exogenous; umx_is_ordered; umx_msg; umx_names; umx_object_as_str; umx_paste_names; umx_print; umx_rename; umx_reorder; umx_rot; umx_set_cores; umx_set_optimizer; umx_string_to_algebra; umx_trim; umx, umx-package

Examples

```
df = cov(mtcars)
umx_is_cov(df)
df = cor(mtcars)
umx_is_cov(df)
umx_is_cov(df, boolean = TRUE)
umx_is_cov(mtcars, boolean = TRUE)
```

---

umx_is_endogenous *umx_is_endogenous*

Description

Return a list of all the endogenous variables (variables with at least one incoming single-arrow path) in a model.

Usage

```
umx_is_endogenous(model, manifests_only = TRUE)
```

Arguments

model	an mxModel from which to get endogenous variables
manifests_only	Whether to check only manifests (default = TRUE)

Value

- list of endogenous variables

References

- http://tbates.github.io, https://github.com/tbates/umx, http://openmx.psyc.virginia.edu

See Also

Other Miscellaneous Functions: umxEval; umxJiggle; umx_APA_pval; umx_add_variances; umx_apply; umx_check_model; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_check; umx_default_option; umx_explode; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_cores; umx_get_optimizer; umx_has_CIs; umx_has_been_run; umx_has_means; umx_has_square_brackets; umx_is_MxMatrix; umx_is_MxModel; umx_is_RAM; umx_is_cov; umx_is_exogenous; umx_is_ordered; umx_msg; umx_names; umx_object_as_str; umx_paste_names; umx_print; umx_rename; umx_reorder; umx_rot; umx_set_cores; umx_set_optimizer; umx_string_to_algebra; umx_trim; umx, umx-package

Examples

```
require(OpenMx)
data(demoOneFactor)
m1 <- umxRAM("One Factor", data = mxData(cov(demoOneFactor), type = "cov", numObs = 500),
mxPath(from = "g", to = names(demoOneFactor))
)
umx_is_endogenous(m1, manifests_only = TRUE)
umx_is_endogenous(m1, manifests_only = FALSE)
```

---

umx_is_exogenous *umx_is_exogenous*

Description

Return a list of all the exogenous variables (variables with no incoming single-arrow path) in a model.

Usage

```
umx_is_exogenous(model, manifests_only = TRUE)
```

Arguments

model	an mxModel from which to get exogenous variables
manifests_only	Whether to check only manifests (default = TRUE)

Value

- list of exogenous variables

References

- http://tbates.github.io, https://github.com/tbates/umx, http://openmx.psyc.virginia.edu

See Also

Other Miscellaneous Functions: umxEval; umxJiggle; umx_APA_pval; umx_add_variances; umx_apply; umx_check_model; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_check; umx_default_option; umx_explode; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_cores; umx_get_optimizer; umx_has_CIs; umx_has_been_run; umx_has_means; umx_has_square_brackets; umx_is_MxMatrix; umx_is_MxModel; umx_is_RAM; umx_is_cov; umx_is_endogenous; umx_is_ordered; umx_msg; umx_names; umx_object_as_str; umx_paste_names; umx_print; umx_rename; umx_reorder; umx_rot; umx_set_cores; umx_set_optimizer; umx_string_to_algebra; umx_trim; umx, umx-package

Examples

```
require(OpenMx)
data(demoOneFactor)
m1 <- umxRAM("One Factor", data = mxData(cov(demoOneFactor), type = "cov", numObs = 500),
mxPath(from = "g", to = names(demoOneFactor))
)
umx_is_exogenous(m1, manifests_only = TRUE)
umx_is_exogenous(m1, manifests_only = FALSE)
```

---

umx_is_MxMatrix *umx_is_MxMatrix*

Description

Utility function returning a binary answer to the question "Is this an OpenMx mxMatrix?"

Usage

```
umx_is_MxMatrix(obj)
```

Arguments

obj	an object to be tested to see if it is an OpenMx mxMatrix

Value

- Boolean

References

- http://www.github.com/tbates/umx

See Also

Other Miscellaneous Functions: umxEval; umxJiggle; umx_APA_pval; umx_add_variances; umx_apply; umx_check_model; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_check; umx_default_option; umx_explode; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_cores; umx_get_optimizer; umx_has_CIs; umx_has_been_run; umx_has_means; umx_has_square_brackets; umx_is_MxModel; umx_is_RAM; umx_is_cov; umx_is_endogenous; umx_is_exogenous; umx_is_ordered; umx_msg; umx_names; umx_object_as_str; umx_paste_names; umx_print; umx_rename; umx_reorder; umx_rot; umx_set_cores; umx_set_optimizer; umx_string_to_algebra; umx_trim; umx, umx-package

Examples

```
x = mxMatrix(name = "eg", type = "Full", nrow = 3, ncol = 3, values = .3)
if(umx_is_MxMatrix(x)){
message("nice OpenMx matrix!")
}
```

---

umx_is_MxModel *umx_is_MxModel*

Description

Utility function returning a binary answer to the question "Is this an OpenMx model?"

Usage

```
umx_is_MxModel(obj)
```

Arguments

obj	an object to be tested to see if it is an OpenMx mxModel

Value

- Boolean

References

- http://www.github.com/tbates/umx

See Also

- mxModel

Other Miscellaneous Functions: umxEval; umxJiggle; umx_APA_pval; umx_add_variances; umx_apply; umx_check_model; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_check; umx_default_option; umx_explode; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_cores; umx_get_optimizer; umx_has_CIs; umx_has_been_run; umx_has_means; umx_has_square_brackets; umx_is_MxMatrix; umx_is_RAM; umx_is_cov; umx_is_endogenous;

umx_is_exogenous; umx_is_ordered; umx_msg; umx_names; umx_object_as_str; umx_paste_names; umx_print; umx_rename; umx_reorder; umx_rot; umx_set_cores; umx_set_optimizer; umx_string_to_algebra; umx_trim; umx, umx-package

Examples

```
m1 = mxModel("test")
if(umx_is_MxModel(m1)){
message("nice OpenMx model!")
}
```

---

umx_is_ordered *umx_is_ordered*

Description

Return the names of any ordinal variables in a dataframe

Usage

```
umx_is_ordered(df, names = FALSE, strict = TRUE, binary.only = FALSE,
  ordinal.only = FALSE, continuous.only = FALSE)
```

Arguments

df	an data.frame to look in for ordinal variables
names	whether to return the names of ordinal variables, or a binary (T,F) list (default = FALSE)
strict	whether to stop when unordered factors are found (default = TRUE)
binary.only	only count binary factors (2-levels) (default = FALSE)
ordinal.only	only count ordinal factors (3 or more levels) (default = FALSE)
continuous.only	use with names = TRUE to get the names of the continuous variables

Value

- vector of variable names or Booleans

References

- http://www.github.com/tbates/umx

See Also

Other Miscellaneous Functions: umxEval; umxJiggle; umx_APA_pval; umx_add_variances; umx_apply; umx_check_model; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_check; umx_default_option; umx_explode; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_cores; umx_get_optimizer; umx_has_CIs; umx_has_been_run; umx_has_means; umx_has_square_brackets; umx_is_MxMatrix; umx_is_MxModel; umx_is_RAM; umx_is_cov; umx_is_endogenous; umx_is_exogenous; umx_msg; umx_names; umx_object_as_str; umx_paste_names; umx_print; umx_rename; umx_reorder; umx_rot; umx_set_cores; umx_set_optimizer; umx_string_to_algebra; umx_trim; umx, umx-package

Examples

```
tmp = mtcars
tmp$cyl = ordered(mtcars$cyl) # ordered factor
tmp$vs = ordered(mtcars$vs) # binary factor
umx_is_ordered(tmp) # numeric indices
umx_is_ordered(tmp, names = TRUE)
umx_is_ordered(tmp, names = TRUE, binary.only = TRUE)
umx_is_ordered(tmp, names = TRUE, ordinal.only = TRUE)
umx_is_ordered(tmp, names = TRUE, continuous.only = TRUE)
umx_is_ordered(tmp, continuous.only = TRUE)
isContinuous = !umx_is_ordered(tmp)
tmp$gear = factor(mtcars$gear) # unordered factor
# nb: Factors are not necessarily ordered! By default unordered factors cause an message...
## Not run:
tmp$cyl = factor(mtcars$cyl)
umx_is_ordered(tmp, names=TRUE)

## End(Not run)
```

---

umx_is_RAM *umx_is_RAM*

Description

Utility function returning a binary answer to the question "Is this a RAM model?"

