Package ‘latrend’

July 4, 2022

Type Package

Version 1.4.1

Date 2022-07-04

Title A Framework for Clustering Longitudinal Data

Description A framework for clustering longitudinal datasets in a standardized way.

The package provides an interface to existing R packages for clustering longitudinal univariate trajectories, facilitating reproducible and transparent analyses.

Additionally, standard tools are provided to support cluster analyses, including repeated estimation, model validation, and model assessment.

The interface enables users to compare results between methods, and to implement and evaluate new methods with ease.

Maintainer Niek Den Teuling <niek.den.teuling@philips.com>

URL https://github.com/philips-software/latrend

BugReports https://github.com/philips-software/latrend/issues

License GPL (>= 2)

Encoding UTF-8

LazyData true

Language en-US

Depends R (>= 3.6.0)

Imports stats, methods, Rdpack, R.utils, assertthat (>= 0.2.1),
foreach, data.table (>= 1.12.0), magrittr, matrixStats, rlang,
markdown (>= 1.18)

Suggests testthat (>= 3.0.0), roxygen2 (>= 7.1.0), knitr (>= 1.24),
rcmdcheck, pkgdown, devtools, evaluate, lme4, covr, lintr,
tinytex, longitudinalData (>= 2.4.1), km1 (>= 2.4.1), lclmm (>= 1.9.3),
mixtools, flexmix, lda, funFEM, gridExtra, igraph,
crimCV, akmedoids (>= 1.3.0), dtwclust, mixAK, clusterCrit,
mclust, mclustcomp, psych, qplot, doParallel, simTool, dplyr,
geplot2, caret, tibble

RoxygenNote 7.2.0
R topics documented:

Rd Macros
Rdpack
Vignette Builder
knitr

Collate
`methodAKMedoids.R` `methodCrimCV.R` `methodDtwclust.R`
`methodFeature.R` `methodFlexmix.R` `methodFlexmixGBTM.R`
`methodFunFEM.R` `methodFunction.R` `methodLMKM.R`
`methodGCKM.R` `methodKML.R` `methodLcmmGMM.R`
`methodLcmmGBTM.R` `methodMclustLLPA.R` `methodMixAK_GLMM.R`
`methodMixTVEM.R` `methodMixtoolsGMM.R` `methodMixtoolsNPRM.R`
`modelDtwclust.R` `modelFlexmix.R` `modelFunFEM.R` `modelKML.R`
`modelLMKM.R` `modelLcmmGMM.R` `modelLcmmGBTM.R`
`modelMclustLLPA.R` `modelMixAK_GLMM.R` `modelMixAK_GLMMList.R`
`modelMixTVEM.R` `modelMixtoolsGMM.R` `modelMixtoolsRM.R`

Needs Compilation no

Author
Niek Den Teuling [aut, cre] (<https://orcid.org/0000-0003-1026-5080>),
Steffen Pauws [ctb],
Edwin van den Heuvel [ctb],
Copyright © 2022 Koninklijke Philips N.V. [cph]

Repository
CRAN

Date/Publication 2022-07-04 11:50:02 UTC

R topics documented:

latrend-package ........................................... 5
APPA .................................................................. 7
as.data.frame.lcMethod ..................................... 8
as.data.frame.lcMethods .................................. 9
as.data.frame.lcModels ................................... 9
as.lcMethods ................................................ 10
as.lcModels .................................................. 11
as.list.lcMethod .......................................... 11
clusterNames ............................................... 12
clusterNames<- ............................................ 13
clusterProportions ....................................... 14
clusterSizes ................................................ 15
clusterTrajectories ...................................... 16
coeff.lcModel ............................................. 17
compose ..................................................... 18
confusionMatrix ......................................... 19
R topics documented:

converged ......................................................... 20
createTestDataFold .............................................. 21
createTestDataFolds ............................................ 22
createTrainDataFolds .......................................... 22
defineExternalMetric .......................................... 23
defineInternalMetric .......................................... 24
deviance.lcModel ................................................. 25
df.residual.lcModel ............................................. 25
estimationTime ................................................... 26
evaluate.lcMethod ................................................. 27
externalMetric.lcModel,lcModel-method ...................... 28
fit ................................................................. 31
fitted.lcModel .................................................... 32
fittedTrajectories ............................................... 34
formula.lcMethod ................................................. 35
formula.lcModel .................................................. 36
generateLongData ................................................. 36
getArgumentDefaults ............................................ 38
getArgumentExclusions ......................................... 39
getExternalMetricDefinition ................................... 40
getExternalMetricNames ....................................... 40
gInternalsMetricDefinition .................................... 41
gInternalsMetricNames ......................................... 41
getId ............................................................. 42
gInternalsMethod ............................................... 42
getName .......................................................... 43
ids ................................................................. 44
idVariable ........................................................ 45
initialize.lcMethod-method .................................... 46
latrend ............................................................. 46
latrend-parallel .................................................. 47
latrendBatch ...................................................... 49
latrendBoot ....................................................... 50
latrendCV .......................................................... 51
latrendData ........................................................ 53
latrendRep ........................................................ 54
lcApproxModel-class ............................................. 55
lcMethod-class .................................................... 56
lcMethodAkmedoids .............................................. 58
lcMethodCrimCV .................................................. 59
lcMethodDtwclust ............................................... 60
lcMethodFeature .................................................. 61
lcMethodFlexmix .................................................. 63
lcMethodFlexmixGBTM .......................................... 64
lcMethodFunction ................................................. 65
lcMethodFunFEM ................................................... 66
lcMethodGCKM ..................................................... 67
lcMethodKML ....................................................... 69
<table>
<thead>
<tr>
<th>R topics documented:</th>
</tr>
</thead>
<tbody>
<tr>
<td>lcMethodLcmmGBTM</td>
</tr>
<tr>
<td>lcMethodLcmmGMM</td>
</tr>
<tr>
<td>lcMethodLMKM</td>
</tr>
<tr>
<td>lcMethodMclustLPA</td>
</tr>
<tr>
<td>lcMethodMixAK_GLMM</td>
</tr>
<tr>
<td>lcMethodMixtoolsGMM</td>
</tr>
<tr>
<td>lcMethodMixtoolsNPRM</td>
</tr>
<tr>
<td>lcMethodMixTVEM</td>
</tr>
<tr>
<td>lcMethodRandom</td>
</tr>
<tr>
<td>lcMethods</td>
</tr>
<tr>
<td>lcMethodStratify</td>
</tr>
<tr>
<td>lcModel-class</td>
</tr>
<tr>
<td>lcModelPartition</td>
</tr>
<tr>
<td>lcModels</td>
</tr>
<tr>
<td>lcModelWeightedPartition</td>
</tr>
<tr>
<td>logLik.lcModel</td>
</tr>
<tr>
<td>max.lcModels</td>
</tr>
<tr>
<td>metric</td>
</tr>
<tr>
<td>min.lcModels</td>
</tr>
<tr>
<td>model.data.lcModel</td>
</tr>
<tr>
<td>model.frame.lcModel</td>
</tr>
<tr>
<td>names.lcMethod-method</td>
</tr>
<tr>
<td>nClusters</td>
</tr>
<tr>
<td>nIds</td>
</tr>
<tr>
<td>nobs.lcModel</td>
</tr>
<tr>
<td>OCC</td>
</tr>
<tr>
<td>OSA.adherence</td>
</tr>
<tr>
<td>PAP.adh</td>
</tr>
<tr>
<td>plot-lcModel-method</td>
</tr>
<tr>
<td>plot-lcModels-method</td>
</tr>
<tr>
<td>plotClusterTrajectories</td>
</tr>
<tr>
<td>plotFittedTrajectories</td>
</tr>
<tr>
<td>plotMetric</td>
</tr>
<tr>
<td>plotTrajectories</td>
</tr>
<tr>
<td>postFit</td>
</tr>
<tr>
<td>postprob</td>
</tr>
<tr>
<td>postprobFromAssignments</td>
</tr>
<tr>
<td>predict.lcModel</td>
</tr>
<tr>
<td>predictAssignments</td>
</tr>
<tr>
<td>predictForCluster</td>
</tr>
<tr>
<td>predictPostprob</td>
</tr>
<tr>
<td>preFit</td>
</tr>
<tr>
<td>prepareData</td>
</tr>
<tr>
<td>print.lcMethod</td>
</tr>
<tr>
<td>print.lcModels</td>
</tr>
<tr>
<td>qqPlot</td>
</tr>
<tr>
<td>residuals.lcModel</td>
</tr>
<tr>
<td>responseVariable</td>
</tr>
</tbody>
</table>
Description

A framework for clustering longitudinal datasets in a standardized way. The package provides an interface to existing R packages for clustering longitudinal univariate trajectories, facilitating reproducible and transparent analyses. Additionally, standard tools are provided to support cluster analyses, including repeated estimation, model validation, and model assessment. The interface enables users to compare results between methods, and to implement and evaluate new methods with ease.

Features

- Unified cluster analysis, independent of the underlying algorithms used. Enabling users to compare the performance of various longitudinal cluster methods on the case study at hand.
- Supports many different methods for longitudinal clustering out of the box (see the list of supported packages below).
- The framework consists of extensible S4 methods based on an abstract model class, enabling rapid prototyping of new cluster methods or model specifications.
- Standard plotting tools for model evaluation across methods (e.g., trajectories, cluster trajectories, model fit, metrics)
- Support for many cluster metrics through the packages clusterCrit, mclustcomp, and igraph.
- The structured and unified analysis approach enables simulation studies for comparing methods.
- Standardized model validation for all methods through bootstrapping or k-fold cross-validation.
Getting started

The `latrendData` dataset is included with the package and is used in all examples. The `plotTrajectories()` function can be used to visualize any longitudinal dataset, given the id and time are specified.

```r
data(latrendData)
head(latrendData)
options(latrend.id = "Id", latrend.time = "Time")
plotTrajectories(latrendData, response = "Y")
```

Discovering longitudinal clusters using the package involves the specification of the longitudinal cluster method that should be used.

```r
kmlMethod <- lcMethodKML("Y", nClusters = 3)
kmlMethod
```

The specified method is then estimated on the data using the generic estimation procedure function `latrend()`:

```r
model <- latrend(kmlMethod, data = latrendData)
```

Analyze the fitted model

```r
summary(model)
plot(model)
metric(model, c("WMAE", "BIC"))
qqPlot(model)
```

Create derivative method specifications for 1 to 5 clusters using the `lcMethods()` function. A series of methods can be estimated using `latrendBatch()`.

```r
kmlMethods <- lcMethods(kmlMethod, nClusters = 1:5)
models <- latrendBatch(kmlMethods, data = latrendData)
```

Determine the number of clusters through one or more internal cluster metrics. This can be done visually using the `plotMetric()` function.

```r
plotMetric(models, c("WMAE", "BIC"))
```

Vignettes

Further step-by-step instructions on how to use the package are described in the vignettes.

- See vignette("demo", package = "latrend") for an introduction to conducting a longitudinal cluster analysis on a example case study.
- See vignette("simulation", package = "latrend") for an example on conducting a simulation study.
- See vignette("validation", package = "latrend") for examples on applying internal cluster validation.
- See vignette("implement", package = "latrend") for examples on constructing your own cluster models.
Useful pages

Method specification: \texttt{lcMethod} \texttt{lcMethods}

Method estimation: \texttt{latrend} \texttt{latrendRep} \texttt{latrendBatch} \texttt{latrendBoot} \texttt{latrendCV} \texttt{latrend-parallel}

Model functions: \texttt{lcModel} \texttt{clusterTrajectories} \texttt{plotClusterTrajectories} \texttt{postprob} \texttt{trajectoryAssignments} \texttt{predictPostprob} \texttt{predictAssignments} \texttt{predict.lcModel} \texttt{predictForCluster} \texttt{fitted.lcModel} \texttt{fittedTrajectories}

Author(s)

Maintainer: Niek Den Teuling <niek.den.teuling@philips.com> (ORCID)

Other contributors:

• Steffen Pauws <s.c.pauws@tilburguniversity.edu> [contributor]
• Edwin van den Heuvel <e.r.v.d.heuvel@tue.nl> [contributor]
• Copyright © 2022 Koninklijke Philips N.V. [copyright holder]

See Also

Useful links:

• \url{https://github.com/philips-software/latrend}
• Report bugs at \url{https://github.com/philips-software/latrend/issues}

\begin{verbatim}
APPA                      Average posterior probability of assignment (APPA)
\end{verbatim}

Description

Computes the average posterior probability of assignment (APPA) for each cluster.

Usage

\texttt{APPA(object)}

Arguments

\begin{verbatim}
object  The model, of type \texttt{lcModel}.
\end{verbatim}

Value

The APPA per cluster, as a numeric vector of length \texttt{nClusters(object)}. Empty clusters will output NA.
as.data.frame.lcMethod

Convert lcMethod arguments to a list of atomic types

Description

Converts the arguments of a lcMethod to a named list of atomic types.

Usage

## S3 method for class 'lcMethod'
as.data.frame(x, ..., eval = TRUE, nullValue = NA, envir = NULL)

Arguments

- **x**: lcMethod to be coerced to a character vector.
- **...**: Additional arguments.
- **eval**: Whether to evaluate the arguments in order to replace expression if the resulting value is of a class specified in evalClasses.
- **nullValue**: Value to use to represent the NULL type. Must be of length 1.
- **envir**: The environment in which to evaluate the arguments. If NULL, the environment associated with the object is used. If not available, the parent.frame() is used.

Value

A single-row data.frame where each columns represents an argument call or evaluation.

See Also

Other lcMethod functions: [[,lcMethod-method, as.data.frame.lcMethods(), as.lcMethods(), as.list.lcMethod(), evaluate.lcMethod(), formula.lcMethod(), lcMethod-class, names, lcMethod-method, update.lcMethod()]

References


as.data.frame.lcMethods

Convert a list of lcMethod objects to a data.frame

Description

Converts a list of lcMethod objects to a data.frame.

