Package 'ctsemOMX'

July 13, 2026

Type Package

Title Continuous Time Structural Equation Modelling - Old 'OpenMx'-Based Version

Version 2.0.1

Date 2026-7-13

Description Original 'ctsem' (continuous time structural equation modelling) functionality, based on the 'OpenMx' software, as described in Driver, Oud, Voelkle (2017) <doi:10.18637/jss.v077.i05>, with updated details in vignette. Combines stochastic differential equations representing latent processes with structural equation measurement models. This package is maintained for consistency with the original 'ctsem' paper, but for the much newer and more capable 'ctsem' package, see <https://cran.r-project.org/package=ctsem>.

License GPL-3

Depends R (>= 3.5.0), OpenMx (>= 2.9.0)

URL https://github.com/cdriveraus/ctsemOMX

Imports data.table, expm, graphics, grDevices, Matrix, methods, plyr, stats, utils

Encoding UTF-8

LazyData true

ByteCompile true

Suggests knitr, testthat

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Charles Driver [aut, cre, cph], Manuel Voelkle [aut, cph], Han Oud [aut, cph]

Maintainer Charles Driver <charles.driver2@uzh.ch>

Repository CRAN

Date/Publication 2026-07-13 09:40:02 UTC

Contents

AnomAuth . . . 2
ctCI . . . 3
ctCompareExpected . . . 4
ctDeintervalise . . . 5
ctDiscretiseData . . . 5
ctExample1 . . . 6
ctExample1TIpred . . . 6
ctExample2 . . . 7
ctExample2level . . . 7
ctExample3 . . . 7
ctExample4 . . . 8
ctFit . . . 8
ctGenerate . . . 12
ctGenerateFromFit . . . 13
ctIndplot . . . 14
ctIntervalise . . . 15
ctLongToWide . . . 17
ctModel . . . 18
ctModelFromFit . . . 22
ctMultigroupFit . . . 23
ctPlot . . . 25
ctRefineTo . . . 26
ctstantestdat . . . 27
ctWideToLong . . . 27
datastructure . . . 28
longexample . . . 29
Oscillating . . . 29
plot.ctsemFit . . . 30
plot.ctsemMultigroupFit . . . 31
summary.ctsemFit . . . 32
summary.ctsemMultigroupFit . . . 33

Index **35**

`AnomAuth` *AnomAuth*

Description

A dataset containing panel data assessments of individuals Anomia and Authoritarianism.

Format

data frame with 2722 rows, 14 columns. Column Y1 represents anomia, Y2 Authoritarianism, dTx the time interval for measurement occasion x.

Source

Example panel data used in the original ctsem methodological papers.

`ctCI` *ctCI Computes confidence intervals on specified parameters / matrices for already fitted ctsem fit object.*

Description

ctCI Computes confidence intervals on specified parameters / matrices for already fitted ctsem fit object.

Usage

```
ctCI(ctfitobj, confidenceintervals, optimizer = "NPSOL", verbose = 0)
```

Arguments

`ctfitobj`	Already fit ctsem fit object (class: ctsemFit) to estimate confidence intervals for.
`confidenceintervals`	character vector of matrices and or parameters for which to estimate 95% confidence intervals for.
`optimizer`	character vector. Defaults to NPSOL (recommended), but other optimizers available within OpenMx (e.g. 'CSOLNP') may be specified.
`verbose`	Integer between 0 and 3 reflecting amount of output while calculating.

Details

If estimating for a multigroup model, specify confidence intervals as normal, e.g. `confidenceintervals = c('DRIFT', 'diffusion_Y1_Y1')` . The necessary group prefixes are added internally.

Value

ctfitobj, with confidence intervals included.

Examples

```
## Examples set to 'donttest' because they take longer than 5s.

data("ctExample3")
model <- ctModel(n.latent = 1, n.manifest = 3, Tpoints = 100,
 LAMBDA = matrix(c(1, "lambda2", "lambda3"), nrow = 3, ncol = 1),
 MANIFESTMEANS = matrix(c(0, "manifestmean2", "manifestmean3"), nrow = 3,
   ncol = 1))
fit <- ctFit(dat = ctExample3, ctmodelobj = model, objective = "Kalman",
 stationary = c("T0VAR"))
```

```
fit <- ctCI(fit, confidenceintervals = 'DRIFT')

summary(fit)$omxsummary$CI
```

`ctCompareExpected` *ctCompareExpected Compares model implied to observed means and covariances for panel data fit with ctsem.*

Description

ctCompareExpected Compares model implied to observed means and covariances for panel data fit with ctsem.

Usage

```
ctCompareExpected(
  fitobj,
  cov = TRUE,
  outputmatrices = FALSE,
  pause = TRUE,
  varlist = "all",
  ylim = c(-1, 1),
  ...
)
```

Arguments

`fitobj`	Fitted model object from OpenMx or ctsem.
`cov`	Logical. If TRUE, show covariance plots, if FALSE show correlations.
`outputmatrices`	if TRUE, output expected, observed, and residual correlation matrices as well as plots.
`pause`	if TRUE (default) output plots interactively, one at a time. If FALSE, output without stopping.
`varlist`	if "all" include all variables in dataset. Otherwise, specify numeric vector of variables to include.
`ylim`	vector of min and max Y axis limits for plot.
`...`	additional arguments passed to plot.

`ctDeintervalise` *ctDeintervalise*

Description

Converts intervals in ctsem long format data to absolute time

Usage

```
ctDeintervalise(datalong, id = "id", dT = "dT", startoffset = 0)
```

Arguments

`datalong`	data to use, in ctsem long format (attained via function ctWideToLong)
`id`	character string denoting column of data containing numeric identifier for each subject.
`dT`	character string denoting column of data containing time interval preceding observations in that row.
`startoffset`	Number of units of time to offset by when converting.

`ctDiscretiseData` *Discretise long format continuous time (ctsem) data to specific timestep.*

Description

Extends and rounds timing information so equal intervals, according to specified timestep, are achieved. NA's are inserted in other columns as necessary, any columns specified by TDpredNames or TIpredNames have zeroes rather than NA's inserted (because some estimation routines do not tolerate NA's in covariates).

Usage

```
ctDiscretiseData(
  dlong,
  timestep,
  timecol = "time",
  idcol = "id",
  TDpredNames = NULL,
  TIpredNames = NULL
)
```

Arguments

dlong	Long format data
timestep	Positive real value to discretise
timecol	Name of column containing absolute (not intervals) time information.
idcol	Name of column containing subject id variable.
TDpredNames	Vector of column names of any time dependent predictors
TIpredNames	Vector of column names of any time independent predictors

Value

long format ctsem data.