Usage

```
umx_is_RAM(obj)
```

Arguments

obj	an object to be tested to see if it is an OpenMx RAM mxModel

Value

- Boolean

References

- http://www.github.com/tbates/umx

See Also

Other Miscellaneous Functions: umxEval; umxJiggle; umx_APA_pval; umx_add_variances; umx_apply; umx_check_model; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_check; umx_default_option; umx_explode; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_cores; umx_get_optimizer; umx_has_CIs; umx_has_been_run; umx_has_means; umx_has_square_brackets; umx_is_MxMatrix; umx_is_MxModel; umx_is_cov; umx_is_endogenous; umx_is_exogenous; umx_is_ordered; umx_msg; umx_names; umx_object_as_str; umx_paste_names; umx_print; umx_rename; umx_reorder; umx_rot; umx_set_cores; umx_set_optimizer; umx_string_to_algebra; umx_trim; umx, umx-package

Examples

```
require(OpenMx)
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
m1 <- mxModel("One Factor", type = "RAM",
manifestVars = manifests, latentVars = latents,
mxPath(from = latents, to = manifests),
mxPath(from = manifests, arrows = 2),
mxPath(from = latents, arrows = 2, free = FALSE, values = 1.0),
mxData(cov(demoOneFactor), type = "cov", numObs = 500)
)
m1 = umxRun(m1, setLabels = TRUE, setValues = TRUE)
umxSummary(m1, show = "std")
if(umx_is_RAM(m1)){
message("nice RAM model!")
}
if(!umx_is_RAM(m1)){
message("model must be a RAM model")
}
```

---

umx_lower2full *umx_lower2full*

Description

Take a lower triangle of data (either from a "lower" mxMatrix, or as you might typed in a journal article) and turn it into a full matrix.

Usage

```
umx_lower2full(lower.data, diag = FALSE, byrow = TRUE)
```

Arguments

lower.data	An mxMatrix
diag	A boolean noting whether the lower matrix includes the diagonal
byrow	Whether the matrix is to be filled by row or by column

Value

- mxMatrix

References

- http://www.github.com/tbates/umx

See Also

Other Miscellaneous Data Functions: umxHetCor; umxPadAndPruneForDefVars; umx_as_numeric; umx_cont_2_ordinal; umx_cov2raw; umx_make_bin_cont_pair_data; umx_merge_CIs; umx_read_lower; umx_residualize; umx_round; umx_scale_wide_twin_data; umx_scale; umx_swap_a_block; umx, umx-package

Examples

```
tmpn = c("ROccAsp", "REdAsp", "FOccAsp", "FEdAsp", "RParAsp",
         "RIQ", "RSES", "FSES", "FIQ", "FParAsp")
tmp = matrix(nrow = 10, ncol = 10, byrow = TRUE, dimnames = list(tmpn,tmpn), data =
c(1.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,  0.0000, 0.0000, 0,
0.6247, 1.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,  0.0000, 0.0000, 0,
0.3269, 0.3669, 1.0000, 0.0000, 0.0000, 0.0000, 0.0000,  0.0000, 0.0000, 0,
0.4216, 0.3275, 0.6404, 1.0000, 0.0000, 0.0000, 0.0000,  0.0000, 0.0000, 0,
0.2137, 0.2742, 0.1124, 0.0839, 1.0000, 0.0000, 0.0000,  0.0000, 0.0000, 0,
0.4105, 0.4043, 0.2903, 0.2598, 0.1839, 1.0000, 0.0000,  0.0000, 0.0000, 0,
0.3240, 0.4047, 0.3054, 0.2786, 0.0489, 0.2220, 1.0000,  0.0000, 0.0000, 0,
0.2930, 0.2407, 0.4105, 0.3607, 0.0186, 0.1861, 0.2707,  1.0000, 0.0000, 0,
0.2995, 0.2863, 0.5191, 0.5007, 0.0782, 0.3355, 0.2302,  0.2950, 1.0000, 0,
0.0760, 0.0702, 0.2784, 0.1988, 0.1147, 0.1021, 0.0931, -0.0438, 0.2087, 1)
)
umx_lower2full(tmp)
tmp = c(
c(1.0000,
0.6247, 1.0000,
0.3269, 0.3669, 1.0000,
0.4216, 0.3275, 0.6404, 1.0000,
0.2137, 0.2742, 0.1124, 0.0839, 1.0000,
0.4105, 0.4043, 0.2903, 0.2598, 0.1839, 1.0000,
0.3240, 0.4047, 0.3054, 0.2786, 0.0489, 0.2220, 1.0000,
0.2930, 0.2407, 0.4105, 0.3607, 0.0186, 0.1861, 0.2707,  1.0000,
0.2995, 0.2863, 0.5191, 0.5007, 0.0782, 0.3355, 0.2302,  0.2950, 1.0000,
0.0760, 0.0702, 0.2784, 0.1988, 0.1147, 0.1021, 0.0931, -0.0438, 0.2087, 1)
)
umx_lower2full(tmp)
tmp = c(
```

```
c(0.6247,
0.3269, 0.3669,
0.4216, 0.3275, 0.6404,
0.2137, 0.2742, 0.1124, 0.0839,
0.4105, 0.4043, 0.2903, 0.2598, 0.1839,
0.3240, 0.4047, 0.3054, 0.2786, 0.0489, 0.2220,
0.2930, 0.2407, 0.4105, 0.3607, 0.0186, 0.1861, 0.2707,
0.2995, 0.2863, 0.5191, 0.5007, 0.0782, 0.3355, 0.2302,  0.2950,
0.0760, 0.0702, 0.2784, 0.1988, 0.1147, 0.1021, 0.0931, -0.0438, 0.2087)
)
umx_lower2full(tmp, diag = FALSE)
```

`umx_make_bin_cont_pair_data` *umx_make_bin_cont_pair_data*

Description

Takes a dataframe of left-censored variables (vars with a floor effect) and does two things to it: 1. It creates new binary (1/0) copies of each column (with the suffix "bin"). These contain 0 where the variable is below the minimum and NA otherwise. 2. In each existing variable, it sets all instances of min for that var to NA

Usage

```
umx_make_bin_cont_pair_data(data, vars = NULL, suffixes = NULL)
```

Arguments

`data`	A `data.frame` to convert
`vars`	The variables to process
`suffixes`	Suffixes if the data are family (wide, more than one persona on a row)

Value

- copy of the dataframe with new binary variables and censoring

References

- https://github.com/tbates/umx, https://tbates.github.io, http://openmx.psyc.virginia.edu

See Also

- umxACE

Other Miscellaneous Data Functions: umxHetCor; umxPadAndPruneForDefVars; umx_as_numeric; umx_cont_2_ordinal; umx_cov2raw; umx_lower2full; umx_merge_CIs; umx_read_lower; umx_residualize; umx_round; umx_scale_wide_twin_data; umx_scale; umx_swap_a_block; umx, umx-package

Examples

```
df = umx_make_bin_cont_pair_data(mtcars, vars = c("mpg"))
str(df)
df[order(df$mpg), c(1,12)]
# Introduce a floor effect
tmp = mtcars; tmp$mpg[tmp$mpg<=15]=15
tmp$mpg_T1 = tmp$mpg_T2 = tmp$mpg
df = umx_make_bin_cont_pair_data(tmp, vars = c("mpg"), suffixes = c("_T1", "_T2"))
df[order(df$mpg), 12:15]
```

---

umx_means *umx_means*

Description

Helper to get means from a df that might contain ordered data. Factor means are set to "ordVar"

Usage

```
umx_means(df, ordVar = 0, na.rm = TRUE)
```

Arguments

df	a dataframe of raw data from which to get variances.
ordVar	value to return for the means of factor data = 0
na.rm	passed to mean - defaults to "na.rm"

Value

- frame of means

See Also

Other Miscellaneous Stats Functions: reliability; umxCov2cor; umx_cor; umx, umx-package

Examples

```
tmp = mtcars[,1:4]
tmp$cyl = ordered(mtcars$cyl) # ordered factor
tmp$hp  = ordered(mtcars$hp)  # binary factor
umx_means(tmp, ordVar = 0, na.rm = TRUE)
```

umx_merge_CIs *umx_merge_CIs*

Description

if you compute some CIs in one model and some in another (copy of the same model, perhaps to get some parallelism), this is a simple helper to cludge them together.