Usage

## S3 method for class 'lcMethods'
as.data.frame(lcMethods)

Arguments

x
the lcMethods or list to be coerced to a data.frame.

... Additional arguments.

eval Whether to evaluate the arguments in order to replace expression if the resulting
value is of a class specified in evalClasses.

nullValue Value to use to represent the NULL type. Must be of length 1.

envir The environment in which to evaluate the arguments. If NULL, the environment
associated with the object is used. If not available, the parent.frame() is used.

Value

A data.frame with each row containing the argument values of a method object.

See Also

Other lcMethod functions: [,lcMethod-method,as.data.frame.lcMethod(),as.lcMethods(),
as.list.lcMethod(),evaluate.lcMethod(),formula.lcMethod(),lcMethod-class,lcMethod=method,
update.lcMethod()
as.lcMethods

Arguments

x
lcModels or a list of lcModel

Arguments passed to as.data.frame.lcMethod.

dots
whether to exclude columns which have the same value across all methods.

eval
Whether to evaluate the arguments in order to replace expression if the resulting value is of a class specified in evalClasses.

Value

A data.frame.

as.lcMethods

Convert a list of lcMethod objects to a lcMethods list

Description

Convert a list of lcMethod objects to a lcMethods list

Usage

as.lcMethods(x)

Arguments

x
A list of lcMethod objects.

Value

A lcMethods object.

See Also

Other lcMethod functions: [[,lcMethod-method, as.data.frame.lcMethods(), as.data.frame.lcMethod(),
as.list.lcMethod(), evaluate.lcMethod(), formula.lcMethod(), lcMethod-class, names,lcMethod-method,
update.lcMethod()]}
as.lcModels

Convert a list of lcModels to a lcModels list

Description
Convert a list of lcModels to a lcModels list

Usage
as.lcModels(x)

Arguments
x A list of lcModel objects, an lcModels object, or NULL.

Value
A lcModels object.

See Also
lcModels
Other lcModel list functions: lcModels, print.lcModels(), subset.lcModels()

as.list.lcMethod
Extract the method arguments as a list

Description
Extract the method arguments as a list

Usage
## S3 method for class 'lcMethod'
as.list(x, ..., args = names(x), eval = TRUE, expand = FALSE, envir = NULL)

Arguments
x The lcMethod object.
... Additional arguments.
args A character vector of argument names to select. Only available arguments are returned. Alternatively, a function or list of functions, whose formal arguments will be selected from the method.
eval Whether to evaluate the arguments.
clusterNames

expand
Whether to return all method arguments when "..." is present among the requested argument names.

envir
The environment in which to evaluate the arguments. If NULL, the environment associated with the object is used. If not available, the parent.frame() is used.

Value
A list with the argument calls or evaluated results depending on the value for eval.

See Also
Other lcMethod functions: [[,lcMethod-method, as.data.frame.lcMethods(), as.data.frame.lcMethod(), as.lcMethods(), evaluate.lcMethod(), formula.lcMethod(), lcMethod-class, names,lcMethod-method, update.lcMethod()]

Examples
data(latrendData)
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
as.list(method)
as.list(method, args = c("id", "time"))

if (require("kml")) {
  method <- lcMethodKML("Y", id = "Id", time = "Time")
as.list(method)

  # select arguments used by kml()
as.list(method, args = kml::kml)

  # select arguments used by either kml() or parALGO()
as.list(method, args = c(kml::kml, kml::parALGO))
}

clusterNames

Get the cluster names

Description
Get the cluster names

Usage
clusterNames(object, factor = FALSE)

Arguments

object The lcModel object.

factor Whether to return the cluster names as a factor.
clusterNames <-

Value

A character of the cluster names.

Examples

data(latrendData)
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
model <- latrend(method, latrendData)
clusterNames(model) # A, B

updateNames <-

Update the cluster names

Description

Update the cluster names

Usage

clusterNames(object) <- value

Arguments

object The lcModel object to update.
value The character with the new names.

Value

The updated lcModel object.

Examples

data(latrendData)
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
model <- latrend(method, latrendData, nClusters = 2)
clusterNames(model) <- c("Group 1", "Group 2")
clusterProportions

Proportional size of each cluster

Description

Obtain the proportional size per cluster, with sizes between 0 and 1. By default, the cluster proportions are determined from the cluster-averaged posterior probabilities of the fitted data (as computed by the `postprob()` function).

Usage

```r
## S4 method for signature 'lcModel'
clusterProportions(object, ...)
```

Arguments

- `object` The `lcModel` to obtain the proportions from.
- `...` Arguments passed on to `postprob`.

Value

A named numeric vector of length `nClusters(object)` with the proportional size of each cluster.

Implementation

Classes extending `lcModel` can override this method to return, for example, the exact estimated mixture proportions based on the model coefficients.

```r
setMethod("clusterProportions", "lcModelExt", function(object, ...) {
  # return cluster proportion vector
})
```

See Also

`clusterSizes`, `postprob`

Examples

```r
data(latrendData)
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
model <- latrend(method, latrendData, nClusters = 2)
clusterProportions(model)
```
clusterSizes

### Description

Obtain the size of each cluster, where the size is determined by the number of assigned trajectories to each cluster.

### Usage

```r
clusterSizes(object, ...)
```

### Arguments

- **object**: The `lcModel` object.
- **...**: Arguments passed on to `trajectoryAssignments`

### Details

The cluster sizes are computed from the trajectory cluster membership as decided by the `trajectoryAssignments()` function.

### Value

A named integer vector of length `nClusters(object)` with the number of assigned trajectories per cluster.

### See Also

- `clusterProportions`
- `trajectoryAssignments`

### Examples

```r
data(latrendData)
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
model <- latrend(method, latrendData, nClusters = 2)
clusterSizes(model)
```
clusterTrajectories

Extract the cluster trajectories

Description

Extracts a data frame of all cluster trajectories.

Usage

```r
## S4 method for signature 'lcModel'
clusterTrajectories(object, at = time(object), what = "mu", ...)
```

Arguments

- `object`: The lcModel object.
- `at`: An optional vector of the times at which to compute the cluster trajectory predictions.
- `what`: The distributional parameter to predict. By default, the mean response 'mu' is predicted. The cluster membership predictions can be obtained by specifying what = 'mb'.
- `...`: Additional arguments.

Value

A data.frame of the estimated values at the given times. The first column should be named "Cluster". The second column should be time, with the name matching the timeVariable(object). The third column should be the expected value of the observations, named after the responseVariable(object).

See Also

Other model-specific methods: coef.lcModel(), converged(), deviance.lcModel(), df.residual.lcModel(), fitted.lcModel(), fittedTrajectories(), lcModel-class, logLik.lcModel(), model.frame.lcModel(), nobs.lcModel(), postprob(), predict.lcModel(), predictAssignments(), predictForCluster(), predictPostprob(), residuals.lcModel(), sigma.lcModel(), time.lcModel()

Examples

```r
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
model <- latrend(method, latrendData)

clusterTrajectories(model)

clusterTrajectories(model, at = c(0, .5, 1))
```
### coef.lcModel

**Extract lcModel coefficients**

#### Description

Extract the coefficients of the lcModel object, if defined. The returned set of coefficients depends on the underlying type of lcModel. The default implementation checks for the existence of a `coef()` function for the internal model as defined in the @model slot, returning the output if available.

#### Usage

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

#### Arguments

- **object**: The lcModel object.
- **...**: Additional arguments.

#### Value

A named numeric vector with all coefficients, or a matrix with each column containing the cluster-specific coefficients. If `coef()` is not defined for the given model, an empty numeric vector is returned.

#### Implementation

Classes extending lcModel can override this method to return model-specific coefficients.

```r
c coef.lcModelExt <- function(object, ...) {  
  # return model coefficients
}
```

#### See Also

Other model-specific methods: `clusterTrajectories()`, `converged()`, `deviance.lcModel()`, `df.residual.lcModel()`, `fitted.lcModel()`, `fittedTrajectories()`, `lcModel-class`, `logLik.lcModel()`, `model.frame.lcModel()`, `nobs.lcModel()`, `postprob()`, `predict.lcModel()`, `predictAssignments()`, `predictForCluster()`, `predictPostprob()`, `residuals.lcModel()`, `sigma.lcModel()`, `time.lcModel()`

#### Examples

```r
data(latrendData)  
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")  
model <- latrend(method, latrendData, nClusters = 2)  
coef(model)
```
**Description**

Note: this function should not be called directly, as it is part of the lcMethod fitting process. For fitting an lcMethod object to a dataset, see `latrend()`.

The `compose()` function of the lcMethod object evaluates and finalizes the lcMethod arguments.

The default implementation returns an updated object with all arguments having been evaluated.

**Usage**

```r
## S4 method for signature 'lcMethod'
compose(method, envir = NULL)
```

**Arguments**

- `method`: The lcMethod object.
- `envir`: The environment in which the lcMethod should be evaluated.

**Value**

The evaluated and finalized lcMethod object.

**Implementation**

In general, there is no need to extend this method for a specific method, as all arguments are automatically evaluated by the `compose,lcMethod` method.

However, in case there is a need to extend processing or to prevent evaluation of specific arguments (e.g., for handling errors), the method can be overridden for the specific lcMethod subclass.

```r
setMethod("compose", "lcMethodExample", function(method, envir = NULL) {
    newMethod <- callNextMethod()
    # further processing
    return(newMethod)
})
```

**Fitting procedure**

Each lcMethod subclass defines a type of methods in terms of a series of steps for estimating the method. These steps, as part of the fitting procedure, are executed by `latrend()` in the following order:

1. `compose()`: Evaluate and finalize the method argument values.
2. `validate()`: Check the validity of the method argument values in relation to the dataset.
3. `prepareData()`: Process the training data for fitting.
confusionMatrix

4. **preFit()**: Prepare environment for estimation, independent of training data.

5. **fit()**: Estimate the specified method on the training data, outputting an object inheriting from `lcModel`.

6. **postFit()**: Post-process the outputted `lcModel` object.

The result of the fitting procedure is an `lcModel` object that inherits from the `lcModel` class.

**See Also**

`evaluate.lcMethod`

---

### confusionMatrix

**Compute the posterior confusion matrix**

**Description**

Compute the posterior confusion matrix (PCM). The entry \((i, j)\) represents the probability (or number, in case of `scale = TRUE`) of a trajectory belonging to cluster \(i\) is assigned to cluster \(j\) under the specified trajectory cluster assignment strategy.

**Usage**

\[
\text{confusionMatrix}(\text{object, strategy = which.max, scale = TRUE, ...})
\]

**Arguments**

- `object`: The model, of type `lcModel`.
- `strategy`: The strategy for assigning trajectories to a specific cluster, see `trajectoryAssignments()`.
  - If `strategy = NULL`, the posterior probabilities are used as weights (analogous to a repeated evaluation of `strategy = which.weight`).
- `scale`: Whether to express the confusion in probabilities (`scale = TRUE`), or in terms of the number of trajectories.
- `...`: Arguments passed on to `trajectoryAssignments`

**Value**

A K-by-K confusion matrix with \(K = nClusters(\text{object})\).

**See Also**

`postprob` `clusterProportions` `trajectoryAssignments` `APPA OCC`
Examples

```r
data(latrendData)

if (rlang::is_installed("lcmm")) {
  method <- lcMethodLcmm(
    fixed = Y ~ Time,
    mixture = ~ Time,
    random = ~ 1,
    id = "Id",
    time = "Time"
  )
  model <- latrend(method, latrendData)
  confusionMatrix(model)
}
```

---

### converged

**Check model convergence**

**Description**

Check convergence of the fitted `lcModel` object. The default implementation returns NA.

**Usage**

```r
## S4 method for signature 'lcModel'
converged(object, ...)
```

**Arguments**

- `object` The `lcModel` to check for convergence.
- `...` Additional arguments.

**Value**

Either logical indicating convergence, or a numeric status code.

**Implementation**

Classes extending `lcModel` can override this method to return a convergence status or code.

```r
setMethod("converged", "lcModelExt", function(object, ...) {
  # return convergence code
})
```

**See Also**

Other model-specific methods: `clusterTrajectories()`, `coef.lcModel()`, `deviance.lcModel()`, `df.residual.lcModel()`, `fitted.lcModel()`, `fittedTrajectories()`, `lcModel-class`, `logLik.lcModel()`, `model.frame.lcModel()`, `nobs.lcModel()`, `postprob()`, `predict.lcModel()`, `predictAssignments()`, `predictForCluster()`, `predictPostprob()`, `residuals.lcModel()`, `sigma.lcModel()`, `time.lcModel()`
**createTestDataFold**

Create the test fold data for validation

### Description

Create the test fold data for validation

### Usage

```r
createTestDataFold(data, trainData, id = getOption("latrend.id"))
```

### Arguments

- **data**
  - A `data.frame` representing the complete dataset.
- **trainData**
  - A `data.frame` representing the training data, which should be a subset of `data`.
- **id**
  - The trajectory identifier variable.

### See Also

- `createTrainDataFolds`
- Other validation methods: `createTestDataFolds()`, `createTrainDataFolds()`, `latrendBoot()`, `latrendCV()`, `lcModel-data-filters`

### Examples

```r
data(latrendData)

if (require("caret")) {
  trainDataList <- createTrainDataFolds(latrendData, id = "Id", folds = 10)
  testData1 <- createTestDataFold(latrendData, trainDataList[[1]], id = "Id")
}
```
createTestDataFolds  
Create all k test folds from the training data

Description
Create all k test folds from the training data

Usage
createTestDataFolds(data, trainDataList, ...)

Arguments
- data: A data.frame representing the complete dataset.
- trainDataList: A list of data.frame representing each of the data training folds. These should be derived from data.
- ...: Arguments passed to createTestDataFold.