Examples

```
long <- ctDiscretiseData(dlong=ctstantestdat, timestep = .1,
TDpredNames=c('TD1'),TIpredNames=c('TI1','TI2','TI3'))
```

---

ctExample1 *ctExample1*

Description

Simulated example dataset for the ctsem package

Format

100 by 17 matrix containing containing ctsem wide format data. 6 measurement occasions of leisure time and happiness and 5 measurement intervals for each of 100 individuals.

---

ctExample1TIpred *ctExample1TIpred*

Description

Simulated example dataset for the ctsem package

Format

100 by 18 matrix containing containing ctsem wide format data. 6 measurement occasions of leisure time and happiness, 1 measurement of number of friends, and 5 measurement intervals for each of 100 individuals.

ctExample2 *ctExample2*

Description

Simulated example dataset for the ctsem package

Format

100 by 18 matrix containing containing ctsem wide format data. 8 measurement occasions of leisure time and happiness, 7 measurement occasions of a money intervention dummy, and 7 measurement intervals for each of 50 individuals.

ctExample2level *ctExample2level*

Description

Simulated example dataset for the ctsem package

Format

100 by 18 matrix containing ctsem wide format data. 8 measurement occasions of leisure time and happiness, 7 measurement occasions of a money intervention dummy, and 7 measurement intervals for each of 50 individuals.

ctExample3 *ctExample3*

Description

Simulated example dataset for the ctsem package

Format

1 by 399 matrix containing containing ctsem wide format data. 100 observations of variables Y1 and Y2 and 199 measurement intervals, for 1 subject.

ctExample4 *ctExample4*

Description

Simulated example dataset for the ctsem package

Format

20 by 79 matrix containing 20 observations of variables Y1, Y2, Y3, and 19 measurement intervals dTx, for each of 20 individuals.

ctFit *Fit a ctsem object*

Description

This function fits continuous time SEM models specified via `ctModel` to a dataset containing one or more subjects.

Usage

```
ctFit(
  dat,
  ctmodelobj,
  dataform = "auto",
  objective = "auto",
  stationary = c("T0TRAITEFFECT", "T0TIPREDEFFECT"),
  optimizer = "CSOLNP",
  retryattempts = 5,
  iterationSummary = FALSE,
  carefulFit = TRUE,
  carefulFitWeight = 100,
  showInits = FALSE,
  asymptotes = FALSE,
  meanIntervals = FALSE,
  crossEffectNegStarts = TRUE,
  fit = TRUE,
  nofit = FALSE,
  discreteTime = FALSE,
  verbose = 0,
  useOptimizer = TRUE,
  omxStartValues = NULL,
  transformedParams = TRUE,
  datawide = NA
)
```

Arguments

dat	the data you wish to fit a ctsem model to, in either wide format (one individual per row), or long format (one time point of one individual per row). See details.
ctmodelobj	the ctsem model object you wish to use, specified via the ctModel function.
dataform	either "wide" or "long" depending on which input format you wish to use for the data. See details and or vignette.
objective	'auto' selects either 'Kalman', if fitting to single subject data, or 'mxRAM' for multiple subjects. For single subject data, 'Kalman' uses the mxExpectationStateSpace function from OpenMx to implement the Kalman filter. For more than one subject, 'mxRAM' specifies a wide format SEM with a row of data per subject. 'cov' may be specified, in which case the 'meanIntervals' argument is set to TRUE, and the covariance matrix of the supplied data is calculated and fit instead of the raw data. This is much faster but only a rough approximation, unless there are no individual differences in time interval and no missing data. 'Kalman' may be specified for multiple subjects, however as no trait matrices are used by the Kalman filter one must consider how average level differences between subjects are accounted for. See ctMultigroupFit for the possibility to apply the Kalman filter over multiple subjects)
stationary	Character vector of T0 matrix names in which to constrain any free parameters to stationarity. Defaults to c('T0TRAITEFFECT','T0TIPREDEFFECT'), constraining only between person effects to stationarity. Use NULL for no constraints, or 'all' to constrain all T0 matrices.
optimizer	character string, defaults to the open-source 'CSOLNP' optimizer that is distributed in all versions of OpenMx.
retryattempts	Number of times to retry the start value randomisation and fit procedure, if non-convergance or uncertain fits occur.
iterationSummary	if TRUE, outputs limited fit details after every fit attempt.
carefulFit	if TRUE, first fits the specified model with a penalised likelihood function to force MANIFESTVAR, DRIFT, TRAITVAR, MANIFESTTRAITVAR parameters to remain close to 0, then fits the specified model normally, using these estimates as starting values. Can help to ensure optimization begins at sensible, non-exteme values, though results in any user specified start values being ignored for the final fit (though they are still used for initial fit).
carefulFitWeight	Positive numeric. Sets the weight for the penalisation (or prior) applied by the carefulFit algorithm. Generally unnecessary to adjust, may be helpful to try a selection of values (perhaps between 0 and 1000) when optimization is problematic.
showInits	if TRUE, prints the list of starting values for free parameters. These are the 'raw' values used by OpenMx, and reflect the log (var / cov matrices) or -log(DRIFT matrices) transformations used in ctsem. These are saved in the fit object under fitobject$omxStartValues.
asymptotes	when TRUE, optimizes over asymptotic parameter matrices instead of continuous time parameter matrices. Can be faster for optimization and in some cases

makes reliable convergance easier. Will result in equivalent models when continuous time input matrices (DRIFT, DIFFUSION, CINT) are free, but fixing the values of any such matrices will result in large differences - a value of 0 in a cell of the normal continuous time DIFFUSION matrix does not necessarily result in a value of 0 for the asymptotic DIFFUSION matrix, for instance.

`meanIntervals` Use average time intervals for each column for calculation (both faster and inaccurate to the extent that intervals vary across individuals).

`crossEffectNegStarts`
Logical. If TRUE (default) free DRIFT matrix cross effect parameters have starting values set to small negative values (e.g. -.05), if FALSE, the start values are 0. The TRUE setting is useful for easy initialisation of higher order models, while the FALSE setting is useful when one has already estimated a model without cross effects, and wishes to begin optimization from those values by using the omxStartValues switch. are re-transformed into regular continuous time parameter matrices, and may be interpreted as normal.

`fit` if FALSE, output only openmx model without fitting

`nofit` Deprecated. If TRUE, output only openmx model without fitting

`discreteTime` Estimate a discrete time model - ignores timing information, parameter estimates will correspond to those of classical vector autoregression models, OpenMx fit object will be directly output, thus ctsem summary and plot functionality will be unavailable. Time dependent predictor type also becomes irrelevant.

`verbose` Integer between 0 and 3. Sets mxComputeGradientDescent messaging level, defaults to 0.

`useOptimizer` Logical. Defaults to TRUE. Passes argument to `mxRun`, useful for using custom optimizers or fitting to specified parameters.