Usage

```
umx_merge_CIs(m1, m2)
```

Arguments

m1	first copy of the model
m2	second copy of the model

Value

- mxModel

References

- http://www.github.com/tbates/umx

See Also

Other Miscellaneous Data Functions: umxHetCor; umxPadAndPruneForDefVars; umx_as_numeric; umx_cont_2_ordinal; umx_cov2raw; umx_lower2full; umx_make_bin_cont_pair_data; umx_read_lower; umx_residualize; umx_round; umx_scale_wide_twin_data; umx_scale; umx_swap_a_block; umx, umx-package

Examples

```
## Not run:
umx_merge_CIs(m1, m2)
# TODO remove duplicates...
# TODO (check they are the same as well!)
# TODO Support arbitrarily long list of input models with ...
# TODO check the models are the same, with same fit
# TODO check the models have CIs

## End(Not run)
```

umx_move_file *umx_move_file*

Description

move files. On OS X, the function can access the current frontmost Finder window. The file moves are fast and, because you can use regular expressions, powerful

Usage

```
umx_move_file(baseFolder = NA, findStr = NULL, fileNameList = NA,
  destFolder = NA, test = TRUE, overwrite = FALSE)
```

Arguments

baseFolder	The folder to search in. If set to "Finder" (and you are on OS X) it will use the current frontmost Finder window. If it is blank, a choose folder dialog will be thrown.
findStr	= regex string select files to move (WARNING: NOT IMPLEMENTED YET)
fileNameList	List of files to move
destFolder	Folder to move files into
test	Boolean determining whether to change the names, or just report on what would have happened
overwrite	Boolean determining whether to overwrite files or not (default = FALSE (safe))

Value

-

See Also

- umx_rename_file, file.rename

Other Miscellaneous File Functions: dl_from_dropbox; umx_open; umx_rename_file; umx, umx-package

Examples

```
## Not run:
base = "/Users/tim/Music/iTunes/iTunes Music/"
dest = "/Users/tim/Music/iTunes/iTunes Music/Music/"
umx_move_file(baseFolder = base, fileNameList = toMove, destFolder = dest, test= FALSE)

## End(Not run)
```

umx_msg *umx_msg*

Description

Helper function to make dumping "ObjectName has the value: <objectvalue>" easy

Usage

```
umx_msg(x)
```

Arguments

x	the thing you want to print

Value

- NULL

References

- http://tbates.github.io, https://github.com/tbates/umx, http://openmx.psyc.virginia.edu

See Also

Other Miscellaneous Functions: umxEval; umxJiggle; umx_APA_pval; umx_add_variances; umx_apply; umx_check_model; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_check; umx_default_option; umx_explode; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_cores; umx_get_optimizer; umx_has_CIs; umx_has_been_run; umx_has_means; umx_has_square_brackets; umx_is_MxMatrix; umx_is_MxModel; umx_is_RAM; umx_is_cov; umx_is_endogenous; umx_is_exogenous; umx_is_ordered; umx_names; umx_object_as_str; umx_paste_names; umx_print; umx_rename; umx_reorder; umx_rot; umx_set_cores; umx_set_optimizer; umx_string_to_algebra; umx_trim; umx, umx-package

Examples

```
a = "brian"
umx_msg(a)
a = c("brian", "sally", "jane")
umx_msg(a)
```

umx_names *umx_names*

Description

Convenient equivalent of grep("fa[rl].*", names(df), value=T, ignore.case=T)

Usage

```
umx_names(df, pattern = ".*", ignore.case = TRUE, perl = FALSE,
  value = TRUE, fixed = FALSE, useBytes = FALSE, invert = FALSE)
```

Arguments

df	dataframe to get names from
pattern	= "find.*"
ignore.case	default = TRUE (opposite default to grep)
perl	= FALSE
value	= default = TRUE (opposite default to grep)
fixed	= FALSE
useBytes	= FALSE
invert	= FALSE

Value

- vector of matches

References

- http://tbates.github.io, https://github.com/tbates/umx

See Also

Other Miscellaneous Functions: umxEval; umxJiggle; umx_APA_pval; umx_add_variances; umx_apply; umx_check_model; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_check; umx_default_option; umx_explode; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_cores; umx_get_optimizer; umx_has_CIs; umx_has_been_run; umx_has_means; umx_has_square_brackets; umx_is_MxMatrix; umx_is_MxModel; umx_is_RAM; umx_is_cov; umx_is_endogenous; umx_is_exogenous; umx_is_ordered; umx_msg; umx_object_as_str; umx_paste_names; umx_print; umx_rename; umx_reorder; umx_rot; umx_set_cores; umx_set_optimizer; umx_string_to_algebra; umx_trim; umx, umx-package

Examples

```
umx_names(mtcars, "mpg") #"mpg" "cyl" "disp" "hp" "drat" "wt" "qsec" "vs" "am" "gear" "carb"
umx_names(mtcars, "^d") # "disp", drat
umx_names(mtcars, "r[ab]") # "drat", "carb"
```

umx_object_as_str *umx_object_as_str*

Description

Utility to return an object's name as a string

Usage

```
umx_object_as_str(x)
```

Arguments

x	an object

Value

- name as string

References

- http://www.github.com/tbates/umx

See Also

Other Miscellaneous Functions: umxEval; umxJiggle; umx_APA_pval; umx_add_variances; umx_apply; umx_check_model; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_check; umx_default_option; umx_explode; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_cores; umx_get_optimizer; umx_has_CIs; umx_has_been_run; umx_has_means; umx_has_square_brackets; umx_is_MxMatrix; umx_is_MxModel; umx_is_RAM; umx_is_cov; umx_is_endogenous; umx_is_exogenous; umx_is_ordered; umx_msg; umx_names; umx_paste_names; umx_print; umx_rename; umx_reorder; umx_rot; umx_set_cores; umx_set_optimizer; umx_string_to_algebra; umx_trim; umx, umx-package

Examples

```
umx_object_as_str(mtcars) # "mtcars"
```

umx_open *umx_open*

Description

Open a file or folder. So far only works on OS X

Usage

```
umx_open(filepath = getwd())
```

Arguments

filepath	The file to open

Value

-

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

Other Miscellaneous File Functions: dl_from_dropbox; umx_move_file; umx_rename_file; umx, umx-package

Examples

```
## Not run:
umx_open(getwd())

## End(Not run)
```

umx_paste_names *umx_paste_names*

Description

Helper to add suffixes to names: useful for expanding twin vars like "bmi" into c("bmi_T1", "bmi_T2") Use textConstant to turning add a constant"E_T1", by adding "_T" and 1.

Usage

```
umx_paste_names(varNames, textConstant = "", suffixes)
```

Arguments

varNames	a list of base names, e.g c("bmi", "IQ")
textConstant	The suffix added to all names, e.g. "_T" (default is "")
suffixes	a list of terminal suffixes differentiating the var names (e.g c("1", "2"))

Value

- vector of suffixed var names, i.e., c("a_T1", "b_T1", "a_T2", "b_T2")

References

- http://tbates.github.io, https://github.com/tbates/umx

See Also

Other Miscellaneous Functions: umxEval; umxJiggle; umx_APA_pval; umx_add_variances; umx_apply; umx_check_model; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_check; umx_default_option; umx_explode; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_cores; umx_get_optimizer; umx_has_CIs; umx_has_been_run; umx_has_means; umx_has_square_brackets; umx_is_MxMatrix; umx_is_MxModel; umx_is_RAM; umx_is_cov; umx_is_endogenous; umx_is_exogenous; umx_is_ordered; umx_msg; umx_names; umx_object_as_str; umx_print; umx_rename; umx_reorder; umx_rot; umx_set_cores; umx_set_optimizer; umx_string_to_algebra; umx_trim; umx, umx-package

Examples

```
umx_paste_names("bmi", "_T", 1:2)
umx_paste_names("bmi", suffixes = c("_T1", "_T2"))
varNames = umx_paste_names(c("N", "E", "O", "A", "C"), "_T", 1:2)
```

---

umx_pb_note *umx_pb_note*

Description

Use the pushbullet service to push a note. You can also initialise this service by providing your autho_key one time

Usage

```
umx_pb_note(title = "test", body = "default body", auth_key = NA)
```

Arguments

title	of the note
body	of the note
auth_key	optional authkey (default = NA, set to value of your key to store key.

Details

If you supply auth_key, It will be writen to "~/.pushbulletkey" umx_pb_note(auth_key="mykeystring") once it exists there, you dont need to store it in code, so code is sharable.

You can get your autho code at https://www.pushbullet.com/account

Note: You can show the existing stored key using "GET"

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

- umx_msg

Other Miscellaneous Utility Functions: qm; umx_check_OS; umx_find_object; umx_grep; umx, umx-package

Examples

```
## Not run:
umx_pb_note("done!", umx_time(m1))

## End(Not run)
```

---

umx_print *umx_print*

Description

A helper to aid the interpretability of printed tables from OpenMx (and elsewhere). Its most useful characteristic is allowing you to change how NA and zero appear. By default, Zeros have the decimals suppressed, and NAs are suppressed altogether.