See Also
Other validation methods: createTestDataFold(), createTrainDataFolds(), latrendBoot(), latrendCV(), lcModel-data-filters

Examples
```r
data(latrendData)
if (require("caret")) {
  trainDataList <- createTrainDataFolds(latrendData, folds = 10, id = "Id")
  testDataList <- createTestDataFolds(latrendData, trainDataList)
}
```

createTrainDataFolds  
Create the training data for each of the k models in k-fold cross validation evaluation

Description
Create the training data for each of the k models in k-fold cross validation evaluation

Usage
createTrainDataFolds(
data,
folds = 10L,
id =getOption("latrend.id"),
seed = NULL
)

**defineExternalMetric**

**Arguments**

- **data**
  - A `data.frame` representing the complete dataset.

- **folds**
  - The number of folds. By default, a 10-fold scheme is used.

- **id**
  - The trajectory identifier variable.

- **seed**
  - The seed to use, in order to ensure reproducible fold generation at a later moment.

**Value**

A list of `data.frame` of the folds training datasets.

**See Also**

Other validation methods: `createTestDataFolds()`, `createTestDataFold()`, `latrendBoot()`, `latrendCV()`, `lcModel-data-filters`

**Examples**

```r
data(latrendData)

if (require("caret")) {
  trainFolds <- createTrainDataFolds(latrendData, folds = 10, id = "Id")
  trainFolds <- createTrainDataFolds(latrendData, folds = 10, id = "Id", seed = 1)
}
```

---

**defineExternalMetric**  
*Define an external metric for lcModels*

**Description**

Define an external metric for lcModels

**Usage**

```r
defineExternalMetric(
  name,
  fun,
  warnIfExists = getOption("latrend.warnMetricOverride", TRUE)
)
```

**Arguments**

- **name**
  - The name of the metric.

- **fun**
  - The function to compute the metric, accepting a lcModel object as input.

- **warnIfExists**
  - Whether to output a warning when the metric is already defined.
**defineInternalMetric**

Define an internal metric for lcModels

**Usage**

```r
defineInternalMetric(
  name, 
  fun, 
  warnIfExists = getOption("latrend.warnMetricOverride", TRUE)
)
```

**Arguments**

- **name**: The name of the metric.
- **fun**: The function to compute the metric, accepting a lcModel object as input.
- **warnIfExists**: Whether to output a warning when the metric is already defined.

**Examples**

```r
defineInternalMetric("BIC", fun = BIC)

mae <- function(object) {
  mean(abs(residuals(object)))
}
defineInternalMetric("MAE", fun = mae)
```

**See Also**

Other metric functions: `defineInternalMetric()`, `externalMetric`, `lcModel`, `lcModel-method`, `getExternalMetricDefinition()`, `getExternalMetricNames()`, `getInternalMetricDefinition()`, `getInternalMetricNames()`, `metric()`
deviance.lcModel

## S3 method for class 'lcModel'
deviance(object, ...)

Arguments

- **object**: The lcModel object.
- **...**: Additional arguments.

Details

The default implementation checks for the existence of the deviance() function for the internal model, and returns the output, if available.

Value

A numeric with the deviance value. If unavailable, NA is returned.

See Also

stats::deviance metric

Other model-specific methods: clusterTrajectories(), coef.lcModel(), converged(), df.residual.lcModel(), fitted.lcModel(), fittedTrajectories(), lcModel-class, logLik.lcModel(), model.frame.lcModel(), nobs.lcModel(), postprob(), predict.lcModel(), predictAssignments(), predictForCluster(), predictPostprob(), residuals.lcModel(), sigma.lcModel(), time.lcModel()

---

df.residual.lcModel

## S3 method for class 'lcModel'
df.residual(object, ...)

Description

Extract the residual degrees of freedom from a lcModel

Usage

## S3 method for class 'lcModel'
df.residual(object, ...)

Description

Extract the residual degrees of freedom from a lcModel
estimationTime

Arguments

- **object**: The lcModel object.
- **...**: Additional arguments.

Value

A numeric with the residual degrees of freedom. If unavailable, NA is returned.

See Also

- stats::df.residual
- nobs
- residuals

Other model-specific methods: clusterTrajectories(), coef.lcModel(), converged(), deviance.lcModel(), fitted.lcModel(), fittedTrajectories(), lcModel-class, logLik.lcModel(), model.frame.lcModel(), nobls.lcModel(), postprob(), predict.lcModel(), predictAssignments(), predictForCluster(), predictPostprob(), residuals.lcModel(), sigma.lcModel(), time.lcModel()

---

estimationTime  
*Get the model estimation time*

Description

Get the estimation time of the model, determined by the time taken for the associated fit() function to finish.

Usage

```r
## S4 method for signature 'lcModel'
estimationTime(object, unit = "secs", ...)
```

```r
## S4 method for signature 'lcModels'
estimationTime(object, unit = "secs", ...)
```

```r
## S4 method for signature 'list'
estimationTime(object, unit = "secs", ...)
```

Arguments

- **object**: The list of lcModel objects.
- **unit**: The time unit in which the estimation time should be outputted. By default, estimation time is in seconds. For accepted units, see base::difftime.
- **...**: Additional arguments.

Value

A numeric representing the model estimation time, in the specified unit.
evaluate.lcMethod

Examples

data(latrendData)
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
model <- latrend(method, latrendData)
estimationTime(model)
estimationTime(model, unit = 'mins')
estimationTime(model, unit = 'days')

evaluate.lcMethod

Substitute the call arguments for their evaluated values

Description

Substitutes the call arguments if they can be evaluated without error.

Usage

## S3 method for class 'lcMethod'
evaluate(
  object,
  classes = "ANY",
  try = TRUE,
  exclude = character(),
  envir = NULL
)

Arguments

object The lcMethod object.
classes Substitute only arguments with specific class types. By default, all types are substituted.
try Whether to try to evaluate arguments and ignore errors (the default), or to fail on any argument evaluation error.
exclude Arguments to exclude from evaluation.
envir The environment in which to evaluate the arguments. If NULL, the environment associated with the object is used. If not available, the parent.frame() is used.

Value

A new lcMethod object with the substituted arguments.

See Also

compose

Other lcMethod functions: [[,lcMethod-method,as.data.frame.lcMethods(),as.data.frame.lcMethod(),as.lcMethods(),as.list.lcMethod(),formula.lcMethod(),lcMethod-class,names,lcMethod-method,update.lcMethod()]
Description

Compute one or more external metrics for two or more lcModel objects.

Note that there are many external metrics available, and there exists no external metric that works best in all scenarios. It is recommended to carefully consider which metric is most appropriate for your use case.

Many of the external metrics depend on implementations in other packages:

- clusterCrit (Desgraupes 2018)
- mclustcomp (You 2018)
- igraph (Csardi and Nepusz 2006)
- psych (Revelle 2019)

See `mclustcomp::mclustcomp()` for a grouped overview of similarity metrics.

Call `getInternalMetricNames()` to retrieve the names of the defined internal metrics. Call `getExternalMetricNames()` to retrieve the names of the defined internal metrics.

Usage

```r
## S4 method for signature 'lcModel,lcModel'
externalMetric(
  object,
  object2,
  name = getOption("latrend.externalMetric"),
  ...
)
```

```r
## S4 method for signature 'lcModels,missing'
externalMetric(object, object2, name = "adjustedRand")
```

```r
## S4 method for signature 'lcModels,character'
externalMetric(object, object2 = "adjustedRand")
```

```r
## S4 method for signature 'lcModels,lcModel'
externalMetric(object, object2, name, drop = TRUE)
```

```r
## S4 method for signature 'list,lcModel'
externalMetric(object, object2, name, drop = TRUE)
```
Arguments

- **object**: The `lcModel`, `lcModels`, or list of `lcModel` objects to compute the metrics for.
- **object2**: The other `lcModel` to compare with.
- **name**: The name(s) of the external metric(s) to compute. If no names are given, the names specified in the `latrend.externalMetric` option (none by default) are used.
- **drop**: Whether to return a numeric vector instead of a data.frame in case of a single metric.

Details

List of currently supported external metrics:

<table>
<thead>
<tr>
<th>Metric name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>adjustedRand</td>
<td>Adjusted Rand index. Based on the Rand index, but adjusted for agreements occurring by chance. A score of 1 indicates a perfect agreement, whereas a score of 0 indicates an agreement no better than chance.</td>
</tr>
<tr>
<td>CohensKappa</td>
<td>Cohen’s kappa. A partitioning agreement metric correcting for random chance. A score of 1 indicates a perfect agreement, whereas a score of 0 indicates an agreement no better than chance.</td>
</tr>
<tr>
<td>F</td>
<td>F-score, also referred to as the Sørensen–Dice Coefficient, or Dice similarity coefficient</td>
</tr>
<tr>
<td>F1</td>
<td>F1-score, also referred to as the Sørensen–Dice Coefficient, or Dice similarity coefficient</td>
</tr>
<tr>
<td>FolkesMallows</td>
<td>Fowlkes-Mallows index</td>
</tr>
<tr>
<td>Hubert</td>
<td>Hubert index</td>
</tr>
<tr>
<td>Jaccard</td>
<td>Jaccard index</td>
</tr>
<tr>
<td>jointEntropy</td>
<td>Joint entropy between model assignments</td>
</tr>
<tr>
<td>Kulczynski</td>
<td>Kulczynski index</td>
</tr>
<tr>
<td>MaximumMatch</td>
<td>Maximum match measure</td>
</tr>
<tr>
<td>McNemar</td>
<td>McNemar statistic</td>
</tr>
<tr>
<td>MeilaHeckerman</td>
<td>Meila-Heckerman measure</td>
</tr>
<tr>
<td>Mirkin</td>
<td>Mirkin metric</td>
</tr>
<tr>
<td>MI</td>
<td>Mutual information</td>
</tr>
<tr>
<td>NMI</td>
<td>Normalized mutual information</td>
</tr>
<tr>
<td>NSJ</td>
<td>Normalized version of splitJoin. The proportion of edits relative to the maximum changes (twice the number of ids)</td>
</tr>
<tr>
<td>NVI</td>
<td>Normalized variation of information</td>
</tr>
<tr>
<td>Overlap</td>
<td>Overlap coefficient, also referred to as the Szymkiewicz–Simpson coefficient</td>
</tr>
<tr>
<td>PD</td>
<td>Partition difference</td>
</tr>
<tr>
<td>Phi</td>
<td>Phi coefficient.</td>
</tr>
<tr>
<td>precision</td>
<td>precision</td>
</tr>
<tr>
<td>Rand</td>
<td>Rand index</td>
</tr>
<tr>
<td>recall</td>
<td>recall</td>
</tr>
<tr>
<td>RogersTanimoto</td>
<td>Rogers-Tanimoto dissimilarity</td>
</tr>
<tr>
<td>RussellRao</td>
<td>Russell-Rao dissimilarity</td>
</tr>
<tr>
<td>SMC</td>
<td>Simple matching coefficient</td>
</tr>
<tr>
<td>splitJoin</td>
<td>total split-join index</td>
</tr>
<tr>
<td>splitJoin.ref</td>
<td>Split-join index of the first model to the second model. In other words, it is the edit-distance between the two partitions</td>
</tr>
<tr>
<td>SokalSneath1</td>
<td>Type-1 Sokal-Sneath dissimilarity</td>
</tr>
<tr>
<td>SokalSneath2</td>
<td>Type-2 Sokal-Sneath dissimilarity</td>
</tr>
<tr>
<td>VI</td>
<td>Variation of information</td>
</tr>
</tbody>
</table>
Wallace1  Type-1 Wallace criterion
Wallace2  Type-2 Wallace criterion
WMSSE    Weighted minimum sum of squared errors between cluster trajectories
WMSE     Weighted minimum mean of squared errors between cluster trajectories
WMMAE    Weighted minimum mean of absolute errors between cluster trajectories

Value

For `externalMetric(lcModel, lcModel)`: A numeric vector of the computed metrics.
A named numeric vector containing the computed model metrics.
For `externalMetric(lcModels)`: A distance matrix of class `dist` representing the pairwise comparisons.
For `externalMetric(lcModels, name)`: A distance matrix of class `dist` representing the pairwise comparisons.
For `externalMetric(lcModels, lcModel)`: A named numeric vector or `data.frame` containing the computed model metrics.
For `externalMetric(list, lcModel)`: A named numeric vector or `data.frame` containing the computed model metrics.

Implementation

See the documentation of the `defineExternalMetric()` function for details on how to define your own external metrics.

References


See Also

metric

Other metric functions: `defineExternalMetric()`, `defineInternalMetric()`, `getExternalMetricDefinition()`, `getExternalMetricNames()`, `getInternalMetricDefinition()`, `getInternalMetricNames()`, `metric()`

Examples

data(latrendData)
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
model2 <- latrend(method, latrendData, nClusters = 2)
model3 <- latrend(method, latrendData, nClusters = 3)

if (require("mclustcomp")) {
  externalMetric(model2, model3, "adjustedRand")
}

Description

Note: this function should not be called directly, as it is part of the lcMethod fitting process. For fitting an lcMethod object to a dataset, see `latrend()`.

The `fit()` function of the lcMethod object estimates the model with the evaluated method specification, processed training data, and prepared environment.

Usage

```
## S4 method for signature 'lcMethod'
fit(method, data, envir, verbose)
```

Arguments

- **method**: An object inheriting from lcMethod with all its arguments having been evaluated and finalized.
- **data**: A data.frame representing the transformed training data.
- **envir**: The environment containing variables generated by `prepareData()` and `preFit()`.
- **verbose**: A `R.utils::Verbose` object indicating the level of verbosity.

Value

The fitted object inheriting from lcModel.
Implementation

This method should be implemented for all lcMethod subclasses.

```r
setMethod("fit", "lcMethodExample", function(method, data, envir, verbose) {
# estimate the model or cluster parameters
coefs <- FIT_CODE

# create the lcModel object
new("lcModelExample",
    method = method,
    data = data,
    model = coefs,
    clusterNames = make.clusterNames(method$nClusters)
})
})
```

Fitting procedure

Each lcMethod subclass defines a type of methods in terms of a series of steps for estimating the method. These steps, as part of the fitting procedure, are executed by `latrend()` in the following order:

1. `compose()`: Evaluate and finalize the method argument values.
2. `validate()`: Check the validity of the method argument values in relation to the dataset.
3. `prepareData()`: Process the training data for fitting.
4. `preFit()`: Prepare environment for estimation, independent of training data.
5. `fit()`: Estimate the specified method on the training data, outputting an object inheriting from `lcModel`.
6. `postFit()`: Post-process the outputted `lcModel` object.