`omxStartValues` A named vector containing the raw (potentially log transformed) OpenMx starting values for free parameters, as captured by OpenMx function `omxGetParameters(ctmodelobj$mxobj`
These values will take precedence over any starting values already specified using ctModel.

`transformedParams`
Logical indicating whether or not to log transform certain parameters internally to allow unconstrained estimation over entire 'sensible' range for parameters. When TRUE (default) raw OpenMx parameters (only reported if `verbose=TRUE` argument used for summary function) will reflect these transformations and may be harder to interpret, but summary matrices are reported as normal.

`datawide` included for compatibility with scripts written for earlier versions of ctsem. Do not use this argument, instead use the dat argument, and the dataform argument now specifies whether the data is in wide or long format.

Details

For full discussion of how to structure the data and use this function, see the vignette using: `vignette('ctsem')`, or the data examples `data("longexample")` ; `longexample` for long and `data("datastructure")` ; `datastructure` for wide. If using long format, the subject id column must be numeric and grouped by ascending time within subject, and named 'id'. The time column must also be numeric, and representing absolute time (e.g., since beginning of study, *not* time

intervals), and called 'time'. Models are specified using the ctModel function. For help regarding the summary function, see summary.ctsemFit, and for the plot function, plot.ctsemFit. Multigroup models may be specified using ctMultigroupFit. Confidence intervals for any matrices and or parameters may be estimated using ctCI. Difficulties during estimation can sometimes be alleviated using ctRefineTo instead of ctFit – this uses a multistep fit procedure.

Examples

```
## Examples set to 'donttest' because they take longer than 5s.

mfrowOld<-par()$mfrow
par(mfrow=c(2, 3))

### example from Driver, Oud, Voelkle (2017),
### simulated happiness and leisure time with unobserved heterogeneity.
data(ctExample1)
traitmodel <- ctModel(n.manifest=2, n.latent=2, Tpoints=6, LAMBDA=diag(2),
  manifestNames=c('LeisureTime', 'Happiness'),
  latentNames=c('LeisureTime', 'Happiness'), TRAITVAR="auto")
traitfit <- ctFit(dat=ctExample1, ctmodelobj=traitmodel)
summary(traitfit)
plot(traitfit, wait=FALSE)


###Example from Voelkle, Oud, Davidov, and Schmidt (2012) - anomia and authoritarianism.
data(AnomAuth)
AnomAuthmodel <- ctModel(LAMBDA = matrix(c(1, 0, 0, 1), nrow = 2, ncol = 2),
Tpoints = 5, n.latent = 2, n.manifest = 2, MANIFESTVAR=diag(0, 2), TRAITVAR = NULL)
AnomAuthfit <- ctFit(AnomAuth, AnomAuthmodel)
summary(AnomAuthfit)


### Single subject time series - using Kalman filter (OpenMx statespace expectation)
data('ctExample3')
model <- ctModel(n.latent = 1, n.manifest = 3, Tpoints = 100,
  LAMBDA = matrix(c(1, 'lambda2', 'lambda3'), nrow = 3, ncol = 1),
  CINT= matrix('cint'),
  MANIFESTMEANS = matrix(c(0, 'manifestmean2', 'manifestmean3'), nrow = 3,
    ncol = 1))
fit <- ctFit(dat = ctExample3, ctmodelobj = model, objective = 'Kalman',
  stationary = c('T0VAR'))


###Oscillating model from Voelkle & Oud (2013).
data("Oscillating")

inits <- c(-39, -.3, 1.01, 10.01, .1, 10.01, 0.05, .9, 0)
names(inits) <- c("crosseffect","autoeffect", "diffusion",
  "T0var11", "T0var21", "T0var22","m1", "m2", 'manifestmean')

oscillatingm <- ctModel(n.latent = 2, n.manifest = 1, Tpoints = 11,
  MANIFESTVAR = matrix(c(0), nrow = 1, ncol = 1),
```

```
  LAMBDA = matrix(c(1, 0), nrow = 1, ncol = 2),
  T0MEANS = matrix(c('m1', 'm2'), nrow = 2, ncol = 1),
  T0VAR = matrix(c("T0var11", "T0var21", 0, "T0var22"), nrow = 2, ncol = 2),
  DRIFT = matrix(c(0, "crosseffect", 1, "autoeffect"), nrow = 2, ncol = 2),
  CINT = matrix(0, ncol = 1, nrow = 2),
  MANIFESTMEANS = matrix('manifestmean', nrow = 1, ncol = 1),
  DIFFUSION = matrix(c(0, 0, 0, "diffusion"), nrow = 2, ncol = 2),
  startValues=inits)

oscillatingf <- ctFit(Oscillating, oscillatingm, carefulFit = FALSE)
```

---

ctGenerate *ctGenerate*

Description

This function generates data according to the specified ctsem model object.

Usage

```
ctGenerate(
  ctmodelobj,
  n.subjects = 100,
  burnin = 0,
  dtmean = 1,
  logdtsd = 0,
  dtmat = NA,
  wide = FALSE
)
```

Arguments

ctmodelobj	ctsem model object from ctModel.
n.subjects	Number of subjects to output.
burnin	Number of initial time points to discard (to simulate stationary data)
dtmean	Positive numeric. Average time interval (delta T) to use.
logdtsd	Numeric. Standard deviation for variability of the time interval.
dtmat	Either NA, or numeric matrix of n.subjects rows and Tpoints-1 columns, containing positive numeric values for all time intervals between measurements. If not NA, dtmean and logdtsd are ignored.
wide	Logical. Output in wide format?

Details

Covariance related matrices are treated as Cholesky factors. TRAITTDPREDCOV and TIPREDCOV matrices are not accounted for, at present. The first 1:n.TDpred rows and columns of TDPREDVAR are used for generating tdpreds at each time point.

Examples

```
#generate data for 2 process model, each process measured by noisy indicator,
#stable individual differences in process levels.

generatingModel<-ctModel(Tpoints=8,n.latent=2,n.TDpred=0,n.TIpred=0,n.manifest=2,
 MANIFESTVAR=diag(.1,2),
 LAMBDA=diag(1,2),
 DRIFT=matrix(c(-.2,-.05,-.1,-.1),nrow=2),
 TRAITVAR=matrix(c(.5,.2,0,.8),nrow=2),
 DIFFUSION=matrix(c(1,.2,0,4),2),
 CINT=matrix(c(1,0),nrow=2),
 T0MEANS=matrix(0,ncol=1,nrow=2),
 T0VAR=diag(1,2))

data<-ctGenerate(generatingModel,n.subjects=15,burnin=10)
```

---

ctGenerateFromFit *Generates data according to the model estimated in a ctsemFit object.*

Description

Generates data according to the model estimated in a ctsemFit object.