Usage

```
umx_print(x, digits = getOption("digits"), quote = FALSE, na.print = "",
  zero.print = "0", justify = "none", file = c(NA, "tmp.html"),
  suppress = NULL, ...)
```

Arguments

x	A data.frame to print
digits	The number of decimal places to print (defaults to getOption("digits")
quote	Parameter passed to print (defaults to FALSE)
na.print	String to replace NA with (default to blank "")
zero.print	String to replace 0.000 with (defaults to "0")

justify	Parameter passed to print (defaults to "none")
file	whether to write to a file (defaults to NA (no file). Use "tmp.html" to open as tables in browser.
suppress	minimum numeric value to print (default = NULL, print all values, no matter how small)
...	Optional parameters for print

See Also

- print

Other Miscellaneous Functions: umxEval; umxJiggle; umx_APA_pval; umx_add_variances; umx_apply; umx_check_model; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_check; umx_default_option; umx_explode; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_cores; umx_get_optimizer; umx_has_CIs; umx_has_been_run; umx_has_means; umx_has_square_brackets; umx_is_MxMatrix; umx_is_MxModel; umx_is_RAM; umx_is_cov; umx_is_endogenous; umx_is_exogenous; umx_is_ordered; umx_msg; umx_names; umx_object_as_str; umx_paste_names; umx_rename; umx_reorder; umx_rot; umx_set_cores; umx_set_optimizer; umx_string_to_algebra; umx_trim; umx, umx-package

Other Reporting Functions: umxStandardizeACE; umx_APA_CI; umx_APA_pval; umx_aggregate; umx_show; umx_time; umx, umx-package

Examples

```
umx_print(mtcars[1:10,], digits = 2, zero.print = ".", justify = "left")
## Not run:
umx_print(model)
umx_print(mtcars[1:10,], file = "Rout.html")

## End(Not run)
```

---

umx_RAM_ordinal_objective

umx_RAM_ordinal_objective

Description

umx_RAM_ordinal_objective builds an appropriate thresholds matrix It also sets latent means and variances to 0 and 1 respectively.

Usage

```
umx_RAM_ordinal_objective(df, deviationBased = TRUE, droplevels = TRUE,
  verbose = FALSE)
```

Arguments

df	Dataframe to make a threshold matrix for
deviationBased	whether to use the deviation system to ensure order thresholds (default = TRUE)
droplevels	whether to also drop unused levels (default = TRUE)
verbose	whether to say what the function is doing (default = FALSE)

Details

TODO: more detail about what we are doing here

Value

- mxModel

References

- http://www.github.com/tbates/umx

See Also

- umxLabel, umxRun, umxValues

Other Advanced Helpers: umx, umx-package

Examples

```
## Not run:
model = umx_RAM_ordinal_objective(model)

## End(Not run)
```

---

umx_read_lower *Read lower-triangle of data matrix from console or file*

Description

umx_read_lower will read a lower triangle of data, either from the console, or from file, and return a full matrix, optionally coerced to positive definite. This is useful, especially when copying data from a paper that includes just the lower triangle of a correlation matrix.

Usage

```
umx_read_lower(file = "", diag = TRUE, names = as.character(paste("X",
  1:n, sep = "")), ensurePD = FALSE)
```

Arguments

file	Path to a file to read (Default "" will read from user input)
diag	Whether the data unclude the diagonal or not: Defaults to TRUE
names	The default names for the variables. Defaults to as.character(paste("X", 1:n, sep=""))
ensurePD	Whether to coerce the resultant matrix to positive definite (Defaults to FALSE)

Value

- matrix

References

- https://github.com/tbates/umx, https://tbates.github.io

See Also

Other Miscellaneous Data Functions: umxHetCor; umxPadAndPruneForDefVars; umx_as_numeric; umx_cont_2_ordinal; umx_cov2raw; umx_lower2full; umx_make_bin_cont_pair_data; umx_merge_CIs; umx_residualize; umx_round; umx_scale_wide_twin_data; umx_scale; umx_swap_a_block; umx, umx-package

Examples

```
require(umx) # for umxRAM
## Not run:
df = umx_read_lower(file = "", diag = F, ensurePD=TRUE)
0.38
0.86 0.30
0.42 0.12 0.27
0.66 0.21 0.38 0.18
0.80 0.13 0.50 0.25 0.43
0.19 0.11 0.19 0.12 -0.06 0.22
0.27 0.09 0.33 0.05 -0.04 0.28 .73
0.52 0.17 0.38 0.37 0.39 0.44 0.18 0.13

IQtests = c("brainstorm", "matrix", "moral", "shopping", "typing")
n       = c("C", IQtests, "avgIQ", "maxIQ", "video")

dimnames(df) = list(n,n)

m1 = umxRAM("wooley", data = mxData(df, type="cov", numObs = 90),
umxPath("g", to = IQtests),
umxPath(var = "g", fixedAt=1),
umxPath(var = IQtests)
)
summary(m1)

## End(Not run)
```

umx_rename *umx_rename*

Description

Returns a dataframe with variables renamed as desired. Unlike some functions, it checks that the variables exist, and that the new names are not already used.

Usage

```
umx_rename(x, replace = NULL, old = NULL, grep = NULL, test = FALSE)
```

Arguments

x	the dataframe in which to rename variables
replace	If used alone, a named collection of c(oldName = "newName") pairs OR, if "old" is a list of existing names, the list of new names) OR, if "grep" is a regular expression, the replace string)
old	Optional list of old names that will be found and replaced by the contents of replace. Defaults to NULL
grep	Optional grep string. Matches will be replaced using replace as the replace string. Defaults to NULL
test	whether to report a "dry run" - and not actually change anything (defaults to false)

Details

note: to use replace list, you must say c(old = "new"), not c(old -> "new")

Value

- dataframe with columns renamed.

See Also

Other Miscellaneous Functions: umxEval; umxJiggle; umx_APA_pval; umx_add_variances; umx_apply; umx_check_model; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_check; umx_default_option; umx_explode; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_cores; umx_get_optimizer; umx_has_CIs; umx_has_been_run; umx_has_means; umx_has_square_brackets; umx_is_MxMatrix; umx_is_MxModel; umx_is_RAM; umx_is_cov; umx_is_endogenous; umx_is_exogenous; umx_is_ordered; umx_msg; umx_names; umx_object_as_str; umx_paste_names; umx_print; umx_reorder; umx_rot; umx_set_cores; umx_set_optimizer; umx_string_to_algebra; umx_trim; umx, umx-package

Examples

```
# Re-name "cyl" to "cylinder"
x = mtcars
x = umx_rename(x, replace = c(cyl = "cylinder"))
# alternate style
x = umx_rename(x, old = c("disp"), replace = c("displacement"))
umx_check_names("displacement", data = x, die = TRUE)
# This will warn that "disp" doesn't exist (anymore)
x = umx_rename(x, old = c("disp"), replace = c("displacement"))
x = umx_rename(x, grep = "lacement", replace = "") # use grep to set it back
umx_names(x, "^d")
```

---

umx_rename_file *umx_rename_file*

Description

rename files. On OS X, the function can access the current frontmost Finder window. The file renaming is fast and, because you can use regular expressions, powerful

Usage

```
umx_rename_file(findStr = NA, replaceStr = NA, baseFolder = "Finder",
  listPattern = NA, test = TRUE, overwrite = FALSE)
```

Arguments

findStr	The (regex) string to find, i.e., "c[ao]t"
replaceStr	The (regex) replacement string "\1 are not dogs"
baseFolder	The folder to search in. If set to "Finder" (and you are on OS X) it will use the current frontmost Finder window. If it is blank, a choose folder dialog will be thrown.
listPattern	A pre-filter for files
test	Boolean determining whether to change files on disk, or just report on what would have happened (Defaults to test = TRUE)
overwrite	Boolean determining if an existing file will be overwritten (Defaults to the safe FALSE)

Value

-

References

- http://www.github.com/tbates/umx

See Also

- grep, umxRun, umxValues

Other Miscellaneous File Functions: dl_from_dropbox; umx_move_file; umx_open; umx, umx-package

Examples

```
## Not run:
umx_rename_file(baseFolder = "~/Downloads/", findStr = "", replaceStr = "", test = TRUE)
umx_rename_file("[Ss]eason +([0-9]+)", replaceStr="S\1", baseFolder = "Finder", test = TRUE)

## End(Not run)
```

---

umx_reorder *umx_reorder*

Description

Reorder the variables in a correlation matrix. Can also remove one or more variables from a matrix using this function

Usage

```
umx_reorder(old, newOrder)
```

Arguments

old	a square matrix of correlation or covariances to reorder
newOrder	The order you'd like the variables to be in

Value

- the re-ordered (and/or resized) matrix

References

- http://www.github.com/tbates/umx

See Also

- umxLabel, umxRun, umxValues

Other Miscellaneous Functions: umxEval; umxJiggle; umx_APA_pval; umx_add_variances; umx_apply; umx_check_model; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_check; umx_default_option; umx_explode; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_cores; umx_get_optimizer; umx_has_CIs; umx_has_been_run; umx_has_means; umx_has_square_brackets; umx_is_MxMatrix; umx_is_MxModel; umx_is_RAM; umx_is_cov; umx_is_endogenous; umx_is_exogenous; umx_is_ordered; umx_msg; umx_names; umx_object_as_str; umx_paste_names; umx_print; umx_rename; umx_rot; umx_set_cores; umx_set_optimizer; umx_string_to_algebra; umx_trim; umx, umx-package

Examples

```
oldMatrix = cov(mtcars)
umx_reorder(oldMatrix, newOrder = c("mpg", "cyl", "disp")) # first 3
umx_reorder(oldMatrix, newOrder = c("hp", "disp", "cyl")) # subset and reordered
```

`umx_residualize` *umx_residualize*

Description

Return one or more variables residualised against covs.