The result of the fitting procedure is an `lcModel` object that inherits from the `lcModel` class.

---

**fitted.lcModel**

*Extract lcModel fitted values*

---

**Description**

Returns the cluster-specific fitted values for the given `lcModel` object. The default implementation calls `predict()` with `newdata = NULL`.

**Usage**

```r
## S3 method for class 'lcModel'
fitted(object, ..., clusters = trajectoryAssignments(object))
```
Arguments

object The lcModel object.

... Additional arguments.

clusters Optional cluster assignments per id. If unspecified, a matrix is returned containing the cluster-specific predictions per column.

Value

A numeric vector of the fitted values for the respective class, or a matrix of fitted values for each cluster.

Implementation

Classes extending lcModel can override this method to adapt the computation of the predicted values for the training data. Note that the implementation of this function is only needed when predict() and predictForCluster() are not defined for the lcModel subclass.

fitted.lcModelExt <- function(object, ..., clusters = trajectoryAssignments(object)) {
  pred = predict(object, newdata = NULL)
  transformFitted(pred = pred, model = object, clusters = clusters)
}

The transformFitted() function takes care of transforming the prediction input to the right output format.

See Also

fittedTrajectories plotFittedTrajectories stats::fitted predict.lcModel trajectoryAssignments transformFitted

Other model-specific methods: clusterTrajectories(), coef.lcModel(), converged(), deviance.lcModel(), df.residual.lcModel(), fittedTrajectories(), lcModel-class, logLik.lcModel(), model.frame.lcModel(), nobs.lcModel(), postprob(), predict.lcModel(), predictAssignments(), predictForCluster(), predictPostprob(), residuals.lcModel(), sigma.lcModel(), time.lcModel()

Examples

data(latrendData)
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
model <- latrend(method, latrendData)
fitted(model)
fittedTrajectories  Extract the fitted trajectories for all strata

Description

Extract the fitted trajectories for all strata

Usage

```r
## S4 method for signature 'lcModel'
fittedTrajectories(
  object,
  at = time(object),
  what = "mu",
  clusters = trajectoryAssignments(object),
  ...
)
```

Arguments

- `object`: The model.
- `at`: The time points at which to compute the id-specific trajectories. The default implementation merely filters the output of `fitted()`, so fitted values can only be outputted for times at which the model was trained.
- `what`: The distributional parameter to compute the response for.
- `clusters`: The cluster assignments for the strata to base the trajectories on.
- `...`: Additional arguments.

Details

The default implementation uses the output of `fitted()` of the respective model.

Value

A `data.frame` representing the fitted response per trajectory per moment in time for the respective cluster.

See Also

Other model-specific methods: `clusterTrajectories()`, `coef.lcModel()`, `converged()`, `deviance.lcModel()`, `df.residual.lcModel()`, `Fitted.lcModel()`, `lcModel-class`, `logLik.lcModel()`, `model.frame.lcModel()`, `mobs.lcModel()`, `postprob()`, `predict.lcModel()`, `predictAssignments()`, `predictForCluster()`, `predictPostprob()`, `residuals.lcModel()`, `sigma.lcModel()`, `time.lcModel()`
Examples

data(latrendData)
# Note: not a great example because the fitted trajectories
# are identical to the respective cluster trajectory
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
model <- latrend(method, latrendData)
fittedTrajectories(model)
fittedTrajectories(model, at = time(model)[c(1, 2)])

Description

Extracts the associated formula for the given distributional parameter.

Usage

## S3 method for class 'lcMethod'
formula(x, what = "mu", envir = NULL, ...)

Arguments

x The lcMethod object.
what The distributional parameter to which this formula applies. By default, the formula specifies "mu".
envir The environment in which to evaluate the arguments. If NULL, the environment associated with the object is used. If not available, the parent.frame() is used.
...
Additional arguments.

Value

The formula for the given distributional parameter.

See Also

Other lcMethod functions: [[,lcMethod-method,as.data.frame.lcMethods(),as.data.frame.lcMethod(),as.lcMethods(),as.list.lcMethod(),evaluate.lcMethod(),lcMethod-class,names,lcMethod-method,update.lcMethod()}

Examples

method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
formula(method) # Y ~ Time
### formula.lcModel

*Extract the formula of a lcModel*

**Description**

Get the formula associated with the fitted lcModel object. This is determined by the `formula` argument of the `lcMethod` specification that was used to fit the model.

**Usage**

```r
## S3 method for class 'lcModel'
formula(x, what = "mu", ...)
```

**Arguments**

- `x`: The `lcModel` object.
- `what`: The distributional parameter.
- `...`: Additional arguments.

**Value**

Returns the associated formula, or `response ~ 0` if not specified.

**See Also**

`stats::formula`

**Examples**

```r
data(latrendData)
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
model <- latrend(method, data = latrendData)
formula(model) # Y ~ Time
```

---

### generateLongData

*Generate longitudinal test data*

**Description**

Generate longitudinal test data
generateLongData

Usage

generateLongData(
    sizes = c(40, 60),
    fixed = Value ~ 1,
    cluster = ~1 + Time,
    random = ~1,
    id = getOption("latrend.id"),
    data = data.frame(Time = seq(0, 1, by = 0.1)),
    fixedCoefs = 0,
    clusterCoefs = cbind(c(-2, 1), c(2, -1)),
    randomScales = cbind(0.1, 0.1),
    rrandom = rnorm,
    noiseScales = c(0.1, 0.1),
    rnoise = rnorm,
    clusterNames = LETTERS[seq_along(sizes)],
    shuffle = FALSE,
    seed = NULL
)

Arguments

sizes Number of strata per cluster.
fixed Fixed effects formula.
cluster Cluster effects formula.
random Random effects formula.
id Name of the strata.
data Data with covariates to use for generation. Stratified data may be specified by
    adding a grouping column.
fixedCoefs Coefficients matrix for the fixed effects.
clusterCoefs Coefficients matrix for the cluster effects.
randomScales Standard deviations matrix for the size of the variance components (random
    effects).
rrandom Random sampler for generating the variance components at location 0.
noiseScales Scale of the random noise passed to rnoise. Either scalar or defined per cluster.
rnoise Random sampler for generating noise at location 0 with the respective scale.
clusterNames A character vector denoting the names of the generated clusters.
shuffle Whether to randomly reorder the strata in which they appear in the data.frame.
seed Optional seed to set for the PRNG. The set PRNG state persists after the function
    completes.

Examples

longdata <- generateLongData(
    sizes = c(40, 70), id = "Id",

cluster = ~poly(Time, 2, raw = TRUE),
clusterCoefs = cbind(c(1, 2, 5), c(-3, 4, .2))
)

if (require("ggplot2")) {
  plotTrajectories(longdata, response = "Value", id = "Id", time = "Time")
}

---

**getArgumentDefaults**  
*Default argument values for lcMethod subclass*

**Description**

Returns the default arguments associated with the respective lcMethod subclass. These arguments are automatically included into the lcMethod object during initialization.

**Usage**

```r
## S4 method for signature 'lcMethod'
getArgumentDefaults(object)
```

**Arguments**

- `object` The lcMethod object.

**Value**

A named list of argument values.

**Implementation**

Although implementing this method is optional, it prevents users from having to specify all arguments every time they want to create a method specification.

In this example, most of the default arguments are defined as arguments of the function lcMethodExample, which we can include in the list by calling `formals`. Copying the arguments from functions is especially useful when your method implementation is based on an existing function.

```r
setMethod("getArgumentDefaults", "lcMethodExample", function(object) {
  list(
    formals(lcMethodExample),
    formals(funFEM::funFEM),
    extra = Value ~ 1,
    tol = 1e-4,
    callNextMethod()
  )
})
```

It is recommended to add `callNextMethod()` to the end of the list. This enables inheriting the default arguments from superclasses.
getArgumentExclusions

See Also

lcMethod getArgumentExclusions


getArgumentExclusions  Arguments to be excluded for lcMethod subclass

Description

Returns the names of arguments that should be excluded during instantiation of the lcMethod.

Usage

## S4 method for signature 'lcMethod'
getArgumentExclusions(object)

Arguments

object  The lcMethod object.

Value

A character vector of argument names.

Implementation

This function only needs to be implemented if you want to avoid users from specifying redundant arguments or arguments that are set automatically or conditionally on other arguments.

```
setMethod("getArgumentExclusions", "lcMethodExample", function(object) {
  c("doPlot", "verbose", callNextMethod())
})
```

Adding `callNextMethod()` to the end of the return vector enables inheriting exclusions from superclasses.
getExternalMetricNames

**See Also**

lcMethod getArgumentExclusions


getExternalMetricDefinition

*Get the external metric definition*

**Description**

Get the external metric definition

**Usage**

getExternalMetricDefinition(name)

**Arguments**

name The name of the metric.

**Value**

The metric function, or NULL if not defined.

**See Also**

Other metric functions: defineExternalMetric(), defineInternalMetric(), externalMetric, lcModel, lcModel-method, getExternalMetricNames(), getInternalMetricDefinition(), getInternalMetricNames(), metric()

genericExternalMetricNames

*Get the names of the available external metrics*

**Description**

Get the names of the available external metrics

**Usage**

genericExternalMetricNames()
**getInternalMetricDefinition**

*Get the internal metric definition*

### Description

Get the internal metric definition

### Usage

```r
getInternalMetricDefinition(name)
```

### Arguments

- **name**
  
The name of the metric.

### Value

The metric function, or NULL if not defined.

### See Also

Other metric functions: `defineExternalMetric()`, `defineInternalMetric()`, `externalMetric`, `lcModel`, `lcModel-method`, `getExternalMetricDefinition()`, `getInternalMetricDefinition()`, `getInternalMetricNames()`, `metric()`

---

**getInternalMetricNames**

*Get the names of the available internal metrics*

### Description

Get the names of the available internal metrics

### Usage

```r
getInternalMetricNames()
```

### See Also

Other metric functions: `defineExternalMetric()`, `defineInternalMetric()`, `externalMetric`, `lcModel`, `lcModel-method`, `getExternalMetricDefinition()`, `getExternalMetricNames()`, `getInternalMetricDefinition()`, `metric()`
**getLabel**

Extract the method label.

**Description**

Extracts the assigned label from the given lcMethod or lcModel object. By default, the label is determined from the "label" argument of the lcMethod object. The label of an lcModel object is set upon estimation by `latrend()` to the label of its associated lcMethod object.

**Usage**

```r
## S4 method for signature 'lcMethod'
getLabel(object, ...)

## S4 method for signature 'lcModel'
getLabel(object, ...)
```

**Arguments**

- `object` The lcMethod or lcModel object.
- `...` Additional arguments.

**Value**

The extracted label, as character.

**See Also**

`getName` `getShortName`

**Examples**

```r
method <- lcMethodLMKM(Y ~ Time, time = "Time")
getLabel(method) # ""
getLabel(update(method, label = "v2")) # "v2"
```

---

**getLcMethod**

Get the method specification of a lcModel

**Description**

Get the lcMethod specification object that was used for fitting the given lcModel object.

**Usage**

```r
getLcMethod(object)
```
getName

Arguments

object The lcModel object.

Value

An lcMethod object.

See Also
getcAll.lcModel

Examples

method <- lcMethodRandom("Y", id = "Id", time = "Time")
model <- latrend(method, latrendData)
getLcMethod(model)

Description

Extract the full or shortened name of the given lcMethod or lcModel object. The name of the fitted lcModel is determined by its associated lcMethod name and label, unless specified otherwise.

Usage

## S4 method for signature 'lcMethod'
getName(object, ...)

## S4 method for signature 'lcMethod'
getShortName(object, ...)

## S4 method for signature 'lcModel'
getName(object)

## S4 method for signature 'lcModel'
getShortName(object)

Arguments

object The lcMethod or lcModel object.

... Additional arguments.

Value

A character name.
When implementing your own lcMethod subclass, override these methods to provide full and abbreviated names.

```r
setMethod("getName", "lcMethodExample", function(object) "example name")
setMethod("getShortName", "lcMethodExample", function(object) "EX")
```

Similar methods can be implemented for your lcModel subclass, however in practice this is not needed as the names are determined by default from the lcMethod object that was used to fit the lcModel object.

**See Also**

`getLabel`

**Examples**

```r
method <- lcMethodLMKM(Y ~ Time)
getName(method) # "lm-kmeans"
method <- lcMethodLMKM(Y ~ Time)
getShortName(method) # "LMKM"
```

### ids

*Get the trajectory ids on which the model was fitted*

#### Description

Get the trajectory ids on which the model was fitted

#### Usage

```r
ids(object)
```

#### Arguments

- `object`: The lcModel object.

#### Details

The order returned by `ids(object)` determines the id order for any output involving id-specific values, such as in `trajectoryAssignments()` or `postprob()`.

#### Value

A character vector or integer vector of the identifier for every fitted trajectory.
Examples

```r
data(latrendData)
method <- lcMethodRandom("Y", id = "Id", time = "Time")
model <- latrend(method, latrendData)
ids(model) # 1, 2, ..., 200
```

---

**idVariable**  
*Extract the trajectory identifier variable*

### Description

Extracts the trajectory identifier variable (i.e., column name) from the given object.

### Usage

```r
## S4 method for signature 'lcMethod'
idVariable(object, ...)
```

```r
## S4 method for signature 'lcModel'
idVariable(object)
```

### Arguments

- `object` The object to extract the variable from.
- `...` Not used.

### Value

The trajectory identifier name, as character.