Usage

```
ctGenerateFromFit(
  fit,
  timestep = "asdata",
  n.subjects = 100,
  timerange = "asdata",
  predictorSubjects = "all",
  ...
)
```

Arguments

`fit`	object of class ctsemFit as returned from ctFit.
`timestep`	positive numeric value indicating the time interval to use for data generation.
`n.subjects`	integer. Number of subjects worth of data to generate
`timerange`	either 'asdata' to calculate range based on data in fit object, or vector of length 2 specifying min and max times for generation.
`predictorSubjects`	vector of integers, or string 'all', defining which subjects to sample time dependent and independent predictors from.
`...`	parameters to pass to ctGenerate function, such as wide=FALSE.

Value

matrix of generated data

Examples

```
data(AnomAuth)
AnomAuthmodel <- ctModel(LAMBDA = matrix(c(1, 0, 0, 1), nrow = 2, ncol = 2),
  Tpoints = 5, n.latent = 2, n.manifest = 2, MANIFESTVAR=diag(0, 2))
AnomAuthfit <- ctFit(AnomAuth, AnomAuthmodel)

dwide <- ctGenerateFromFit(AnomAuthfit,timestep=1,n.subjects=5,wide=TRUE)
head(dwide)
```

---

ctIndplot *ctIndplot*

Description

Convenience function to simply plot individuals trajectories from ctsem wide format data

Usage

```
ctIndplot(
  datawide,
  n.manifest,
  Tpoints,
  n.subjects = "all",
  colourby = "variable",
  vars = "all",
  opacity = 1,
  varnames = NULL,
  xlab = "Time",
  ylab = "Value",
  type = "b",
  start = 0,
  legend = TRUE,
  legendposition = "topright",
  new = TRUE,
  jittersd = 0.05,
  ...
)
```

Arguments

datawide	ctsem wide format data
n.manifest	Number of manifest variables in data structure

Tpoints	Number of discrete time points per case in data structure
n.subjects	Number of subjects to randomly select for plotting, or character vector 'all'.
colourby	set plot colours by "subject" or "variable"
vars	either 'all' or a numeric vector specifying which manifest variables to plot.
opacity	Opacity of plot lines
varnames	vector of variable names for legend (defaults to NULL)
xlab	X axis label.
ylab	Y axis label.
type	character specifying plot type, as per usual base R plot commands. Defaults to 'b', both points and lines.
start	Measurement occasion to start plotting from - defaults to T0.
legend	Logical. Plot a legend?
legendposition	Where to position the legend.
new	logical. If TRUE, creates a new plot, otherwise overlays on current plot.
jittersd	positive numeric indicating standard deviation of noise to add to observed data for plotting purposes.
...	additional plotting parameters.

Examples

```
data(ctExample1)
ctIndplot(ctExample1,n.subjects=1, n.manifest=2,Tpoints=6, colourby='variable')
```

---

ctIntervalise *Converts absolute times to intervals for wide format ctsem panel data*

Description

Converts absolute times to intervals for wide format ctsem panel data

Usage

```
ctIntervalise(
  datawide,
  Tpoints,
  n.manifest,
  n.TDpred = 0,
  n.TIpred = 0,
  imputedefs = F,
  manifestNames = "auto",
  TDpredNames = "auto",
  TIpredNames = "auto",
```

```
  digits = 5,
  mininterval = 0.001,
  individualRelativeTime = TRUE,
  startoffset = 0
)
```

Arguments

datawide	Wide format data, containing absolute time measurements, to convert to interval time scale. See ctLongToWide to easily convert long format data.
Tpoints	Maximum number of discrete time points (waves of data, or measurement occasions) for an individual in the input data structure.
n.manifest	number of manifest variables per time point in the data.
n.TDpred	number of time dependent predictors in the data structure.
n.TIpred	number of time independent predictors in the data structure.
imputedefs	if TRUE, impute time intervals based on the measurement occasion (i.e. column) they are in, if FALSE (default), set related observations to NA. FALSE is recommended unless you are certain that the imputed value (mean of the relevant time column) is appropriate. Noise and bias in estimates will result if wrongly set to TRUE.
manifestNames	vector of character strings giving variable names of manifest indicator variables (without _Tx suffix for measurement occasion).
TDpredNames	vector of character strings giving variable names of time dependent predictor variables (without _Tx suffix for measurement occasion).
TIpredNames	vector of character strings giving variable names of time independent predictor variables.
digits	How many digits to round to for interval calculations.
mininterval	set to lower than any possible observed measurement interval, but above 0 - this is used for filling NA values where necessary and has no impact on estimates when set in the correct range. (If all observed intervals are greater than 1, mininterval=1 may be a good choice)
individualRelativeTime	if TRUE (default), the first measurement for each individual is assumed to be taken at time 0, and all other times are adjusted accordingly. If FALSE, new columns for an initial wave are created, consisting only of observations which occurred at the earliest observation time of the entire sample.
startoffset	if 0 (default) uses earliest observation as start time. If greater than 0, all first observations are NA, with distance of startoffset to first recorded observation.

Details

Time column must be numeric!

Examples

```
wideexample <- ctLongToWide(datalong = ctstantestdat, id = "id",
time = "time", manifestNames = c("Y1", "Y2"),
TDpredNames = "TD1", TIpredNames = c("TI1", "TI2","TI3"))

#Then convert the absolute times to intervals, using the Tpoints reported from the prior step.
wide <- ctIntervalise(datawide = wideexample, Tpoints = 10, n.manifest = 2,
n.TDpred = 1, n.TIpred = 3, manifestNames = c("Y1", "Y2"),
TDpredNames = "TD1", TIpredNames = c("TI1", "TI2","TI3") )

print(wide)
```

`ctLongToWide` *ctLongToWide Restructures time series / panel data from long format to wide format for ctsem analysis*

Description

ctLongToWide Restructures time series / panel data from long format to wide format for ctsem analysis

Usage

```
ctLongToWide(
  datalong,
  id,
  time,
  manifestNames,
  TDpredNames = NULL,
  TIpredNames = NULL
)
```

Arguments

`datalong`	dataset in long format, including subject/id column, observation time (or change in observation time, with 0 for first observation) column, indicator (manifest / observed) variables, any time dependent predictors, and any time independent predictors.
`id`	character string giving column name of the subject/id column
`time`	character string giving column name of the time columnn
`manifestNames`	vector of character strings giving column names of manifest indicator variables
`TDpredNames`	vector of character strings giving column names of time dependent predictor variables
`TIpredNames`	vector of character strings giving column names of time independent predictor variables

Details

Time column must be numeric

See Also

ctIntervalise

Examples

```
wideexample <- ctLongToWide(datalong = ctstantestdat, id = "id",
time = "time", manifestNames = c("Y1", "Y2"),
TDpredNames = "TD1", TIpredNames = c("TI1", "TI2","TI3"))

#Then convert the absolute times to intervals, using the Tpoints reported from the prior step.
wide <- ctIntervalise(datawide = wideexample, Tpoints = 10, n.manifest = 2,
n.TDpred = 1, n.TIpred = 3, manifestNames = c("Y1", "Y2"),
TDpredNames = "TD1", TIpredNames = c("TI1", "TI2","TI3") )

print(wide)
```

---

ctModel *Define a ctsem model*

Description

This function is used to specify a continuous time structural equation model, which can then be fit to data with function ctFit.