Usage

```
umx_residualize(var, covs = NULL, suffixes = NULL, data)
```

Arguments

`var`	The base name of the variable you want to residualize. Alternatively, a regression `formula` containing var on the lhs, and covs on the rhs
`covs`	Covariates to residualize on.
`suffixes`	Suffixes that identify the variable for each twin, i.e. c("_T1", "_T2") Up to you to check all variables are present!
`data`	The dataframe containing all the variables

Details

This is the same as:

```
tmp <- residuals(lm(var ~ cov1 + cov2, data = data, na.action = na.exclude))
```

Optionally, this also works on wide (ie., twin) data. Just supply suffixes to identify the paired-wide columns (see examples)

Value

- dataframe with var residualized in place (i.e under its original column name)

References

- http://tbates.github.io, https://github.com/tbates/umx

See Also

Other Miscellaneous Data Functions: `umxHetCor`; `umxPadAndPruneForDefVars`; `umx_as_numeric`; `umx_cont_2_ordinal`; `umx_cov2raw`; `umx_lower2full`; `umx_make_bin_cont_pair_data`; `umx_merge_CIs`; `umx_read_lower`; `umx_round`; `umx_scale_wide_twin_data`; `umx_scale`; `umx_swap_a_block`; `umx`, `umx-package`

Examples

```
library(formula.tools)
tmp = mtcars
# Residualise mpg on cylinders and displacement
r1 = umx_residualize("mpg", c("cyl", "disp"), data = tmp)$mpg
r2 = residuals(lm(mpg ~ cyl + disp, data = tmp, na.action = na.exclude))
all(r1 == r2)
# plot(r1 ~ r2)
# formula interface
r1 = umx_residualize(mpg ~ cyl + I(cyl^2) + disp, data = tmp)$mpg
# Same again, but now on wide data (i.e. with family data on each row)
tmp$mpg_T1  = tmp$mpg_T2  = tmp$mpg
tmp$cyl_T1  = tmp$cyl_T2  = tmp$cyl
tmp$disp_T1 = tmp$disp_T2 = tmp$disp
umx_residualize("mpg", c("cyl", "disp"), c("_T1", "_T2"), data = tmp)
```

---

umx_rot *umx_rot*

Description

rotate a vector (default, rotate by 1)

Usage

```
umx_rot(vec)
```

Arguments

vec	vector to rotate

Value

- mxModel

References

- http://tbates.github.io, https://github.com/tbates/umx, http://openmx.psyc.virginia.edu

See Also

- umxLabel, umxRun, umxStart

Other Miscellaneous Functions: umxEval; umxJiggle; umx_APA_pval; umx_add_variances; umx_apply; umx_check_model; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_check; umx_default_option; umx_explode; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_cores; umx_get_optimizer; umx_has_CIs; umx_has_been_run; umx_has_means; umx_has_square_brackets; umx_is_MxMatrix; umx_is_MxModel; umx_is_RAM;

umx_is_cov; umx_is_endogenous; umx_is_exogenous; umx_is_ordered; umx_msg; umx_names; umx_object_as_str; umx_paste_names; umx_print; umx_rename; umx_reorder; umx_set_cores; umx_set_optimizer; umx_string_to_algebra; umx_trim; umx, umx-package

Examples

```
umx_rot(1:10)
umx_rot(c(3,4,5,6,7))
# [1] 4 5 6 7 3
```

---

umx_round *umx_round*

Description

A version of round() which works on dataframes that contain non-numeric data (or data that cannot be coerced to numeric) Helpful for dealing with table output that mixes numeric and string types.

Usage

```
umx_round(df, digits = getOption("digits"), coerce = FALSE)
```

Arguments

df	a dataframe to round in
digits	how many digits to round to (defaults to getOption("digits"))
coerce	whether to make the column numeric if it is not (default = FALSE)

Value

- mxModel

References

- http://www.github.com/tbates/umx

See Also

Other Miscellaneous Data Functions: umxHetCor; umxPadAndPruneForDefVars; umx_as_numeric; umx_cont_2_ordinal; umx_cov2raw; umx_lower2full; umx_make_bin_cont_pair_data; umx_merge_CIs; umx_read_lower; umx_residualize; umx_scale_wide_twin_data; umx_scale; umx_swap_a_block; umx, umx-package

Examples

```
head(umx_round(mtcars, coerce = FALSE))
head(umx_round(mtcars, coerce = TRUE))
```

umx_scale *umx_scale*

Description

Scale data columns, skipping ordinal

Usage

```
umx_scale(df, varsToScale = NULL, coerce = FALSE)
```

Arguments

df	a dataframe to scale
varsToScale	(leave blank for all)
coerce	Whether to coerce non-numerics to numeric (Defaults to FALSE)

Value

- new dataframe with scaled variables

References

- http://www.github.com/tbates/umx

See Also

- umxLabel, umxRun, umxValues

Other Miscellaneous Data Functions: umxHetCor; umxPadAndPruneForDefVars; umx_as_numeric; umx_cont_2_ordinal; umx_cov2raw; umx_lower2full; umx_make_bin_cont_pair_data; umx_merge_CIs; umx_read_lower; umx_residualize; umx_round; umx_scale_wide_twin_data; umx_swap_a_block; umx, umx-package

Examples

```
data(twinData)
df = umx_scale(twinData, varsToScale = NULL)
plot(wt1 ~ wt2, data= df)
```

umx_scale_wide_twin_data *umx_scale_wide_twin_data*

Description

Scale wide data across all cases: currently twins

Usage

```
umx_scale_wide_twin_data(varsToScale, suffixes, df)
```

Arguments

varsToScale	the base names of the variables ("weight" etc)
suffixes	the suffix that distinguishes each case (T1, T2 etc.)
df	a wide dataframe

Value

- new dataframe with scaled variables

References

- http://www.github.com/tbates/umx

See Also

- umxLabel, umxRun, umxValues

Other Miscellaneous Data Functions: umxHetCor; umxPadAndPruneForDefVars; umx_as_numeric; umx_cont_2_ordinal; umx_cov2raw; umx_lower2full; umx_make_bin_cont_pair_data; umx_merge_CIs; umx_read_lower; umx_residualize; umx_round; umx_scale; umx_swap_a_block; umx, umx-package

Examples

```
data(twinData)
df = umx_scale_wide_twin_data(twinData, varsToScale = c("ht", "wt"), suffixes = c("1","2") )
plot(wt1 ~ wt2, data = df)
```

umx_set_checkpoint *umx_set_checkpoint*

Description

Set the checkpoint status for a model or global options

Usage

```
umx_set_checkpoint(interval = 1, units = c("evaluations", "iterations",
  "minutes"), prefix = "", directory = getwd(), model = NULL)
```

Arguments

interval	How many units between checkpoints: Default = 1. A value of zero sets always to 'No' (i.e., do not checkpoint all models during optimization)
units	units to count in: Default unit is 'evaluations' ('minutes' is also legal)
prefix	string prefix to add to all checkpoint filenames (default = "")
directory	a directory, i.e "~/Desktop" (defaults to getwd())
model	(optional) model to set options in (default = NULL)

Value

- mxModel if provided

References

- http://tbates.github.io, https://github.com/tbates/umx, http://openmx.psyc.virginia.edu

See Also

Other Miscellaneous Functions: umxEval; umxJiggle; umx_APA_pval; umx_add_variances; umx_apply; umx_check_model; umx_check_multi_core; umx_check; umx_default_option; umx_explode; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_cores; umx_get_optimizer; umx_has_CIs; umx_has_been_run; umx_has_means; umx_has_square_brackets; umx_is_MxMatrix; umx_is_MxModel; umx_is_RAM; umx_is_cov; umx_is_endogenous; umx_is_exogenous; umx_is_ordered; umx_msg; umx_names; umx_object_as_str; umx_paste_names; umx_print; umx_rename; umx_reorder; umx_rot; umx_set_cores; umx_set_optimizer; umx_string_to_algebra; umx_trim; umx, umx-package

Examples

```
umx_set_checkpoint(interval = 1, "evaluations", dir = "~/Desktop/")
# turn off checkpointing with interval = 0
umx_set_checkpoint(interval = 0)
umx_set_checkpoint(2, "evaluations", prefix="SNP_1")
require(OpenMx)
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
m1 <- umxRAM("One Factor", mxData(cov(demoOneFactor), type = "cov", numObs = 500),
umxPath(latents, to = manifests),
umxPath(var = manifests),
umxPath(var = latents, fixedAt = 1.0)
)
m1 = umx_set_checkpoint(model = m1)
m1 = mxRun(m1)
umx_checkpoint(0)
```

---

umx_set_cores *umx_set_cores*

Description

set the number of cores (threads) used by OpenMx

Usage

```
umx_set_cores(cores = parallel::detectCores(), model = NULL)
```

Arguments

cores	number of cores to use (defaults to max - 1 to preserve UI responsiveness)
model	an (optional) model to set. If left NULL, the global option is updated.