### See Also

Other `lcModel` variables: `responseVariable()`, `timeVariable()`

### Examples

```r
method <- lcMethodLMKM(Y ~ Time, id = "Traj")
idVariable(method) # "Traj"
```

```r
method <- lcMethodRandom("Y", id = "Id", time = "Time")
model <- latrend(method, latrendData)
idVariable(model) # "Id"
```
### initialize,lcMethod-method

_**lcMethod initialization**_

#### Description

Initialization of lcMethod objects, converting arbitrary arguments to arguments as part of an lcMethod object.

#### Usage

```r
## S4 method for signature 'lcMethod'
initialize(.Object, ...)
```

#### Arguments

- `...`: Other method arguments.

#### Examples

```r
new("lcMethodLMM", formula = Y ~ Time, id = "Id", time = "Time")
```

---

### latrend

_Cluster longitudinal data_

#### Description

Fit a longitudinal cluster method to the given training data, according to the specification provided by the lcMethod object.

This function runs all steps as part of the method fitting procedure.

#### Usage

```r
latrend(
  method,
  data,
  ...,
  envir = NULL,
  verbose =getOption("latrend.verbose")
)
```
Arguments

- **method**: An `lcMethod` object specifying the longitudinal cluster method to apply, or the name (as character) of an `lcMethod` subclass. See `lcMethod` for details.
- **data**: The data frame to which to apply the method. Inputs supported by `trajectories()` can also be used.
- **...**: Any other arguments to update the `lcMethod` definition with.
- **envir**: The environment in which to evaluate the method arguments (by `compose()`). This environment is also used to evaluate the data argument if it is of type `call`.
- **verbose**: The level of verbosity. Either an object of class `Verbose` (see `R.utils::Verbose` for details), a logical indicating whether to show basic computation information, a numeric indicating the verbosity level (see `Verbose`), or one of `c('info', 'fine', 'finest')`.

Details

If a seed value is specified in the `lcMethod` object or arguments to `latrend`, this seed is set using `set.seed` prior to the cluster preparation step.

Value

A `lcModel` object representing the fitted model.

See Also

Other longitudinal cluster fit functions: `latrendBatch()`, `latrendBoot()`, `latrendCV()`, `latrendRep()`

Examples

```r
data(latrendData)
model <- latrend(lcMethodLMKM(Y ~ Time, id = "Id", time = "Time"), data = latrendData)

model <- latrend("lcMethodLMKM", formula = Y ~ Time, id = "Id", time = "Time", data = latrendData)

method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
model <- latrend(method, data = latrendData, nClusters = 3)
model <- latrend(method, data = latrendData, nClusters = 3, seed = 1)
```
Windows

On Windows, the parallel-package can be used to define parallel socket workers.

```r
nCores <- parallel::detectCores(logical = FALSE)
cl <- parallel::makeCluster(nCores)
```

Then, register the cluster as the parallel back-end using the doParallel package:

```r
doParallel::registerDoParallel(cl)
```

If you defined your own lcMethod or lcModel extension classes, make sure to load them on the workers as well. This can be done, for example, using:

```r
parallel::clusterEvalQ(cl,
  expr = setClass('lcMethodMyImpl', contains = "lcMethod"))
```

Unix

On Unix systems, it is easier to setup parallelization as the R process is forked. In this example we use the doMC package:

```r
nCores <- parallel::detectCores(logical = FALSE)
doMC::registerDoMC(nCores)
```

See Also

latrendRep, latrendBatch, latrendBoot, latrendCV

Examples

```r
data(latrendData)

# parallel latrendRep()
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
models <- latrendRep(method, data = latrendData, .rep = 5, parallel = TRUE)

# parallel latrendBatch()
methods <- lcMethods(method, nClusters = 1:3)
models <- latrendBatch(methods, data = latrendData, parallel = TRUE)
```
latrendBatch

Cluster longitudinal data for a list of method specifications

Description

Fit a list of longitudinal cluster methods on one or more datasets.

Usage

latrendBatch(
  methods,
  data,
  cartesian = TRUE,
  seed = NULL,
  parallel = FALSE,
  errorHandling = "stop",
  envir = NULL,
  verbose = getOption("latrend.verbose")
)

Arguments

- methods: A list of lcMethod objects.
- data: The dataset(s) to which to fit the respective lcMethod on. Either a data.frame, matrix, list or an expression evaluating to one of the supported types. Multiple datasets can be supplied by encapsulating the datasets using data = .(df1, df2, ..., dfN). Doing this results in a more readable call associated with each fitted lcModel object.
- cartesian: Whether to fit the provided methods on each of the datasets. If cartesian=FALSE, only a single dataset may be provided or a list of data matching the length of methods.
- seed: Sets the seed for generating the respective seed for each of the method fits. Seeds are only set for methods without a seed argument.
- parallel: Whether to enable parallel evaluation. See latrend-parallel. Method evaluation and dataset transformation is done on the calling thread.
- errorHandling: Whether to "stop" on an error, or to "remove" evaluations that raised an error.
- envir: The environment in which to evaluate the lcMethod arguments.
- verbose: The level of verbosity. Either an object of class Verbose (see R.utils::Verbose for details), a logical indicating whether to show basic computation information, a numeric indicating the verbosity level (see Verbose), or one of c('info', 'fine', 'finest').

Details

Methods and datasets are evaluated and validated prior to any fitting. This ensures that the batch estimation fails as early as possible in case of errors.
Value

A lcModels object. In case of a model fit error under errorHandling = pass, a list is returned.

See Also

lcMethods

Other longitudinal cluster fit functions: latrendBoot(), latrendCV(), latrendRep(), latrend()

Examples

data(latrendData)
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
methods <- lcMethods(method, nClusters = 1:3)
models <- latrendBatch(methods, data = latrendData)

# different dataset per method
models <- latrendBatch(lcMethods(refMethod, nClusters = 1:2),
                      data = .(
                      subset(latrendData, Time > .5),
                      subset(latrendData, Time < .5)
                      )
)

latrendBoot

Cluster longitudinal data using bootstrapping

Description

Performs bootstrapping, generating samples from the given data at the id level, fitting a lcModel to each sample.

Usage

latrendBoot(
method,
data,
samples = 50,
seed = NULL,
parallel = FALSE,
errorHandling = "stop",
envir = NULL,
verbose =getOption("latrend.verbose")
)
**Arguments**

- **method**: An lcMethod object specifying the longitudinal cluster method to apply, or the name (as character) of an lcMethod subclass. See lcMethod for details.
- **data**: A data.frame.
- **samples**: The number of bootstrap samples to evaluate.
- **seed**: The seed to use. Optional.
- **parallel**: Whether to enable parallel evaluation. See latrend-parallel. Method evaluation and dataset transformation is done on the calling thread.
- **errorHandling**: Whether to "stop" on an error, or to "remove" evaluations that raised an error.
- **envir**: The environment in which to evaluate the method arguments (by compose()). This environment is also used to evaluate the data argument if it is of type call.
- **verbose**: The level of verbosity. Either an object of class Verbose (see R.utils::Verbose for details), a logical indicating whether to show basic computation information, a numeric indicating the verbosity level (see Verbose), or one of c('info', 'fine', 'finest').

**Value**

A lcModels object of length samples.

**See Also**

Other longitudinal cluster fit functions: latrendBatch(), latrendCV(), latrendRep(), latrend()

Other validation methods: createTestDataFolds(), createTestDataFold(), createTrainDataFolds(), latrendCV(), lcModel-data-filters

**Examples**

```r
data(latrendData)
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
model <- latrendBoot(method, latrendData, samples = 10)
```

---

**latrendCV**

*Cluster longitudinal data over k folds*

**Description**

Apply k-fold cross validation for internal cluster validation. Creates k random subsets ("folds") from the data, estimating a model for each of the k-1 combined folds.
latrendCV

Usage

latrendCV(
  method,  # An lcMethod object specifying the longitudinal cluster method to apply, or the
  data,    # name (as character) of an lcMethod subclass. See lcMethod for details.
  folds = 10,  # The number of folds. Ten folds by default.
  seed = NULL,  # The seed to use. Optional.
  parallel = FALSE,  # Whether to enable parallel evaluation. See latrend-parallel. Method evaluation
  errorHandling = "stop",  # and dataset transformation is done on the calling thread.
  envir = NULL,  # Whether to "stop" on an error, or to "remove' evaluations that raised an error.
  verbose = getOption("latrend.verbose")  # The level of verbosity. Either an object of class Verbose (see R.utils::Verbose
)  # for details), a logical indicating whether to show basic computation information,
)

Arguments

  method      An lcMethod object specifying the longitudinal cluster method to apply, or the
                  name (as character) of an lcMethod subclass. See lcMethod for details.
  data        A data.frame.
  folds       The number of folds. Ten folds by default.
  seed        The seed to use. Optional.
  parallel    Whether to enable parallel evaluation. See latrend-parallel. Method evaluation
                  and dataset transformation is done on the calling thread.
  errorHandling Whether to "stop" on an error, or to "remove' evaluations that raised an error.
  envir       The environment in which to evaluate the method arguments (by compose()).
                  This environment is also used to evaluate the data argument if it is of type call.
  verbose     The level of verbosity. Either an object of class Verbose (see R.utils::Verbose
                  for details), a logical indicating whether to show basic computation information,
                  a numeric indicating the verbosity level (see Verbose), or one of c('info',
                  'fine', 'finest').

Value

A lcModels object of containing the folds training models.

See Also

Other longitudinal cluster fit functions: latrendBatch(), latrendBoot(), latrendRep(), latrend()
Other validation methods: createTestDataFolds(), createTestDataFold(), createTrainDataFolds(),
latrendBoot(), lcModel-data-filters

Examples

data(latrendData)
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")

if (require("caret")) {
  model <- latrendCV(method, latrendData, folds = 5)
latrendData

model <- latrendCV(method, subset(latrendData, Time < .5), folds = 5, seed = 1)
}

latrendData

Artificial longitudinal dataset comprising three classes

Description

An artificial longitudinal dataset comprising 200 trajectories belonging to one of 3 classes. Each trajectory deviates in intercept and slope from its respective class trajectory.

Usage

latrendData

Format

A data.frame comprising longitudinal observations from 200 trajectories. Each row represents the observed value of a trajectory at a specific moment in time.

Id  integer: The trajectory identifier.
Time  numeric: The measurement time, between 0 and 2.
Y  numeric: The observed value at the respective time Time for trajectory Id.
Class  factor: The reference class.

Source

This dataset was generated using generateLongData.

See Also

generateLongData

Examples

data(latrendData)

if (require("ggplot2")) {
  plotTrajectories(latrendData, id = "Id", time = "Time", response = "Y")

  # plot according to the reference class
  plotTrajectories(latrendData, id = "Id", time = "Time", response = "Y", cluster = "Class")
}
latrendRep  

Cluster longitudinal data repeatedly

Description

Performs a repeated fit of the specified latrend model on the given data.

Usage

latrendRep(
  method,  
data,  
  .rep = 10,  
  ...,  
  .errorHandling = "stop",  
  .seed = NULL,  
  .parallel = FALSE,  
  envir = NULL,  
  verbose = getOption("latrend.verbose")
)

Arguments

  method        An lcMethod object specifying the longitudinal cluster method to apply, or the name (as character) of an lcMethod subclass. See lcMethod for details.
  data          The data.frame to which to apply the method. Inputs supported by trajectories() can also be used.
  .rep          The number of repeated fits.
  ...           Any other arguments to update the lcMethod definition with.
  .errorHandling Whether to "stop" on an error, or to "remove' evaluations that raised an error.
  .seed         Set the seed for generating the respective seed for each of the repeated fits.
  .parallel     Whether to use parallel evaluation. See latrend-parallel.
  envir         The environment in which to evaluate the method arguments (by compose()). This environment is also used to evaluate the data argument if it is of type call.
  verbose       The level of verbosity. Either an object of class Verbose (see R.utils::Verbose for details), a logical indicating whether to show basic computation information, a numeric indicating the verbosity level (see Verbose), or one of c('info', 'fine', 'finest').

Details

This method is faster than repeatedly calling latrend as it only prepares the data via prepareData() once.
Value

A lcModels object containing the resulting models.

See Also

Other longitudinal cluster fit functions: \texttt{latrendBatch()}, \texttt{latrendBoot()}, \texttt{latrendCV()}, \texttt{latrend()}

Examples

data(latrendData)
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
models <- latrendRep(method, data = latrendData, .rep = 5)  # 5 repeated runs

models <- latrendRep(method, data = latrendData, .seed = 1, .rep = 3)

Description

approx models have defined cluster trajectories at fixed moments in time, which should be interpolated. For a correct implementation, \texttt{lcApproxModel} requires the extending class to implement \texttt{clusterTrajectories(at=NULL)} to return the fixed cluster trajectories.

Usage

## S3 method for class 'lcApproxModel'
fitted(object, ..., clusters = trajectoryAssignments(object))

## S4 method for signature 'lcApproxModel'
predictForCluster(
  object,
  newdata,
  cluster,
  what = "mu",
  approxFun = approx,
  ...
)

Arguments

- \texttt{object} The \texttt{lcModel} object.
- \texttt{...} Additional arguments.
- \texttt{clusters} Optional cluster assignments per id. If unspecified, a matrix is returned containing the cluster-specific predictions per column.
lcMethod-class

Description

lcMethod objects represent the specification of a method for longitudinal clustering. Furthermore, the object class contains the logic for estimating the respective method.

You can specify a longitudinal cluster method through one of the method-specific constructor functions, e.g., lcMethodKML(), lcMethodLcmmGBTM(), or lcMethodDtwclust(). Alternatively, you can instantiate methods through methods::new(), e.g., by calling new("lcMethodKML", response = "Value"). In both cases, default values are specified for omitted arguments.

Details

Because the lcMethod arguments may be unevaluated, argument retrieval functions such as [[ accept an envir argument. A default environment can be assigned or obtained from a lcMethod object using the environment() function.

Slots

arguments A list representing the arguments of the lcMethod object. Arguments are not evaluated upon creation of the method object. Instead, arguments are stored similar to a call object, and are only evaluated when a method is fitted. Do not modify or access.

sourceCalls A list of calls for tracking the original call after substitution. Used for printing objects which require too many characters (e.g., function definitions, matrices). Do not modify or access.