Usage

```
ctModel(
  LAMBDA,
  n.manifest = "auto",
  n.latent = "auto",
  Tpoints = NULL,
  manifestNames = "auto",
  manifesttype = rep(0, nrow(LAMBDA)),
  latentNames = "auto",
  id = "id",
  time = "time",
  silent = FALSE,
  T0VAR = "auto",
  T0MEANS = "auto",
  MANIFESTMEANS = "auto",
  MANIFESTVAR = "diag",
  DRIFT = "auto",
  CINT = 0,
  DIFFUSION = "auto",
```

```
  n.TDpred = "auto",
  TDpredNames = "auto",
  n.TIpred = "auto",
  TIpredNames = "auto",
  TRAITVAR = NULL,
  T0TRAITEFFECT = NULL,
  MANIFESTTRAITVAR = NULL,
  TDPREDMEANS = "auto",
  TDPREDEFFECT = "auto",
  T0TDPREDCOV = "auto",
  TDPREDVAR = "auto",
  TRAITTDPREDCOV = "auto",
  TDTIPREDCOV = "auto",
  TIPREDMEANS = "auto",
  TIPREDEFFECT = "auto",
  T0TIPREDEFFECT = "auto",
  TIPREDVAR = "auto",
  startValues = NULL
)
```

Arguments

LAMBDA	n.manifest*n.latent loading matrix relating latent to manifest variables, with latent processes 1:n.latent along the columns, and manifest variables 1:n.manifest in the rows.
n.manifest	Number of manifest indicators per individual at each measurement occasion / time point. Manifest variables are included as the first element of the wide data matrix, with all the 1:n.manifest manifest variables at time 1 followed by those of time 2, and so on.
n.latent	Number of latent processes.
Tpoints	Number of time points, or measurement occasions, in the data. This will generally be the maximum number of time points for a single individual, but may be one extra if sample relative time intervals are used, see ctIntervalise.
manifestNames	n.manifest length vector of manifest variable names as they appear in the data structure, without any _Tx time point suffix that may be present in wide data. Defaults to Y1, Y2, etc.
manifesttype	n.manifest length vector of manifest variable types,defaults to 0 for continuous vars, 1 for binary vars is also possible.
latentNames	n.latent length vector of latent variable names (used for naming parameters, defaults to eta1, eta2, etc).
id	character string denoting column name containing subject identification variables. id data may be of any form, though will be coerced internally to an integer sequence rising from 1.
time	character string denoting column name containing timing data. Timing data must be numeric.
silent	Suppress all output to console.

T0VAR lower triangular n.latent*n.latent cholesky matrix of latent process initial variance / covariance. "auto" freely estimates all parameters.

T0MEANS n.latent*1 matrix of latent process means at first time point, T0. "auto" freely estimates all parameters.

MANIFESTMEANS n.manifest*1 matrix of manifest intercept parameters. "auto" frees all parameters.

MANIFESTVAR lower triangular n.manifest*n.manifest cholesky matrix of variance / covariance between manifests at each measurement occasion (i.e. measurement error / residual). "auto" freely estimates variance parameters, and fixes covariances between manifests to 0. "free" frees all values, including covariances.

DRIFT n.latent*n.latent DRIFT matrix of continuous auto and cross effects, relating the processes over time. "auto" freely estimates all parameters.

CINT n.latent * 1 matrix of latent process intercepts, allowing for non 0 asymptotic levels of the latent processes. Generally only necessary for additional trends and more complex dynamics.

DIFFUSION lower triangular n.latent*n.latent cholesky matrix of diffusion process variance and covariance (latent error / dynamic innovation). "auto" freely estimates all parameters.

n.TDpred Number of time dependent predictor variables in the dataset.

TDpredNames n.TDpred length vector of time dependent predictor variable names, as they appear in the data structure, without any _Tx time point suffix that may appear in wide data. Default names are TD1, TD2, etc.

n.TIpred Number of time independent predictors. Each TIpredictor is inserted at the right of the data matrix, after the time intervals.

TIpredNames n.TIpred length vector of time independent predictor variable names, as they appear in the data structure. Default names are TI1, TI2, etc.

TRAITVAR Either NULL, if no trait / unobserved heterogeneity effect, or lower triangular n.latent*n.latent cholesky matrix of trait variance / covariance across subjects. "auto" freely estimates all parameters.

T0TRAITEFFECT Either NULL, if no trait / individual heterogeneity effect, or lower triangular n.latent*n.latent cholesky matrix of initial trait variance / covariance. "auto" freely estimates all parametrers, if the TRAITVAR matrix is specified.

MANIFESTTRAITVAR
Either NULL (default) if no trait variance / individual heterogeneity in the level of the manifest indicators, otherwise a lower triangular n.manifest * n.manifest variance / covariance matrix. Set to "auto" to include and free all parameters - but identification problems will arise if TRAITVAR is also set.

TDPREDMEANS (n.TDpred * (Tpoints - 1)) rows * 1 column matrix of time dependent predictor means. If 'auto', the means are freely estimated. Otherwise, the means for the Tpoints observations of your first time dependent predictor are followed by those of TDpred 2, and so on.

TDPREDEFFECT n.latent*n.TDpred matrix of effects from time dependent predictors to latent processes. Effects from 1:n.TDpred columns TDpredictors go to 1:n.latent rows of latent processes. "auto" freely estimates all parameters.

T0TDPREDCOV — n.latent rows * (Tpoints * n.TDpred) columns covariance matrix between latents at T0 and time dependent predictors. Default of "auto" restricts covariance to 0, which is consistent with covariance to other time points. To freely estimate parameters, specify either 'free', or the desired matrix.

TDPREDVAR — lower triangular (n.TDpred * Tpoints) rows * (n.TDpred * Tpoints) columns variance / covariance cholesky matrix for time dependent predictors. "auto" (default) freely estimates all parameters.

TRAITTDPREDCOV — n.latent rows * (n.TDpred*Tpoints) columns covariance matrix of latent traits and time dependent predictors. Defaults to zeroes, assuming predictors are independent of subjects baseline levels. When predictors depend on the subjects, this should instead be set to 'free' or manually specified. The Tpoints columns of the first preditor are followed by those of the second and so on. Covariances with the trait variance of latent process 1 are specified in row 1, process 2 in row 2, etc. "auto" (default) sets this matrix to zeroes, (if both traits and time dependent predictors exist, otherwise this matrix is set to NULL, and ignored in any case).