Value

- NULL

References

- http://tbates.github.io, https://github.com/tbates/umx, http://openmx.psyc.virginia.edu

See Also

Other Miscellaneous Functions: umxEval; umxJiggle; umx_APA_pval; umx_add_variances; umx_apply; umx_check_model; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_check; umx_default_option; umx_explode; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_cores; umx_get_optimizer; umx_has_CIs; umx_has_been_run; umx_has_means; umx_has_square_brackets; umx_is_MxMatrix; umx_is_MxModel; umx_is_RAM; umx_is_cov; umx_is_endogenous; umx_is_exogenous; umx_is_ordered; umx_msg; umx_names; umx_object_as_str; umx_paste_names; umx_print; umx_rename; umx_reorder; umx_rot; umx_set_optimizer; umx_string_to_algebra; umx_trim; umx, umx-package

Examples

```
library(OpenMx)
manifests = c("mpg", "disp", "gear")
m1 <- mxModel("ind", type = "RAM",
manifestVars = manifests,
mxPath(from = manifests, arrows = 2),
mxPath(from = "one", to = manifests),
mxData(mtcars[, manifests], type = "raw")
)
oldCores <- umx_get_cores() # get global value
umx_set_cores() # set to default (max)
umx_set_cores(parallel::detectCores() - 1) # set to max - 1
umx_get_cores() # show new value
umx_set_cores(1, m1)  # set m1 useage to 1 core
umx_get_cores(model = m1)  # show new value
```

---

umx_set_optimizer *umx_set_optimizer*

Description

set the optimizer in OpenMx

Usage

```
umx_set_optimizer(opt = c("NPSOL", "SLSQP", "CSOLNP"))
```

Arguments

opt	defaults to "NPSOL". Current alternatives are "SLSQP" and "CSOLNP"

Value

-

References

- http://tbates.github.io, https://github.com/tbates/umx

See Also

Other Miscellaneous Functions: umxEval; umxJiggle; umx_APA_pval; umx_add_variances; umx_apply; umx_check_model; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_check; umx_default_option; umx_explode; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_cores; umx_get_optimizer; umx_has_CIs; umx_has_been_run; umx_has_means; umx_has_square_brackets; umx_is_MxMatrix; umx_is_MxModel; umx_is_RAM; umx_is_cov; umx_is_endogenous; umx_is_exogenous; umx_is_ordered; umx_msg; umx_names; umx_object_as_str; umx_paste_names; umx_print; umx_rename; umx_reorder; umx_rot; umx_set_cores; umx_string_to_algebra; umx_trim; umx, umx-package

Examples

```
library(umx)
old = umx_get_optimizer() # get the existing state
umx_set_optimizer("SLSQP") # update globally
umx_set_optimizer(old) # set back
```

---

umx_show *umx_show*

Description

Show matrix contents. The user can select values, free, and/or labels, and which matrices to display

Usage

```
umx_show(model, what = c("values", "free", "labels", "nonzero_or_free"),
  matrices = c("S", "A"), digits = 2)
```

Arguments

model	an mxModel to show data from
what	legal options are "values" (default), "free", or "labels")
matrices	to show (default is c("S", "A"))
digits	precision to report, defaults to rounding to 2 decimal places

Value

- mxModel

References

- http://tbates.github.io, https://github.com/tbates/umx, http://openmx.psyc.virginia.edu

See Also

- umxLabel, umxRun, umxStart

Other Reporting Functions: umxStandardizeACE; umx_APA_CI; umx_APA_pval; umx_aggregate; umx_print; umx_time; umx, umx-package

Examples

```
require(OpenMx)
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
m1 <- mxModel("One Factor", type = "RAM",
manifestVars = manifests, latentVars = latents,
mxPath(from = latents, to = manifests),
mxPath(from = manifests, arrows = 2),
mxPath(from = latents, arrows = 2, free = FALSE, values = 1.0),
mxData(cov(demoOneFactor), type = "cov", numObs = 500)
)
m1 = umxRun(m1, setLabels = TRUE, setValues = TRUE)
umx_show(m1)
umx_show(m1, digits = 3)
umx_show(m1, matrices = "S")
umx_show(m1, what = "free")
umx_show(m1, what = "labels")
umx_show(m1, what = "free", "A")
```

---

umx_string_to_algebra *umx_string_to_algebra*

Description

This is useful because it lets you use paste() and rep() to quickly and easily insert values from R variables into the string, then parse the string as an mxAlgebra argument. The use case this time was to include a matrix exponent (that is A

Usage

```
umx_string_to_algebra(algString, name = NA, dimnames = NA)
```

Arguments

algString	a string to turn into an algebra
name	of the returned algebra
dimnames	of the returned algebra

Value

- mxAlgebra

References

- http://www.github.com/tbates/umx

See Also

- umxLabel, umxRun, umxValues

Other Miscellaneous Functions: umxEval; umxJiggle; umx_APA_pval; umx_add_variances; umx_apply; umx_check_model; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_check; umx_default_option; umx_explode; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_cores; umx_get_optimizer; umx_has_CIs; umx_has_been_run; umx_has_means; umx_has_square_brackets; umx_is_MxMatrix; umx_is_MxModel; umx_is_RAM; umx_is_cov; umx_is_endogenous; umx_is_exogenous; umx_is_ordered; umx_msg; umx_names; umx_object_as_str; umx_paste_names; umx_print; umx_rename; umx_reorder; umx_rot; umx_set_cores; umx_set_optimizer; umx_trim; umx, umx-package

Examples

```
## Not run:
alg = umx_string_to_algebra(paste(rep("A", nReps), collapse = " %*% "), name = "test_case")

## End(Not run)
```

---

umx_swap_a_block *umx_swap_a_block*

Description

Swap a block of rows of a datset between two lists variables (typically twin 1 and twin2)

Usage

```
umx_swap_a_block(theData, rowSelector, T1Names, T2Names)
```

Arguments

theData	a data frame to swap within
rowSelector	rows to swap amongst columns
T1Names	the first set of columns
T2Names	the second set of columns

Value

- dataframe

See Also

- subset

Other Miscellaneous Data Functions: umxHetCor; umxPadAndPruneForDefVars; umx_as_numeric; umx_cont_2_ordinal; umx_cov2raw; umx_lower2full; umx_make_bin_cont_pair_data; umx_merge_CIs; umx_read_lower; umx_residualize; umx_round; umx_scale_wide_twin_data; umx_scale; umx, umx-package

Examples

```
test = data.frame(
a = paste0("a", 1:10),
b = paste0("b", 1:10),
c = paste0("c", 1:10),
d = paste0("d", 1:10), stringsAsFactors = FALSE)
umx_swap_a_block(test, rowSelector = c(1,2,3,6), T1Names = "b", T2Names = "c")
umx_swap_a_block(test, rowSelector = c(1,2,3,6), T1Names = c("a","c"), T2Names = c("b","d"))
```

---

umx_time *umx_time*

Description

A function to compactly report how long a model took to execute. Comes with some preset styles User can set the format with C-style string formatting.

Usage

```
umx_time(model, formatStr = c("simple", "std", "custom %H %M %OS3"),
  tz = "GMT")
```

Arguments

model	An mxModel from which to get the elapsed time
formatStr	A format string, defining how to show the time (defaults to human readable)
tz	time zone in which the model was executed (defaults to "GMT")

Details

The default is "simple", which gives only the biggest unit used. i.e., "x seconds" for times under 1 minute. "std" shows time in the format adopted in OpenMx 2.0 e.g. "Wall clock time (HH:MM:SS.hh): 00:00:01.16"

If a list of models is provided, time deltas will also be reported.

If the model hasn't been run, this function will run it for you.