Method arguments

An lcMethod objects represent the specification of a method with a set of configurable parameters (referred to as arguments).

Arguments can be of any type. It is up to the lcMethod implementation of validate() to ensure that the required arguments are present and are of the expected type.

Arguments can have almost any name. Exceptions include the names "data", "envir", and "verbose". Furthermore, argument names may not start with a period (".").

Arguments cannot be directly modified, i.e., lcMethod objects are immutable. Modifying an argument involves creating an altered copy through the update.lcMethod method.
Fitting procedure

Each lcMethod subclass defines a type of methods in terms of a series of steps for estimating the method. These steps, as part of the fitting procedure, are executed by `latrend()` in the following order:

1. `compose()`: Evaluate and finalize the method argument values.
2. `validate()`: Check the validity of the method argument values in relation to the dataset.
3. `prepareData()`: Process the training data for fitting.
4. `preFit()`: Prepare environment for estimation, independent of training data.
5. `fit()`: Estimate the specified method on the training data, outputting an object inheriting from `lcModel`.
6. `postFit()`: Post-process the outputted `lcModel` object.

The result of the fitting procedure is an `lcModel` object that inherits from the `lcModel` class.

Implementation

The base class `lcMethod` provides the logic for storing, evaluating, and printing the method parameters.

Subclasses of `lcMethod` differ only in the fitting procedure logic (see above).

To implement your own `lcMethod` subclass, you’ll want to implement at least the following functions:

- `fit()`: The main function for estimating your method.
- `getName()`: The name of your method.
- `getShortName()`: The abbreviated name of your method.
- `getArgumentDefaults()`: Sensible default argument values to your method.

For more complex methods, the additional functions as part of the fitting procedure (see the Fitting procedure section above) will be of use.

See Also

- environment


Other lcMethod functions: `[[`, `lcMethod-method`, `as.data.frame.lcMethods()`, `as.data.frame.lcMethod()`, `as.lcMethods()`, `as.list.lcMethod()`, `evaluate.lcMethod()`, `formula.lcMethod()`, `names.lcMethod-method`, `update.lcMethod()`
**Examples**

```r
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time", nClusters = 2)
method

method <- new("lcMethodLMKM", formula = Y ~ Time, id = "Id", time = "Time", nClusters = 2)

# get argument names
names(method)

# evaluate argument
method$nClusters

# create a copy with updated nClusters argument
method3 <- update(method, nClusters = 3)
```

---

**lcMethodAkmedoids**  
*Specify AKMedoids method*

**Description**

Specify AKMedoids method

**Usage**

```r
lcMethodAkmedoids(
  response,
  time = getOption("latrend.time"),
  id = getOption("latrend.id"),
  nClusters = 3,
  clusterCenter = median,
  crit = "Calinski_Harabasz",
  ...
)
```

**Arguments**

- **response**: The name of the response variable.
- **time**: The name of the time variable.
- **id**: The name of the trajectory identification variable.
- **nClusters**: The number of clusters to estimate.
- **clusterCenter**: A function for computing the cluster center representation.
- **crit**: Criterion to apply for internal model selection. Not applicable.
- **...**: Arguments passed to `akmedoids::akclustr`. The following external arguments are ignored: traj, id_field, k
lcMethodCrimCV

References


See Also


Examples

data(latrendData)
if (require("akmedoids")) {
  method <- lcMethodAkmedoids(response = "Y", time = "Time", id = "Id", nClusters = 3)
  model <- latrend(method, data = latrendData)
}

---

lcMethodCrimCV | Specify a zero-inflated repeated-measures GBTM method

Description

Specify a zero-inflated repeated-measures GBTM method

Usage

lcMethodCrimCV(
  response,
  time = getOption("latrend.time"),
  id = getOption("latrend.id"),
  nClusters = 2,
  ...
)

Arguments

response | The name of the response variable.

 time | The name of the time variable.

 id | The name of the trajectory identifier variable.

 nClusters | The number of clusters to estimate.

 ... | Arguments passed to crimCV::crimCV. The following external arguments are ignored: Dat, ng.
lcMethodDtwclust

Specify time series clustering via dtwclust

Description

Specify time series clustering via dtwclust

Usage

lcMethodDtwclust(
  response,
  time = getOption("latrend.time"),
  id = getOption("latrend.id"),
  nClusters = 2,
  ...)

References


See Also

lcMethodFeature

Feature-based clustering

Description

Feature-based clustering.

Usage

lcMethodFeature(
  response,
  representationStep,
  clusterStep,
  standardize = scale,
  center = meanNA,
  time =getOption("latrend.time"),
  id =getOption("latrend.id"),
  ...
)
Arguments

response The name of the response variable.
representationStep A function with signature function(method, data) that computes the representation per strata, returned as a matrix. Alternatively, representationStep is a pre-computed representation matrix.
clusterStep A function with signature function(repdata) that outputs a lcModel.
standardize A function to standardize the output matrix of the representation step. By default, the output is shifted and rescaled to ensure zero mean and unit variance.
center The function for computing the longitudinal cluster centers, used for representing the cluster trajectories.
time The name of the time variable.
id The name of the trajectory identification variable.
... Additional arguments.

Linear regression & k-means example

In this example we define a feature-based approach where each trajectory is represented using a linear regression model. The coefficients of the trajectories are then clustered using k-means. Note that this method is already implemented as lcMethodLMKM().

Representation step:

repStep <- function(method, data, verbose) {
  library(data.table)
  library(magrittr)
  xdata = as.data.table(data)
  coefdata <- xdata[, lm(method$formula, .SD)
    keyby = c(method$id)
  ]
  # exclude the id column
  coefmat <- subset(coefdata, select = -1)
  rownames(coefmat) <- coefdata[[method$id]]
  return(coefmat)
}

Cluster step:

clusStep <- function(method, data, repMat, envir, verbose) {
  km <- kmeans(repMat, centers = method$nClusters)
  lcModelPartition(
    response = method$response,
    data = data,
    trajectoryAssignments = km$cluster
  )
}
Now specify the method and fit the model:

data(latrendData)
method <- lcMethodFeature(
  formula = Y ~ Time,
  response = "Y",
  id = "Id",
  time = "Time",
  representationStep = repStep,
  clusterStep = clusStep
)

model <- latrend(method, data = latrendData)

See Also


----------

**lcMethodFlexmix**

*Method interface to flexmix()*

----------

**Description**

Wrapper to the flexmix() method from the flexmix package.

**Usage**

```r
lcMethodFlexmix(
  formula,           # A formula specifying the model.
  formula.mb = ~1,  # A formula specifying the class membership model. By default, an intercept-only model is used.
  time = getOption("latrend.time"),# The name of the time variable.
  id = getOption("latrend.id"),    # The name of the trajectory identifier variable.
  nClusters = 2,           # Additional arguments passed to flexmix.
  ...                      # Additional arguments passed to flexmix.
)
```

**Arguments**

- `formula`:
  - **Description**: A formula specifying the model.

- `formula.mb`:
  - **Description**: A formula specifying the class membership model. By default, an intercept-only model is used.

- `time`:
  - **Description**: The name of the time variable.

- `id`:
  - **Description**: The name of the trajectory identifier variable.
lcMethodFlexmixGBTM

nClusters  The number of clusters to estimate.

Arguments passed to `flexmix::flexmix`. The following arguments are ignored:
data, concomitant, k.

References


See Also

Other lcMethod package interfaces: *lcMethodFlexmixGBTM*

Examples

```r
data(latrendData)
if (require("flexmix")) {
  method <- lcMethodFlexmix(Y ~ Time, id = "Id", time = "Time", nClusters = 3)
  model <- latrend(method, latrendData)
}
```

Description

Fits a GBTM based on the `flexmix::FLXMRglm` driver.

Usage

```r
lcMethodFlexmixGBTM(
  formula,
  formula.mb = ~1,
  time =getOption("latrend.time"),
  id =getOption("latrend.id"),
  nClusters = 2,
  ...)
```

Arguments

- **formula**: A formula specifying the model.
- **formula.mb**: A formula specifying the class membership model. By default, an intercept-only model is used.
- **time**: The name of the time variable.
- **id**: The name of the trajectory identifier variable.
lcMethodFunction

Specify a custom method based on a function

Description

Specify a custom method based on a function

Usage

lcMethodFunction(
    response,
    fun,
    center = meanNA,
    time =getOption("latrend.time"),
    id =getOption("latrend.id"),
    name = "custom"
)

Arguments

response The name of the response variable.
fun The cluster function with signature (method, data) that returns a lcModel object.
center Optional function for computing the longitudinal cluster centers, with signature (x).
Specify a FunFEM method

Description

Specify a FunFEM method

Usage

lcMethodFunFEM(
  response,
  time = getOption("latrend.time"),
  id = getOption("latrend.id"),
  nClusters = 2,
  basis = function(time) fda::create.bspline.basis(time, nbasis = 10, norder = 4),
  ...
)
lcMethodGCKM

Arguments

- **response**: The name of the response variable.
- **time**: The name of the time variable.
- **id**: The name of the trajectory identifier variable.
- **nClusters**: The number of clusters to estimate.
- **basis**: The basis function. By default, a 3rd-order B-spline with 10 breaks is used.
- **...**: Arguments passed to `funFEM::funFEM`. The following external arguments are ignored: fd, K, disp, graph.

References


See Also


Examples

```r
data(latrendData)
if (require("funFEM") && require("fda")) {
  method <- lcMethodFunFEM("Y", id = "Id", time = "Time", nClusters = 3)
  model <- latrend(method, latrendData)

  method <- lcMethodFunFEM("Y",
    basis = function(time) {
      create.bspline.basis(time, nbasis = 10, norder = 4)
    }
  )
}
```

lcMethodGCKM | Two-step clustering through latent growth curve modeling and k-means

Description

Two-step clustering through latent growth curve modeling and k-means.
Usage

```r
lcMethodGCKM(
  formula,
  time = getOption("latrend.time"),
  id = getOption("latrend.id"),
  nClusters = 2,
  center = meanNA,
  standardize = scale,
  ...
)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>formula</td>
<td>Formula, including a random effects component for the trajectory. See <code>lme4::lmer</code> formula syntax.</td>
</tr>
<tr>
<td>time</td>
<td>The name of the time variable.</td>
</tr>
<tr>
<td>id</td>
<td>The name of the trajectory identifier variable.</td>
</tr>
<tr>
<td>nClusters</td>
<td>The number of clusters.</td>
</tr>
<tr>
<td>center</td>
<td>A function that computes the cluster center based on the original trajectories associated with the respective cluster. By default, the mean is computed.</td>
</tr>
<tr>
<td>standardize</td>
<td>A function to standardize the output matrix of the representation step. By default, the output is shifted and rescaled to ensure zero mean and unit variance.</td>
</tr>
</tbody>
</table>

... Arguments passed to `lme4::lmer`. The following external arguments are ignored: data, centers, trace.

See Also


Examples

```r
data(latrendData)

if (require("lme4")) {
  method <- lcMethodGCKM(Y ~ (Time | Id), id = "Id", time = "Time", nClusters = 3)
  model <- latrend(method, latrendData)
}
```
Specify a longitudinal k-means (KML) method

Usage

\[
\text{lcMethodKML}(\text{response}, \text{time} = \text{getOption("latrend.time")}, \text{id} = \text{getOption("latrend.id")}, \text{nClusters} = 2, ...)
\]

Arguments

- **response**: The name of the response variable.
- **time**: The name of the time variable.
- **id**: The name of the trajectory identifier variable.
- **nClusters**: The number of clusters to estimate.
- **...**: Arguments passed to \texttt{kml::parALGO} and \texttt{kml::kml}.

The following external arguments are ignored: \texttt{object}, \texttt{nbClusters}, \texttt{parAlgo}, \texttt{toPlot}, \texttt{saveFreq}

References


See Also

Other \texttt{lcMethod} implementations: \texttt{getArgumentDefaults()}, \texttt{getArgumentExclusions()}, \texttt{lcMethod-class}, \texttt{lcMethodAkmedoids}, \texttt{lcMethodCrimCV}, \texttt{lcMethodDtwclust}, \texttt{lcMethodFeature}, \texttt{lcMethodFunFEM}, \texttt{lcMethodFunction}, \texttt{lcMethodGCKM}, \texttt{lcMethodLMKM}, \texttt{lcMethodLcmmGBTM}, \texttt{lcMethodLcmmGMM}, \texttt{lcMethodMclustLLPA}, \texttt{lcMethodMixAK_GLMM}, \texttt{lcMethodMixtoolsGMM}, \texttt{lcMethodMixtoolsNPRM}, \texttt{lcMethodRandom}, \texttt{lcMethodStratify}

Examples

```r
if (require("kml")) {
  method <- lcMethodKML("Y", id = "Id", time = "Time", nClusters = 3)
  model <- latrend(method, latrendData)
}
```
lcMethodLcmmGBTM

Specify GBTM method

Description

Group-based trajectory modeling through fixed-effects modeling.

Usage