TDTIPREDCOV — (n.TDpred * Tpoints) rows * n.TIpred columns covariance matrix between time dependent and time independent predictors. "auto" (default) freely estimates all parameters.

TIPREDMEANS — n.TIpred * 1 matrix of time independent predictor means. If 'auto', the means are freely estimated.

TIPREDEFFECT — n.latent*n.TIpred effect matrix of time independent predictors on latent processes. "auto" freely estimates all parameters and generates starting values.

T0TIPREDEFFECT — n.latent*n.TIpred effect matrix of time independent predictors on latents at T0. "auto" freely estimates all parameters, though note that under the default setting of `stationary` for ctFit, this matrix is ignored as the effects are determined based on the overall process parameters.

TIPREDVAR — lower triangular n.TIpred * n.TIpred Cholesky decomposed covariance matrix for all time independent predictors. "auto" (default) freely estimates all parameters.

startValues — A named vector, where the names of each value must match a parameter in the specified model, and the value sets the starting value for that parameter during optimization. If not set, random starting values representing relatively stable processes with small effects and covariances are generated by ctFit. Better starting values may improve model fit speed and the chance of an appropriate model fit.

Examples

```
### Frequentist example:
### impulse and level change time dependent predictor
### example from Driver, Oud, Voelkle (2015)
data('ctExample2')
tdpredmodel <- ctModel(n.manifest = 2, n.latent = 3, n.TDpred = 1,
Tpoints = 8, manifestNames = c('LeisureTime', 'Happiness'),
TDpredNames = 'MoneyInt',
latentNames = c('LeisureTime', 'Happiness', 'MoneyIntLatent'),
```

```
LAMBDA = matrix(c(1,0, 0,1, 0,0), ncol = 3), TRAITVAR = "auto")

tdpredmodel$TRAITVAR[3, ] <- 0
tdpredmodel$TRAITVAR[, 3] <- 0
tdpredmodel$DIFFUSION[, 3] <- 0
tdpredmodel$DIFFUSION[3, ] <- 0
tdpredmodel$T0VAR[3, ] <- 0
tdpredmodel$T0VAR[, 3] <- 0
tdpredmodel$CINT[3] <- 0
tdpredmodel$T0MEANS[3] <- 0
tdpredmodel$TDPREDEFFECT[3, ] <- 1
tdpredmodel$DRIFT[3, ] <- 0
```

`ctModelFromFit` *Extract a ctsem model structure with parameter values from a ctsem fit object.*

Description

Extract a ctsem model structure with parameter values from a ctsem fit object.

Usage

```
ctModelFromFit(fit)
```

Arguments

`fit`	object output by `ctFit`

Value

object of class 'ctsemInit' (as generated by `ctModel`), which can be used with `ctFit` and functions.

Examples

```
data(AnomAuth)
AnomAuthmodel <- ctModel(LAMBDA = matrix(c(1, 0, 0, 1), nrow = 2, ncol = 2),
  Tpoints = 5, n.latent = 2, n.manifest = 2, MANIFESTVAR=diag(0, 2))
AnomAuthfit <- ctFit(AnomAuth, AnomAuthmodel)

fitmodel <- ctModelFromFit(AnomAuthfit)
```

`ctMultigroupFit` *Fits a multiple group continuous time model.*

Description

Fits a single continuous time structural equation models to multiple groups (where each group contains 1 or more subjects), by default, all parameters are free across groups. Can also be used to easily estimate seperate models for each group.

Usage

```
ctMultigroupFit(
  dat,
  groupings,
  ctmodelobj,
  dataform = "wide",
  fixedmodel = NA,
  freemodel = NA,
  carefulFit = TRUE,
  omxStartValues = NULL,
  retryattempts = 5,
  showInits = FALSE,
  ...
)
```

Arguments

`dat`	Wide format data, as used in `ctFit`. See `ctLongToWide` to easily convert long format data.
`groupings`	For wide format: Vector of character labels designating group membership for each row of dat. For long format: Named list of groups, with each list element containing a vector of subject id's for the group. In both cases, group names will be prefixed on relevant parameter estimates in the summary.
`ctmodelobj`	Continuous time model to fit, specified via `ctModel` function.
`dataform`	either "wide" or "long" depending on which input format you wish to use for the data. See details of `ctFit` and or vignette.
`fixedmodel`	Modified version of ctmodelobj, wherein any parameters you wish to keep fixed over groups should be given the value 'groupfixed'. If specified, all other parameters will be free across groups.
`freemodel`	Modified version of ctmodelobj, wherein any parameters you wish to free across groups should be given the label 'groupfree'. If specified, all other parameters will be fixed across groups. If left NULL, the default, all parameters are free across groups.

carefulFit	if TRUE, first fits the specified model with a penalised likelihood function to discourage parameters from boundary conditions, then fits the specified model normally, using these estimates as starting values. Can help / speed optimization, though results in user specified inits being ignored for the final fit.
omxStartValues	A named vector containing the raw (potentially log transformed) OpenMx starting values for free parameters, as captured by OpenMx function omxGetParameters(ctmodelobj$mxobj These values will take precedence over any starting values already specified using ctModel.
retryattempts	Number of fit retries to make.
showInits	Displays start values prior to optimization
...	additional arguments to pass to ctFit.

Details

Additional ctFit parameters may be specified as required. Confidence intervals for any matrices and or parameters may be estimated afer fitting using ctCI.

Value

Returns an OpenMx fit object.

See Also

ctFit and ctModel

Examples

```
#Two group model, all parameters except LAMBDA[3,1] constrained across groups.
data(ctExample4)
basemodel<-ctModel(n.latent=1, n.manifest=3, Tpoints=20,
                   LAMBDA=matrix(c(1, 'lambda2', 'lambda3'), nrow=3, ncol=1),
                   MANIFESTMEANS=matrix(c(0, 'manifestmean2', 'manifestmean3'),
                   nrow=3, ncol=1), TRAITVAR = 'auto')

freemodel<-basemodel
freemodel$LAMBDA[3,1]<-'groupfree'
groups<-paste0('g',rep(1:2, each=10),'_')

multif<-ctMultigroupFit(dat=ctExample4, groupings=groups,
                        ctmodelobj=basemodel, freemodel=freemodel)
summary(multif,group=1)



#fixed model approach
fixedmodel<-basemodel
fixedmodel$LAMBDA[2,1]<-'groupfixed'
groups<-paste0('g',rep(1:2, each=10),'_')
```

```
multif<-ctMultigroupFit(dat=ctExample4, groupings=groups,
                        ctmodelobj=basemodel, fixedmodel=fixedmodel)
summary(multif,group=2)
```

---

ctPlot *ctPlot*

Description

Plots mean trajectories, autoregression, and crossregression plots, for ctsemFit objects. More customizeable than basic plot.ctsemFit function.