Value

- invisible time string

References

- http://www.github.com/tbates/umx

See Also

Other Reporting Functions: umxStandardizeACE; umx_APA_CI; umx_APA_pval; umx_aggregate; umx_print; umx_show; umx, umx-package

Examples

```
require(OpenMx)
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
m1 <- mxModel("One Factor", type = "RAM",
manifestVars = manifests, latentVars = latents,
mxPath(from = latents, to = manifests),
mxPath(from = manifests, arrows = 2),
mxPath(from = latents, arrows = 2, free = FALSE, values = 1.0),
mxData(cov(demoOneFactor), type = "cov", numObs = 500)
)
m1 = umxRun(m1, setLabels = TRUE, setValues = TRUE)
umx_time(m1)
m2 = umxRun(m1)
umx_time(c(m1, m2))
```

---

umx_trim *umx_trim*

Description

returns string w/o leading or trailing whitespace

Usage

```
umx_trim(string)
```

Arguments

string	to trim

Value

- string

References

- http://tbates.github.io, https://github.com/tbates/umx, http://openmx.psyc.virginia.edu

See Also

Other Miscellaneous Functions: umxEval; umxJiggle; umx_APA_pval; umx_add_variances; umx_apply; umx_check_model; umx_check_multi_core; umx_checkpoint, umx_set_checkpoint; umx_check; umx_default_option; umx_explode; umx_get_CI_as_APA_string; umx_get_bracket_addresses; umx_get_checkpoint; umx_get_cores; umx_get_optimizer; umx_has_CIs; umx_has_been_run; umx_has_means; umx_has_square_brackets; umx_is_MxMatrix; umx_is_MxModel; umx_is_RAM; umx_is_cov; umx_is_endogenous; umx_is_exogenous; umx_is_ordered; umx_msg; umx_names; umx_object_as_str; umx_paste_names; umx_print; umx_rename; umx_reorder; umx_rot; umx_set_cores; umx_set_optimizer; umx_string_to_algebra; umx, umx-package

Examples

```
umx_trim(" dog") # "dog"
umx_trim("dog ") # "dog"
umx_trim("\t dog \n") # "dog"
```

---

xmuHasSquareBrackets *xmuHasSquareBrackets*

Description

Tests if an input has square brackets

Usage

```
xmuHasSquareBrackets(input)
```

Arguments

input	an input to test

Value

- TRUE/FALSE

See Also

Other xmu internal not for end user: xmuLabel_MATRIX_Model; xmuLabel_Matrix; xmuLabel_RAM_Model; xmuMI; xmuMakeDeviationThresholdsMatrices; xmuMakeOneHeadedPathsFromPathList; xmuMakeTwoHeadedPathsFro
xmuMaxLevels; xmuMinLevels; xmuPropagateLabels; xmu_dot_make_paths; xmu_dot_make_residuals; xmu_start_value_list

Examples

```
xmuHasSquareBrackets("A[1,2]")
```

xmuLabel_Matrix *xmuLabel_Matrix (not a user function)*

Description

This function will label all the free parameters in an mxMatrix

Usage

```
xmuLabel_Matrix(mx_matrix = NA, baseName = NA, setfree = FALSE,
  drop = 0, jiggle = NA, boundDiag = NA, suffix = "", verbose = TRUE,
  labelFixedCells = FALSE, overRideExisting = FALSE)
```

Arguments

mx_matrix	an mxMatrix
baseName	A base name for the labels NA
setfree	Whether to set free cells FALSE
drop	What values to drop 0
jiggle	= whether to jiggle start values
boundDiag	whether to add bounds to the diagonal
suffix	a string to append to each label
verbose	how much feedback to give
labelFixedCells	= FALSE
overRideExisting	Whether to overRideExisting (Default FALSE)

Details

End users should just call umxLabel

Purpose: label the cells of an mxMatrix Detail: Defaults to the handy "matrixname_r1c1" where 1 is the row or column Use case: You should not use this: call umxLabel umx:::xmuLabel_Matrix(mxMatrix("Lower", 3, 3, values = 1, name = "a", byrow = TRUE), jiggle = .05, boundDiag = NA); umx:::xmuLabel_Matrix(mxMatrix("Full" , 3, 3, values = 1, name = "a", byrow = TRUE)); umx:::xmuLabel_Matrix(mxMatrix("Symm" , 3, 3, values = 1, name = "a", byrow = TRUE), jiggle = .05, boundDiag = NA); umx:::xmuLabel_Matrix(mxMatrix("Full" , 1, 1, values = 1, name = "a", labels= "data.a")); umx:::xmuLabel_Matrix(mxMatrix("Full" , 1, 1, values = 1, name = "a", labels= "data.a"), overRideExisting=TRUE); umx:::xmuLabel_Matrix(mxMatrix("Full" , 1, 1, values = 1, name = "a", labels= "test"), overRideExisting=TRUE); See also: fit2 = omxSetParameters(fit1, labels = "a_r1c1", free = FALSE, value = 0, name = "drop_a_row1_c1")

Value

- The labeled mxMatrix

See Also

Other xmu internal not for end user: xmuHasSquareBrackets; xmuLabel_MATRIX_Model; xmuLabel_RAM_Model; xmuMI; xmuMakeDeviationThresholdsMatrices; xmuMakeOneHeadedPathsFromPathList; xmuMakeTwoHeadedPathsFro xmuMaxLevels; xmuMinLevels; xmuPropagateLabels; xmu_dot_make_paths; xmu_dot_make_residuals; xmu_start_value_list

---

`xmuLabel_MATRIX_Model` *xmuLabel_MATRIX_Model (not a user function)*

Description

This function will label all the free parameters in a (non-RAM) OpenMx mxModel nb: We don't assume what each matrix is for. Instead, the function just sticks labels like "a_r1c1" into each cell i.e., matrixname _ r rowNumber c colNumber

Usage

```
xmuLabel_MATRIX_Model(model, suffix = "", verbose = TRUE)
```

Arguments

model	a matrix-style mxModel to label
suffix	a string to append to each label
verbose	how much feedback to give

Details

End users should just call umxLabel

Value

- The labeled mxModel

See Also

Other xmu internal not for end user: xmuHasSquareBrackets; xmuLabel_Matrix; xmuLabel_RAM_Model; xmuMI; xmuMakeDeviationThresholdsMatrices; xmuMakeOneHeadedPathsFromPathList; xmuMakeTwoHeadedPathsFro xmuMaxLevels; xmuMinLevels; xmuPropagateLabels; xmu_dot_make_paths; xmu_dot_make_residuals; xmu_start_value_list

Examples

```
require(OpenMx)
data(demoOneFactor)
m2 <- mxModel("One Factor",
mxMatrix("Full", 5, 1, values = 0.2, free = TRUE, name = "A"),
mxMatrix("Symm", 1, 1, values = 1, free = FALSE, name = "L"),
mxMatrix("Diag", 5, 5, values = 1, free = TRUE, name = "U"),
mxAlgebra(A %*% L %*% t(A) + U, name = "R"),
mxExpectationNormal("R", dimnames = names(demoOneFactor)),
mxFitFunctionML(),
mxData(cov(demoOneFactor), type = "cov", numObs = 500)
)
m3 = umx:::xmuLabel_MATRIX_Model(m2)
m4 = umx:::xmuLabel_MATRIX_Model(m2, suffix = "male")
# explore these with omxGetParameters(m4)
```

`xmuLabel_RAM_Model` *xmuLabel_RAM_Model (not a user function)*

Description

This function will label all the free parameters in a RAM `mxModel`

Usage

```
xmuLabel_RAM_Model(model, suffix = "", labelFixedCells = TRUE,
  overRideExisting = FALSE, verbose = FALSE)
```

Arguments

`model`	a RAM mxModel to label
`suffix`	a string to append to each label
`labelFixedCells`	Whether to labelFixedCells (Default TRUE)
`overRideExisting`	Whether to overRideExisting (Default FALSE)
`verbose`	how much feedback to give

Details

End users should just call `umxLabel`

Value

- The labeled `mxModel`

See Also

Other xmu internal not for end user: xmuHasSquareBrackets; xmuLabel_MATRIX_Model; xmuLabel_Matrix; xmuMI; xmuMakeDeviationThresholdsMatrices; xmuMakeOneHeadedPathsFromPathList; xmuMakeTwoHeadedPathsFro xmuMaxLevels; xmuMinLevels; xmuPropagateLabels; xmu_dot_make_paths; xmu_dot_make_residuals; xmu_start_value_list

---

`xmuMakeDeviationThresholdsMatrices`
Make deviation threshold matrics

Description

Purpose: return a mxRAMObjective(A = "A", S = "S", F = "F", M = "M", thresholds = "thresh"), mxData(df, type = "raw") usecase see: umxMakeThresholdMatrix

Usage

```
xmuMakeDeviationThresholdsMatrices(df, droplevels, verbose)
```

Arguments

`df`	a dataframe
`droplevels`	whether to droplevels or not
`verbose`	how verbose to be

Value

- list of matrices

See Also

Other xmu internal not for end user: xmuHasSquareBrackets; xmuLabel_MATRIX_Model; xmuLabel_Matrix; xmuLabel_RAM_Model; xmuMI; xmuMakeOneHeadedPathsFromPathList; xmuMakeTwoHeadedPathsFromPathList; xmuMaxLevels; xmuMinLevels; xmuPropagateLabels; xmu_dot_make_paths; xmu_dot_make_residuals; xmu_start_value_list

---

`xmuMakeOneHeadedPathsFromPathList`
xmuMakeOneHeadedPathsFromPathList

Description

Make one-headed paths

Usage

```
xmuMakeOneHeadedPathsFromPathList(sourceList, destinationList)
```

Arguments

`sourceList`	A sourceList
`destinationList`	A destinationList

Value

- added items

See Also

Other xmu internal not for end user: `xmuHasSquareBrackets`; `xmuLabel_MATRIX_Model`; `xmuLabel_Matrix`; `xmuLabel_RAM_Model`; `xmuMI`; `xmuMakeDeviationThresholdsMatrices`; `xmuMakeTwoHeadedPathsFromPathList`; `xmuMaxLevels`; `xmuMinLevels`; `xmuPropagateLabels`; `xmu_dot_make_paths`; `xmu_dot_make_residuals`; `xmu_start_value_list`

`xmuMakeThresholdsMatrices`
xmuMakeThresholdsMatrices (not a user function)

Description

You should not be calling this directly. This is not as reliable a strategy and likely to be superceeded...