```r
lcMethodLcmmGBTM(
  fixed,
  mixture = ~1,
  classmb = ~1,
  time = getOption("latrend.time"),
  id = getOption("latrend.id"),
  nClusters = 2,
  init = "default",
  ...
)
```

Arguments

- `fixed` The fixed effects formula.
- `mixture` The mixture-specific effects formula. See `lcm::hlme` for details.
- `classmb` The cluster membership formula for the multinomial logistic model. See `lcm::hlme` for details.
- `time` The name of the time variable.
- `id` The name of the trajectory identifier variable. This replaces the subject argument of `lcm::hlme`.
- `nClusters` The number of clusters to fit. This replaces the `ng` argument of `lcm::hlme`.
- `init` Alternative for the `B` argument of `lcm::hlme`, for initializing the h1me fitting procedure. If "lme.random" (default): random initialization through a standard linear mixed model. Assigns a fitted standard linear mixed model enclosed in a call to random() to the `B` argument. If "lme", fits a standard linear mixed model and passes this to the `B` argument. If NULL or "default", the default `lcm::hlme` input for `B` is used.

The argument is ignored if the `B` argument is specified, or `nClusters = 1`.

... Arguments passed to `lcm::hlme`. The following arguments are ignored: data, fixed, random, mixture, subject, classmb, returndata, ng, verbose, subset.

References

lcMethodLcmmGMM


See Also


Examples

data(latrendData)
if (rlang::is_installed("lcmm")) {
  method <- lcMethodLcmmGBTM(
    fixed = Y ~ Time,
    mixture = ~ 1,
    id = "Id",
    time = "Time",
    nClusters = 3
  )
  gbtm <- latrend(method, data = latrendData)
  summary(gbtm)

  method <- lcMethodLcmmGBTM(
    fixed = Y ~ Time,
    mixture = ~ Time,
    id = "Id",
    time = "Time",
    nClusters = 3
  )
}

lcMethodLcmmGMM Specify GMM method using lcmm

Description

Growth mixture modeling through latent-class linear mixed modeling.

Usage

lcMethodLcmmGMM(
  fixed, 
  mixture = ~1, 
  random = ~1, 
  classmb = ~1, 
  time = getOption("latrend.time"), 
)
id = getOption("latrend.id"),
init = "lme",
nClusters = 2,
...)

Arguments

fixed The fixed effects formula.
mixture The mixture-specific effects formula. See lcmm::hlme for details.
random The random effects formula. See lcmm::hlme for details.
classmb The cluster membership formula for the multinomial logistic model. See lcmm::hlme for details.
time The name of the time variable.
id The name of the trajectory identifier variable. This replaces the subject argument of lcmm::hlme.
init Alternative for the B argument of lcmm::hlme, for initializing the hlme fitting procedure. If "lme.random" (default): random initialization through a standard linear mixed model. Assigns a fitted standard linear mixed model enclosed in a call to random() to the B argument. If "lme", fits a standard linear mixed model and passes this to the B argument. If NULL or "default", the default lcmm::hlme input for B is used.

The argument is ignored if the B argument is specified, or nClusters = 1.
nClusters The number of clusters to fit. This replaces the ng argument of lcmm::hlme.
... Arguments passed to lcmm::hlme. The following arguments are ignored: data, fixed, random, mixture, subject, classmb, returndata, ng, verbose, subset.

References


See Also

Examples

data(latrendData)

if (rlang::is_installed("lcmm")) {
    method <- lcMethodLcmmGMM(
        fixed = Y ~ Time,
        mixture = ~ Time,
        random = ~ 1,
        id = "Id",
        time = "Time",
        nClusters = 2
    )
    gmm <- latrend(method, data = latrendData)
    summary(gmm)
}

method <- lcMethodLcmmGMM(
    fixed = Y ~ Time,
    mixture = ~ Time,
    random = ~ Time,
    id = "Id",
    time = "Time",
    nClusters = 3
)

lcMethodLMKM

Two-step clustering through linear regression modeling and k-means

Description

Two-step clustering through linear regression modeling and k-means

Usage

lcMethodLMKM(
    formula,
    time = getOption("latrend.time"),
    id = getOption("latrend.id"),
    nClusters = 2,
    center = meanNA,
    standardize = scale,
    ...)

Arguments

formula A formula specifying the linear trajectory model.
time The name of the time variable.
id  The name of the trajectory identification variable.
nClusters  The number of clusters to estimate.
center  A function that computes the cluster center based on the original trajectories associated with the respective cluster. By default, the mean is computed.
standardize  A function to standardize the output matrix of the representation step. By default, the output is shifted and rescaled to ensure zero mean and unit variance.
...
Arguments passed to stats::lm. The following external arguments are ignored: x, data, control, centers, trace.


Examples

data(latrendData)
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time", nClusters = 3)
model <- latrend(method, latrendData)

lcMethodMclustLLPA  Longitudinal latent profile analysis

Description  Latent profile analysis or finite Gaussian mixture modeling.

Usage

lcMethodMclustLLPA(
  response,
  time =getOption("latrend.time"),
  id =getOption("latrend.id"),
  nClusters = 2,
  ...
)

Arguments

response  The name of the response variable.
time  The name of the time variable.
id  The name of the trajectory identifier variable.
nClusters  The number of clusters to estimate.
...
Arguments passed to mclust::Mclust. The following external arguments are ignored: data, G, verbose.
lcMethodMixAK_GLMM

References


See Also


Examples

data(latrendData)
if (require("mclust")) {
  method <- lcMethodMclustLLPA("Y", id = "Id", time = "Time", nClusters = 3)
  model <- latrend(method, latrendData)
}

lcMethodMixAK_GLMM Specify a GLMM iwht a normal mixture in the random effects

Description

Specify a GLMM iwht a normal mixture in the random effects

Usage

lcMethodMixAK_GLMM(
  fixed,
  random,
  time = getOption("latrend.time"),
  id = getOption("latrend.id"),
  nClusters = 2,
  ...
)

Arguments

fixed A formula specifying the fixed effects of the model, including the response. Creates the y and x arguments for the call to mixAK::GLMM_MCMC.
random A formula specifying the random effects of the model, including the random intercept. Creates the z and random.intercept arguments for the call to mixAK::GLMM_MCMC.
time The name of the time variable.
id The name of the trajectory identifier variable. This is used to generate the id vector argument for the call to mixAK::GLMM_MCMC.
lcMethodMixtoolsGMM

nClusters The number of clusters.

... Arguments passed to mixAK::GLMM_MCMC. The following external arguments are ignored: y, x, z, random.intercept, silent.

Note

This method currently does not appear to work under R 4.2 due to an error triggered by the mixAK package during fitting.

References


See Also


Examples

data(latrendData)
# this example only runs when the mixAK package is installed
try{
  method <- lcMethodMixAK_GLMM(fixed = Y ~ 1, random = ~ Time,
                               id = "Id", time = "Time", nClusters = 3)
  model <- latrend(method, latrendData)
  summary(model)
}

lcMethodMixtoolsGMM Specify mixed mixture regression model using mixtools

Description

Specify mixed mixture regression model using mixtools

Usage

lcMethodMixtoolsGMM(
  formula,
  time = getOption("latrend.time"),
  id = getOption("latrend.id"),
  nClusters = 2,
  ...
lcMethodMixtoolsNPRM

Arguments

- **formula**: Formula, including a random effects component for the trajectory. See `lme4::lmer` formula syntax.
- **time**: The name of the time variable.
- **id**: The name of the trajectory identifier variable.
- **nClusters**: The number of clusters.
- **...**: Arguments passed to `mixtools::regmixEM.mixed`. The following arguments are ignored: data, y, x, w, k, addintercept.fixed, verb.

References


See Also


Examples

```r
# Load data
data(latrendData)

# Specify non-parametric estimation for independent repeated measures

if (require("mixtools")) {
  method <- lcMethodMixtoolsNPRM(
    formula = Y ~ Time + (1 | Id),
    id = "Id", time = "Time",
    nClusters = 3,
    arb.R = FALSE
  )
}
```

lcMethodMixtoolsNPRM Specify non-parametric estimation for independent repeated measures

Description

Specify non-parametric estimation for independent repeated measures
Usage

```r
lcMethodMixtoolsNPRM(
  response,
  time = getOption("latrend.time"),
  id = getOption("latrend.id"),
  nClusters = 2,
  blockid = NULL,
  bw = NULL,
  h = NULL,
  ...
)
```

Arguments

- `response` The name of the response variable.
- `time` The name of the time variable.
- `id` The name of the trajectory identifier variable.
- `nClusters` The number of clusters to estimate.
- `blockid` See `mixtools::npEM`.
- `bw` See `mixtools::npEM`.
- `h` See `mixtools::npEM`.
- `...` Arguments passed to `mixtools::npEM`. The following optional arguments are ignored: data, x, mu0, verb.

References


See Also


Examples

```r
data(latrendData)
if (require("mixtools")) {
  method <- lcMethodMixtoolsNPRM("Y", id = "Id", time = "Time", nClusters = 3)
  model <- latrend(method, latrendData)
}
```
Specify a MixTVEM

Usage

lcMethodMixTVEM(
  formula,
  formula.mb = ~1,
  time = getOption("latrend.time"),
  id = getOption("latrend.id"),
  nClusters = 2,
  ...
)

Arguments

formula A formula excluding the time component. Time-invariant covariates are detected automatically as these are a special case in MixTVEM.

formula.mb A formula for cluster-membership prediction. Covariates must be time-invariant. Furthermore, the formula must contain an intercept.

time The name of the time variable.

id The name of the trajectory identifier variable.

nClusters The number of clusters. This replaces the numClasses argument of the TVEMMixNormal function call.

... Arguments passed to the TVEMMixNormal() function. The following optional arguments are ignored: doPlot, getSEs, numClasses.

Note

In order to use this method, you must download and source MixTVEM.R. See the reference below.

References

https://github.com/dziakj1/MixTVEM

lcMethodRandom

Specify a random-partitioning method

Description

Creates a model with random cluster assignments according to the random cluster proportions drawn from a Dirichlet distribution.

Usage

lcMethodRandom(
  response,
  alpha = 10,
  center = meanNA,
  time = getOption("latrend.time"),
  id = getOption("latrend.id"),
  nClusters = 2,
  name = "random",
  ...
)

Arguments

response The name of the response variable.
alpha The Dirichlet parameters. Either scalar or of length nClusters. The higher alpha, the more uniform the clusters will be.
center Optional function for computing the longitudinal cluster centers, with signature (x).
time The name of the time variable.
id The name of the trajectory identification variable.
nClusters The number of clusters.
name The name of the method.
... Additional arguments, such as the seed.
References


See Also


Examples

```r
data(latrendData)
method <- lcMethodRandom(response = "Y", id = "Id", time = "Time")
model <- latrend(method, latrendData)

# uniform clusters
method <- lcMethodRandom(
  alpha = 1e3,
  nClusters = 3,
  response = "Y",
  id = "Id",
  time = "Time"
)

# single large cluster
method <- lcMethodRandom(
  alpha = c(100, 1, 1, 1),
  nClusters = 4,
  response = "Y",
  id = "Id",
  time = "Time"
)
```

lcMethods

Generate a list of lcMethod objects

Description

Generates a list of lcMethod objects for all combinations of the provided argument values.

Usage

```r
lcMethods(method, ..., envir = NULL)
```
lcMethodStratify

Specify a stratification method

Description

Specify a stratification method

Usage

lcMethodStratify(
  response,
  stratify,
  center = meanNA,
  nClusters = NaN,
  clusterNames = NULL,
  time = getOption("latrend.time"),
  envir = NULL
)

Arguments

method
   The lcMethod to use as the template, which will be updated for each of the other arguments.

...  Any other arguments to update the lcMethod definition with. Values must be scalar, vector, list, or encapsulated in a .() call. Arguments wrapped in .() are passed as-is to the model call, ensuring a readable method. Arguments comprising a single symbol (e.g. a variable name) are interpreted as a constant. To force evaluation, specify arg=(var) or arg=force(var). Arguments of type vector or list are split across a series of method fit calls. Arguments of type scalar are constant across the method fits. If a list is intended to be passed as a constant argument, then specifying arg=(listObject) results in it being treated as such.

envir  The environment in which to evaluate the method arguments.

Value

A list of lcMethod objects.

Examples

data(latrendData)
baseMethod <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
methods <- lcMethods(baseMethod, nClusters = 1:6)
nclus <- 1:6
methods <- lcMethods(baseMethod, nClusters = nclus)

# list notation, useful for providing functions
methods <- lcMethods(baseMethod, nClusters = .(1, 3, 5))
length(methods) # 3
id = getOption("latrend.id"),
    name = "stratify"
)

Arguments

- **response**: The name of the response variable.
- **stratify**: An expression returning a number or factor value per trajectory, representing the cluster assignment. Alternatively, a function can be provided that takes separate trajectory data.frame as input.
- **center**: The function for computing the longitudinal cluster centers, used for representing the cluster trajectories.
- **nClusters**: The number of clusters. This is optional, as this can be derived from the largest assignment number by default, or the number of factor levels.
- **clusterNames**: The names of the clusters. If a factor assignment is returned, the levels are used as the cluster names.
- **time**: The name of the time variable.
- **id**: The name of the trajectory identification variable.
- **name**: The name of the method.

See Also


Examples

data(latrendData)
# Stratification based on the mean response level
method <- lcMethodStratify(
  "y",
  mean(Y) > 0,
  clusterNames = c("Low", "High"),
  id = "Id",
  time = "Time"
)
model <- latrend(method, latrendData)
summary(model)

# Stratification function
stratfun <- function(trajdata) {
  trajmean <- mean(trajdata$Y)
  factor(
    trajmean > 1.7,
    levels = c(FALSE, TRUE),
    labels = c("Low", "High")
  )
}
method <- lcMethodStratify("Y", stratfun, id = "Id", time = "Time")

# Multiple clusters
stratfun3 <- function(trajdata) {
  trajmean <- mean(trajdata$Y)
  cut(
    trajmean,
    c(-Inf, .5, 2, Inf),
    labels = c("Low", "Medium", "High")
  )
}
method <- lcMethodStratify("Y", stratfun3, id = "Id", time = "Time")

---

**lcModel-class lcModel class**

**Description**

Abstract class for defining estimated longitudinal cluster models.

**Arguments**

- **object** The lcModel object.
- **...** Any additional arguments.

**Details**

An extending class must implement the following methods to ensure basic functionality:

- `predict.lcModelExt`: Used to obtain the fitted cluster trajectories and trajectories.
- `postprob(lcModelExt)`: The posterior probability matrix is used to determine the cluster assignments of the trajectories.

For predicting the posterior probability for unseen data, the `predictPostprob()` should be implemented.

**Slots**

- **method** The lcMethod-class object specifying the arguments under which the model was fitted.
- **call** The call that was used to create this lcModel object. Typically, this is the call to `latrend()` or any of the other fitting functions.
- **model** An arbitrary underlying model representation.
- **data** A data.frame object, or an expression to resolves to the data.frame object.
- **date** The date-time when the model estimation was initiated.
- **id** The name of the trajectory identifier column.
time  The name of the time variable.
response The name of the response variable.
label  The label assigned to this model.
ids  The trajectory identifier values the model was fitted on.
times The exact times on which the model has been trained
clusterNames  The names of the clusters.
estimationTime  The time, in seconds, that it took to fit the model.
tag  An arbitrary user-specified data structure. This slot may be accessed and updated directly.

See Also

Other model-specific methods: clusterTrajectories(), coef.lcModel(), converged(), deviance.lcModel(),
df.residual.lcModel(), fitted.lcModel(), fittedTrajectories(), logLik.lcModel(), model.frame.lcModel(),
nobs.lcModel(), postprob(), predict.lcModel(), predictAssignments(), predictForCluster(),
predictPostprob(), residuals.lcModel(), sigma.lcModel(), time.lcModel()
lcModelPartition

Arguments

data  A data.frame representing the trajectory data.
response  The name of the response variable.
trajectoryAssignments  A vector of cluster membership per trajectory, a data.frame with an id column and "Cluster" column, or the name of the cluster membership column in the data argument. For vector input, the type must be factor, character, or integer (1 to nClusters). The order of the trajectory, and thus the respective assignments, is determined by the id column of the data. Provide a factor id column for the input data to ensure that the ordering is as you expect.
nClusters  The number of clusters. Should be NA for trajectory assignments of type factor.
clusterNames  The names of the clusters, or a function with input n outputting a character vector of names. If unspecified, the names are determined from the trajectoryAssignments argument.
time  The name of the time variable.
id  The name of the trajectory identification variable.
name  The name of the method.
center  The function for computing the longitudinal cluster centers, used for representing the cluster trajectories.
method  Optional lcMethod object that was used for fitting this model to the data.
model  An optional object to attach to the lcModelPartition object, representing the internal model that was used for obtaining the partition.
envir  The environment associated with the model. Used for evaluating the assigned data object by model.data.lcModel.

Examples

# comparing a model to the ground truth using the adjusted Rand index
data(latrendData)
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
model <- latrend(method, latrendData, nClusters = 3)

# extract the reference class from the Class column
trajLabels <- aggregate(Class ~ Id, head, 1, data = latrendData)
trajLabels$Cluster <- trajLabels$Class
refModel <- lcModelPartition(latrendData, response = "Y", trajectoryAssignments = trajLabels)
if (require("mclustcomp")) {
  externalMetric(model, refModel, "adjustedRand")
}
Construct a flat (named) list of lcModel objects

Description

The lcModels S3 class represents a list of lcModel objects. This makes it easier to work with a set of models in a more structured manner.

The lcModels() function takes the inputs and generates a named lcModels object containing a list of the input models. Duplicates are preserved.

Usage

lcModels(...)

Arguments

... lcModel, lcModels, or a recursive list of lcModel objects. Arguments may be named.

Value

A lcModels object containing all specified lcModel objects.

See Also

Other lcModel list functions: as.lcModels(), print.lcModels(), subset.lcModels()

Examples

data(latrendData)
lmkmMethod <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
lmkmModel <- latrend(lmkmMethod, latrendData)
rngMethod <- lcMethodRandom("Y", id = "Id", time = "Time")
rngModel <- latrend(rngMethod, latrendData)

lcModels(lmkmModel, rngModel)

dl <- defaults = c(lmkmModel, rngModel)
lcModelWeightedPartition

Create a lcModel with pre-defined weighted partitioning

Description

Create a lcModel with pre-defined weighted partitioning

Usage