Usage

```
ctPlot(
  x,
  plotType,
  xlim,
  resolution = 50,
  impulseIndex = NULL,
  subject = 1,
  typeVector = "auto",
  colVector = "auto",
  ltyVector = "auto",
  ...
)
```

Arguments

x	ctsemFit object as generated by ctFit.
plotType	string. "mean" for expectation independent of any data, "AR" for autoregressions, "CR" for cross regressions, "standardiseCR" for standardised cross regressions (standardised based on estimated within subject variance), "withinVar" for within variance and covariance, "randomImpulse" for expected change in processes given a random fluctuation of +1 for each process (so a mixture of DIFFUSION and DRIFT characteristics), "experimentalImpulse" for expected change in processes given an exogenous input of +1 for each process, provides alternate characterisation of autoregressive and cross regressive plots.
xlim	vector. As per usual for plot(), but xlim may not be negative.
resolution	Numeric. Plot points between each unit of time. Default of 'auto' adapts to xlim and results in 500 points in total.
impulseIndex	Numeric. Only required for impulse plot types, specifies which column of the DRIFT matrix the impulse relates to.

subject	numeric. Specifies the subject (row of data from the mxobj) to plot for factorScores type plot.
typeVector	Vector of plot types to use for plotting.
colVector	vector of colours to use for plotting.
ltyVector	Vector of line types to use for plotting.
...	Other options passed to plot(). ylim is required.

Value

Character vector of labels from the DRIFT matrix in order plotted - useful for legends. Side-effect: plots graphs.

Examples

```
## Examples set to 'donttest' because they take longer than 5s.

### example from Driver, Oud, Voelkle (2016),
### simulated happiness and leisure time with unobserved heterogeneity.

data(ctExample1)
traitmodel <- ctModel(n.manifest=2, n.latent=2, Tpoints=6, LAMBDA=diag(2),
  manifestNames=c('LeisureTime', 'Happiness'),
  latentNames=c('LeisureTime', 'Happiness'), TRAITVAR="auto")
traitfit <- ctFit(dat=ctExample1, ctmodelobj=traitmodel)
ctPlot(traitfit, plotType='CR', xlim=c(0,5),ylim=c(-1,1))
```

---

ctRefineTo *ctRefineTo*

Description

Fits a ctsem m in a stepwise fashion to help with difficult optimization.

Usage

```
ctRefineTo(datawide, ctmodelobj, modfunc = NULL, ...)
```

Arguments

datawide	Data in ctsem wide format
ctmodelobj	A continuous time m specified via the ctModel function.
modfunc	function to run prior to each optimization step, that takes ctsem fit object, modifies it as desired, and returns the fit object.
...	additional parameters to pass to ctFit.

Details

This function fits a sequence of ctsem models increasing in complexity, starting with a m involving fixed and relatively strong auto effects, no cross effects, no predictors, and no off-diagonal covariances. For many models this can improve the speed and robustness of fitting

Value

Returns a fitted ctsem object in the same manner as `ctFit`.

---

ctstantestdat *ctstantestdat*

Description

Generated dataset for testing.

Format

matrix

---

ctWideToLong *ctWideToLong Convert ctsem wide to long format*

Description

ctWideToLong Convert ctsem wide to long format

Usage

```
ctWideToLong(
  datawide,
  Tpoints,
  n.manifest,
  n.TDpred = 0,
  n.TIpred = 0,
  manifestNames = "auto",
  TDpredNames = "auto",
  TIpredNames = "auto"
)
```

Arguments

datawide	ctsem wide format data
Tpoints	number of measurement occasions in data
n.manifest	number of manifest variables
n.TDpred	number of time dependent predictors
n.TIpred	number of time independent predictors
manifestNames	Character vector of manifest variable names.
TDpredNames	Character vector of time dependent predictor names.
TIpredNames	Character vector of time independent predictor names.

Details

Names must account for *all* the columns in the data - i.e. do not leave certain variables out just because you do not need them.

Examples

```
#create wide data
wideexample <- ctLongToWide(datalong = ctstantestdat, id = "id",
time = "time", manifestNames = c("Y1", "Y2"),
TDpredNames = "TD1", TIpredNames = c("TI1", "TI2","TI3"))

wide <- ctIntervalise(datawide = wideexample, Tpoints = 10, n.manifest = 2,
n.TDpred = 1, n.TIpred = 3, manifestNames = c("Y1", "Y2"),
TDpredNames = "TD1", TIpredNames = c("TI1", "TI2","TI3") )

 #Then convert to long format
 longexample <- ctWideToLong(datawide = wideexample, Tpoints=10,
 n.manifest=2, manifestNames = c("Y1", "Y2"),
 n.TDpred=1, TDpredNames = "TD1",
 n.TIpred=3, TIpredNames = c("TI1", "TI2","TI3"))

 #Then convert the time intervals to absolute time
 long <- ctDeintervalise(datalong = longexample, id='id', dT='dT')
 head(long,22)
```

---

datastructure *datastructure*

Description

Simulated example dataset for the ctsem package

Format

2 by 15 matrix containing containing ctsem wide format data. 3 measurement occasions of manifest variables Y1 and Y2, 2 measurement occasions of time dependent predictor TD1, 2 measurement intervals dTx, and 2 time independent predictors TI1 and TI2, for 2 individuals.

`longexample` *longexample*

Description

Simulated example dataset for the ctsem package

Format

7 by 8 matrix containing ctsem long format data, for two subjects, with three manifest variables Y1, Y2, Y3, one time dependent predictor TD1, two time independent predictors TI1 and TI2, and absolute timing information Time.

`Oscillating` *Oscillating*

Description

Simulated example dataset for the ctsem package.

Format

200 by 21 matrix containing containing ctsem wide format data. 11 measurement occasions and 10 measurement intervals for each of 200 individuals

Source

See https://bpspsychub.onlinelibrary.wiley.com/doi/abs/10.1111/j.2044-8317.2012.02043.x

`plot.ctsemFit` *Plotting function for object class ctsemFit*

Description

Ouputs mean trajectories, autoregression, and crossregression plots. For more customization possibilities, see `ctPlot`.