Usage

```
xmuMakeThresholdsMatrices(df, droplevels = FALSE, verbose = FALSE)
```

Arguments

`df`	a `data.frame` containing the data for your `mxData` statement
`droplevels`	a binary asking if empty levels should be dropped (defaults to FALSE)
`verbose`	how much feedback to give (defaults to FALSE)

Value

- a list containing an mxMatrix called "thresh", an mxRAMObjective object, and an mxData object

References

- http://openmx.psyc.virginia.edu/

Examples

```
# x = mtcars
# x$cyl = mxFactor(x$cyl, levels = c(4,6,8))
# umx:::xmuMakeThresholdsMatrices(df = x, droplevels=FALSE, verbose= TRUE)
```

---

xmuMakeTwoHeadedPathsFromPathList

xmuMakeTwoHeadedPathsFromPathList

Description

Make two-headed paths

Usage

```
xmuMakeTwoHeadedPathsFromPathList(pathList)
```

Arguments

pathList	A pathlist

Value

- added items

See Also

Other xmu internal not for end user: xmuHasSquareBrackets; xmuLabel_MATRIX_Model; xmuLabel_Matrix; xmuLabel_RAM_Model; xmuMI; xmuMakeDeviationThresholdsMatrices; xmuMakeOneHeadedPathsFromPathList; xmuMaxLevels; xmuMinLevels; xmuPropagateLabels; xmu_dot_make_paths; xmu_dot_make_residuals; xmu_start_value_list

xmuMaxLevels *xmuMaxLevels*

Description

Get the max levels from df

Usage

```
xmuMaxLevels(df)
```

Arguments

df	Dataframe to search through

Value

- max number of levels in frame

See Also

Other xmu internal not for end user: xmuHasSquareBrackets; xmuLabel_MATRIX_Model; xmuLabel_Matrix; xmuLabel_RAM_Model; xmuMI; xmuMakeDeviationThresholdsMatrices; xmuMakeOneHeadedPathsFromPathList; xmuMakeTwoHeadedPathsFromPathList; xmuMinLevels; xmuPropagateLabels; xmu_dot_make_paths; xmu_dot_make_residuals; xmu_start_value_list

xmuMI *xmuMI*

Description

A function to compute and report modifications which would improve fit. You will probably use umxMI instead

Usage

```
xmuMI(model, vector = TRUE)
```

Arguments

model	an mxModel to derive modification indices for
vector	= Whether to report the results as a vector default = TRUE

See Also

Other xmu internal not for end user: xmuHasSquareBrackets; xmuLabel_MATRIX_Model; xmuLabel_Matrix; xmuLabel_RAM_Model; xmuMakeDeviationThresholdsMatrices; xmuMakeOneHeadedPathsFromPathList; xmuMakeTwoHeadedPathsFromPathList; xmuMaxLevels; xmuMinLevels; xmuPropagateLabels; xmu_dot_make_paths; xmu_dot_make_residuals; xmu_start_value_list

---

`xmuMinLevels` *xmuMinLevels*

Description

Get the min levels from df

Usage

```
xmuMinLevels(df)
```

Arguments

df	Dataframe to search through

Value

- min number of levels in frame

See Also

Other xmu internal not for end user: xmuHasSquareBrackets; xmuLabel_MATRIX_Model; xmuLabel_Matrix; xmuLabel_RAM_Model; xmuMI; xmuMakeDeviationThresholdsMatrices; xmuMakeOneHeadedPathsFromPathList; xmuMakeTwoHeadedPathsFromPathList; xmuMaxLevels; xmuPropagateLabels; xmu_dot_make_paths; xmu_dot_make_residuals; xmu_start_value_list

---

`xmuPropagateLabels` *xmuPropagateLabels (not a user function)*

Description

You should be calling umxLabel. This function is called by xmuLabel_MATRIX_Model

Usage

```
xmuPropagateLabels(model, suffix = "", verbose = TRUE)
```

Arguments

model	a model to label
suffix	a string to append to each label
verbose	whether to say what is being done

Value

- mxModel

See Also

Other xmu internal not for end user: xmuHasSquareBrackets; xmuLabel_MATRIX_Model; xmuLabel_Matrix; xmuLabel_RAM_Model; xmuMI; xmuMakeDeviationThresholdsMatrices; xmuMakeOneHeadedPathsFromPathList; xmuMakeTwoHeadedPathsFromPathList; xmuMaxLevels; xmuMinLevels; xmu_dot_make_paths; xmu_dot_make_residuals; xmu_start_value_list

Examples

```
require(OpenMx)
data(demoOneFactor)
latents  = c("G")
manifests = names(demoOneFactor)
m1 <- mxModel("One Factor", type = "RAM",
manifestVars = manifests, latentVars = latents,
mxPath(from = latents, to = manifests),
mxPath(from = manifests, arrows = 2),
mxPath(from = latents, arrows = 2, free = FALSE, values = 1.0),
mxData(cov(demoOneFactor), type = "cov", numObs = 500)
)
m1 = umx:::xmuPropagateLabels(m1, suffix = "MZ")
```

---

xmu_dot_make_paths *xmu_dot_make_paths (not for end users)*

Description

xmu_dot_make_paths (not for end users)

Usage

```
xmu_dot_make_paths(mxMat, stringIn, heads = NULL, showFixed = TRUE,
  comment = "More paths", showResiduals = TRUE, pathLabels = "labels",
  digits = 2)
```

Arguments

mxMat	an MxMatrix
stringIn	input string
heads	1 or 2
showFixed	to show fixed values or not
comment	a comment to include
showResiduals	Whether to show residuals
pathLabels	labels
digits	how many digits to report

Value

- string

See Also

Other xmu internal not for end user: xmuHasSquareBrackets; xmuLabel_MATRIX_Model; xmuLabel_Matrix; xmuLabel_RAM_Model; xmuMI; xmuMakeDeviationThresholdsMatrices; xmuMakeOneHeadedPathsFromPathList; xmuMakeTwoHeadedPathsFromPathList; xmuMaxLevels; xmuMinLevels; xmuPropagateLabels; xmu_dot_make_residuals; xmu_start_value_list

---

xmu_dot_make_residuals

xmu_dot_make_residuals (not for end users)

Description

xmu_dot_make_residuals (not for end users)

Usage

```
xmu_dot_make_residuals(mxMat, showFixed = TRUE, digits = 2)
```

Arguments

mxMat	an MxMatrix
showFixed	to show fixed values or not
digits	how many digits to report

Value

- list of variance names and variances

See Also

Other xmu internal not for end user: xmuHasSquareBrackets; xmuLabel_MATRIX_Model; xmuLabel_Matrix; xmuLabel_RAM_Model; xmuMI; xmuMakeDeviationThresholdsMatrices; xmuMakeOneHeadedPathsFromPathList; xmuMakeTwoHeadedPathsFromPathList; xmuMaxLevels; xmuMinLevels; xmuPropagateLabels; xmu_dot_make_paths; xmu_start_value_list

---

`xmu_start_value_list` *Make start values*

Description

Purpose: Create startvalues for OpenMx paths use cases umx:::xmuStart_value_list(1) umxValues(1) # 1 value, varying around 1, with sd of .1 umxValues(1, n=letters) # length(letters) start values, with mean 1 and sd .1 umxValues(100, 15) # 1 start, with mean 100 and sd 15

Usage

```
xmu_start_value_list(mean = 1, sd = NA, n = 1)
```

Arguments

`mean`	the mean start value
`sd`	the sd of values
`n`	how many to generate

Value

- start value list

See Also

Other xmu internal not for end user: xmuHasSquareBrackets; xmuLabel_MATRIX_Model; xmuLabel_Matrix; xmuLabel_RAM_Model; xmuMI; xmuMakeDeviationThresholdsMatrices; xmuMakeOneHeadedPathsFromPathList; xmuMakeTwoHeadedPathsFromPathList; xmuMaxLevels; xmuMinLevels; xmuPropagateLabels; xmu_dot_make_paths; xmu_dot_make_residuals

Index