Usage

```
## S3 method for class 'ctsemFit'
plot(
  x,
  resolution = 50,
  wait = TRUE,
  max.time = "auto",
  mean = TRUE,
  withinVariance = TRUE,
  AR = TRUE,
  CR = TRUE,
  standardiseCR = FALSE,
  randomImpulse = FALSE,
  experimentalImpulse = FALSE,
  xlab = "Time",
  meansylim = "auto",
  ARylim = "auto",
  CRylim = "auto",
  ylab = "Value",
  ...
)
```

Arguments

`x`	ctsemFit object as generated by `ctFit`.
`resolution`	Numeric. Plot points between each unit of time. Default of 'auto' adapts to max.time and results in 500 in total.
`wait`	If true, user is prompted to continue before plotting next graph. If false, graphs are plotted one after another without waiting.
`max.time`	Time scale on which to plot parameters. If auto, parameters are plotted for full range of observed variables.
`mean`	if TRUE, plot of means from 0 to max.time included in output.
`withinVariance`	if TRUE, plot within subject variance / covariance.
`AR`	if TRUE, plot of autoregressive values from 0 to max.time included in output.
`CR`	if TRUE, plot of cross regressive values from 0 to max.time included in output.

standardiseCR	if TRUE , cross regression values are standardised based on estimated within subject variance.
randomImpulse	if TRUE (default), plots expected change in processes given a random fluctuation of +1 for each process – plot is then a mixture of DIFFUSION and DRIFT characteristics.
experimentalImpulse	if TRUE (default), plots expected change in processes given an exogenous input of +1 for each process – alternate characterisation of autoregressive and cross regressive plots.
xlab	X axis label.
meansylim	Vector of min and max limits for mean trajectory plot. 'auto' calculates automatically.
ARylim	Vector of min and max limits for autoregression plot. 'auto' is c(0,1), and expands if necessary.
CRylim	Vector of min and max limits for cross regression plot. 'auto' is c(-1,1), and expands if necessary.
ylab	Y axis label.
...	Other options passed to plot().

Value

Nothing. Side-effect: plots graphs.

Examples

```
## Examples set to 'donttest' because they take longer than 5s.

### example from Driver, Oud, Voelkle (2015),
### simulated happiness and leisure time with unobserved heterogeneity.

data(ctExample1)
traitmodel <- ctModel(n.manifest=2, n.latent=2, Tpoints=6, LAMBDA=diag(2),
  manifestNames=c('LeisureTime', 'Happiness'),
  latentNames=c('LeisureTime', 'Happiness'), TRAITVAR="auto")
traitfit <- ctFit(dat=ctExample1, ctmodelobj=traitmodel)
plot(traitfit, wait=FALSE)
```

---

plot.ctsemMultigroupFit

Plot function for ctsemMultigroupFit object

Description

Plots ctMultigroupFit objects.

Usage

```
## S3 method for class 'ctsemMultigroupFit'
plot(x, group = "show chooser", ...)
```

Arguments

x	ctsemMultigroupFit object as generated by ctMultigroupFit
group	character string of subgroup to plot. Default of 'show chooser' displays list and lets you select.
...	additional parameters to pass to plot.ctsemFit function.

Value

Nothing. Side-effect: plots graphs.

---

summary.ctsemFit *Summary function for ctsemFit object*

Description

Provides summary details for ctsemFit objects.

Usage

```
## S3 method for class 'ctsemFit'
summary(object, ridging = FALSE, timeInterval = 1, verbose = FALSE, ...)
```

Arguments

object	ctsemFit object as generated by ctFit.
ridging	if TRUE, adds a small amount of variance to diagonals when calculating standardised (correlation) matrices, should only be used if standardised matrices return NAN.
timeInterval	positive numeric value specifying time interval to use for discrete parameter matrices, defaults to 1.
verbose	Logical. If TRUE, displays the raw, internally transformed (when fitting with default arguments) OpenMx parameters and corresponding standard errors, as well as additional summary matrices. Parameter transforms are described in the vignette, vignette('ctsem'). Additional summary matrices include: 'discrete' matrices – matrices representing the effect for the given time interval (default of 1); 'asymptotic' matrices – represents the effect as time interval approaches infinity (therefore asymCINT describes mean level of processes at the asymptote, asymDIFFUSION describes total within- subject variance at the asymptote, etc); 'standardised' matrices – transforms covariance matrices to correlation matrices, and transforms discreteDRIFT based on DIFFUSION, to give effect sizes.
...	additional parameters to pass.

Details

Important: Although ctModel takes cholesky decomposed variance-covariance matrices as input, the summary function displays the full variance-covariance matrices. These can be cholesky decomposed for comparison purposes using `t(chol(summary(ctfitobject)$covariancematrix))`. Standard errors are displayed in the $ctparameters section, however if ctFit was used with transformedParams=TRUE (the default, and recommended) covariance matrix standard errors will have been approximated using the delta method. For inferential purposes, maximum likelihood confidence intervals may be estimated using the ctCI function.

Value

Summary of ctsemFit object

Examples

```
## Examples set to 'donttest' because they take longer than 5s.


### example from Driver, Oud, Voelkle (2015),
### simulated happiness and leisure time with unobserved heterogeneity.
data(ctExample1)
traitmodel <- ctModel(n.manifest=2, n.latent=2, Tpoints=6, LAMBDA=diag(2),
  manifestNames=c('LeisureTime', 'Happiness'),
  latentNames=c('LeisureTime', 'Happiness'), TRAITVAR="auto")
traitfit <- ctFit(dat=ctExample1, ctmodelobj=traitmodel)
summary(traitfit,timeInterval=1)
```

---

summary.ctsemMultigroupFit

Summary function for ctsemMultigroupFit object

Description

Provides summary details for objects fitted with ctMultigroupFit.

Usage

```
## S3 method for class 'ctsemMultigroupFit'
summary(object, group = "show chooser", ...)
```

Arguments

object	ctsemMultigroupFit object as generated by ctMultigroupFit
group	character string of subgroup to display summary parameters for. Default of 'show chooser' displays list and lets you select.
...	additional parameters to pass to summary.ctsemFit.

Value

Summary of ctsemMultigroupFit object

Index

AnomAuth, 2

ctCI, 3, *11*, *24*, *33*
ctCompareExpected, 4
ctDeintervalise, 5
ctDiscretiseData, 5
ctExample1, 6
ctExample1TIpred, 6
ctExample2, 7
ctExample2level, 7
ctExample3, 7
ctExample4, 8
ctFit, 8, *11*, *18*, *22–27*, *30*, *33*
ctGenerate, 12
ctGenerateFromFit, 13
ctIndplot, 14
ctIntervalise, 15, *18, 19*
ctLongToWide, *16*, 17, *23*
ctModel, *8, 9*, *11, 12*, 18, *22–24*, *26*, *33*
ctModelFromFit, 22
ctMultigroupFit, *9*, *11*, 23, *31–33*
ctPlot, 25, *30*
ctRefineTo, *11*, 26
ctstantestdat, 27
ctWideToLong, 27

datastructure, 28

longexample, 29

Oscillating, 29

plot.ctsemFit, *11*, 30, *32*
plot.ctsemMultigroupFit, 31

summary.ctsemFit, *11*, 32, *33*
summary.ctsemMultigroupFit, 33