```
lcModelWeightedPartition(
  data,
  response,
  weights,
  clusterNames = colnames(weights),
  time = getOption("latrend.time"),
  id = getOption("latrend.id"),
  name = "wpart"
)
```

Arguments

- `data`: A `data.frame` representing the trajectory data.
- `response`: The name of the response variable.
- `weights`: A `numIds x numClusters` matrix of partition probabilities.
- `clusterNames`: The names of the clusters, or a function with input `n` outputting a character vector of names.
- `time`: The name of the time variable.
- `id`: The name of the trajectory identification variable.
- `name`: The name of the method.

logLik.lcModel

Extract the log-likelihood of a lcModel

Description

Extract the log-likelihood of a lcModel

Usage

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

- `object`: The `lcModel` object.
- `...`: Additional arguments.

Details

The default implementation checks for the existence of the `logLik()` function for the internal model, and returns the output, if available.

Value

A numeric with the computed log-likelihood. If unavailable, `NA` is returned.

See Also

- `stats::logLik`

Other model-specific methods:
- `clusterTrajectories()`, `coef.lcModel()`, `converged()`, `deviance.lcModel()`, `df.residual.lcModel()`, `fitted.lcModel()`, `fittedTrajectories()`, `lcModel-class`, `model.frame.lcModel()`, `nobs.lcModel()`, `postprob()`, `predict.lcModel()`, `predictAssignments()`, `predictForCluster()`, `predictPostprob()`, `residuals.lcModel()`, `sigma.lcModel()`, `time.lcModel()`

Examples

```r
data(latrendData)
if (rlang::is_installed("lcmm")) {
  method <- lcMethodLcmmGBTM(
    fixed = Y ~ Time,
    mixture = ~ 1,
    id = "Id",
    time = "Time",
    nClusters = 3
  )
  gbtm <- latrend(method, data = latrendData)
  logLik(gbtm)
}
```

Description

Select the `lcModel` with the highest metric value

Usage

```r
## S3 method for class 'lcModels'
max(x, name, ...)
```
Arguments

x  The lcModels object.
name The name of the internal metric.
... Additional arguments.

Value

The lcModel with the highest metric value

See Also

min.lcModels externalMetric

Examples

data(latrendData)
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")

model1 <- latrend(method, latrendData, nClusters = 1)
model2 <- latrend(method, latrendData, nClusters = 2)
model3 <- latrend(method, latrendData, nClusters = 3)

models <- lcModels(model1, model2, model3)

if (require("clusterCrit")) {
  max(models, "Dunn")
}

metric  Compute internal model metric(s)

Description

Compute one or more internal metrics for the given lcModel object.

Note that there are many metrics available, and there exists no metric that works best in all scenarios.

It is recommended to carefully consider which metric is most appropriate for your use case.

Recommended overview papers:

- Arbelaitz et al. (2013) provide an extensive overview validity indices for cluster algorithms.
- van der Nest et al. (2020) provide an overview of metrics for mixture models (GBTM, GMM); primarily likelihood-based or posterior probability-based metrics.
- Henson et al. (2007) provide an overview of likelihood-based metrics for mixture models.

Call getInternalMetricNames() to retrieve the names of the defined internal metrics.

See the Details section below for a list of supported metrics.
Usage

```r
## S4 method for signature 'lcModel'
metric(object, name = getOption("latrend.metric", c("WRSS", "APPA.mean")), ...)

## S4 method for signature 'list'
metric(object, name, drop = TRUE)

## S4 method for signature 'lcModels'
metric(object, name, drop = TRUE)
```

Arguments

- `object`: The lcModel, lcModels, or list of lcModel objects to compute the metrics for.
- `name`: The name(s) of the metric(s) to compute. If no names are given, the names specified in the `latrend.metric` option (WRSS, APPA, AIC, BIC) are used.
- `drop`: Whether to return a numeric vector instead of a data.frame in case of a single metric.

Details

List of currently supported metrics:

<table>
<thead>
<tr>
<th>Metric name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIC</td>
<td>Akaike information criterion. A goodness-of-fit estimator that adjusts for model complexity (i.e., the number of parameters). Only available for models that support the computation of the model log-likelihood through <code>logLik</code>.</td>
</tr>
<tr>
<td>APPA.mean</td>
<td>Mean of the average posterior probability of assignment (APPA) across clusters. A measure of the precision of the trajectory classification. A score of 1 indicates perfect classification.</td>
</tr>
<tr>
<td>APPA.min</td>
<td>Lowest APPA among the clusters</td>
</tr>
<tr>
<td>BIC</td>
<td>Bayesian information criterion. A goodness-of-fit estimator that corrects for the degrees of freedom (i.e., the number of parameters) and sample size. Only available for models that support the computation of the model log-likelihood through <code>logLik</code>.</td>
</tr>
<tr>
<td>CAIC</td>
<td>Consistent Akaike information criterion</td>
</tr>
<tr>
<td>CLC</td>
<td>Classification likelihood criterion</td>
</tr>
<tr>
<td>converged</td>
<td>Whether the model converged during estimation</td>
</tr>
<tr>
<td>deviance</td>
<td>The model deviance</td>
</tr>
<tr>
<td>Dunn</td>
<td>The Dunn index</td>
</tr>
<tr>
<td>entropy</td>
<td>Entropy of the posterior probabilities</td>
</tr>
<tr>
<td>estimationTime</td>
<td>The time needed for fitting the model</td>
</tr>
<tr>
<td>ED</td>
<td>Euclidean distance between the cluster trajectories and the assigned observed trajectories</td>
</tr>
<tr>
<td>ED.fit</td>
<td>Euclidean distance between the cluster trajectories and the assigned fitted trajectories</td>
</tr>
<tr>
<td>ICL.BIC</td>
<td>Integrated classification likelihood (ICL) approximated using the BIC</td>
</tr>
<tr>
<td>logLik</td>
<td>Model log-likelihood</td>
</tr>
<tr>
<td>MAE</td>
<td>Mean absolute error of the fitted trajectories (assigned to the most likely respective cluster) to the observed trajectories</td>
</tr>
<tr>
<td>Mahalanobis</td>
<td>Mahalanobis distance between the cluster trajectories and the assigned observed trajectories</td>
</tr>
<tr>
<td>MSE</td>
<td>Mean squared error of the fitted trajectories (assigned to the most likely respective cluster) to the observed trajectories</td>
</tr>
<tr>
<td>relativeEntropy,RE</td>
<td>A measure of the precision of the trajectory classification. A value of 1 indicates perfect classification.</td>
</tr>
<tr>
<td>RMSE</td>
<td>Root mean squared error of the fitted trajectories (assigned to the most likely respective cluster) to the observed trajectories</td>
</tr>
<tr>
<td>RSS</td>
<td>Residual sum of squares under most likely cluster allocation</td>
</tr>
<tr>
<td>scaledEntropy</td>
<td>See relativeEntropy</td>
</tr>
<tr>
<td>sigma</td>
<td>The residual standard deviation</td>
</tr>
</tbody>
</table>
ssBIC: Sample-size adjusted BIC
SED: Standardized Euclidean distance between the cluster trajectories and the assigned observed trajectories
SED.fit: The cluster-weighted standardized Euclidean distance between the cluster trajectories and the assigned fitted trajectories
WMAE: MAE weighted by cluster-assignment probability
WMSE: MSE weighted by cluster-assignment probability
WRMSE: RMSE weighted by cluster-assignment probability
WRSS: RSS weighted by cluster-assignment probability

Value
For `metric(lcModel)`: A named numeric vector with the computed model metrics.
For `metric(list)`: A data.frame with a metric per column.
For `metric(lcModels)`: A data.frame with a metric per column.

Implementation
See the documentation of the `defineInternalMetric()` function for details on how to define your own metrics.

References


See Also

`externalMetric`, `min.lcModels`, `max.lcModels`

Other metric functions: `defineExternalMetric()`, `defineInternalMetric()`, `externalMetric`, `lcModel`, `lcModel-method`, `getExternalMetricDefinition()`, `getExternalMetricNames()`, `getInternalMetricDefinition()`, `getInternalMetricNames()`

Examples

```r
data(latrendData)
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
model <- latrend(method, latrendData)
metric(model, "WMAE")

if (require("clusterCrit")) {
  metric(model, c("WMAE", "Dunn"))
}
```

```
min.lcModels Select the lcModel with the lowest metric value
```

Description

Select the lcModel with the lowest metric value

Usage

```r
## S3 method for class 'lcModels'
min(x, name, ...)
```
Arguments

x                 The lcModels object
name              The name of the internal metric.
...               Additional arguments.

Value

The lcModel with the lowest metric value

See Also

max.lcModels externalMetric

Examples

data(latrendData)
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")

model1 <- latrend(method, latrendData, nClusters = 1)
model2 <- latrend(method, latrendData, nClusters = 2)
model3 <- latrend(method, latrendData, nClusters = 3)

models <- lcModels(model1, model2, model3)

min(models, "WMAE")

model.data.lcModel

Extract the model data that was used for fitting

Description

Evaluates the data call in the environment that the model was trained in.

Usage

## S3 method for class 'lcModel'
model.data(object, ...)

Arguments

object       The lcModel object.
...          Additional arguments.

Value

The full data.frame that was used for fitting the lcModel.
model.frame.lcModel

See Also

model.frame.lcModel time.lcModel

Examples

data(latrendData)
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
model <- latrend(method, latrendData)
model.frame(model)

---

model.frame.lcModel  Extract model training data

Description

See stats::model.frame() for more details.

Usage

## S3 method for class 'lcModel'
model.frame(formula, ...)

Arguments

formula  The lcModel object.
...
Additional arguments.

Value

A data.frame containing the variables used by the model.

See Also

stats::model.frame model.data.lcModel

Other model-specific methods: clusterTrajectories(), coef.lcModel(), converged(), deviance.lcModel(),
df.residual.lcModel(), fitted.lcModel(), fittedTrajectories(), lcModel-class, logLik.lcModel(),
mobs.lcModel(), postprob(), predict.lcModel(), predictAssignments(), predictForCluster(),
predictPostprob(), residuals.lcModel(), sigma.lcModel(), time.lcModel()

Examples

data(latrendData)
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
model <- latrend(method, data = latrendData)
model.frame(model)
names,lcMethod-method  lcMethod argument names

Description

Extract the argument names or number of arguments from an lcMethod object.

Usage

```r
## S4 method for signature 'lcMethod'
length(x)

## S4 method for signature 'lcMethod'
names(x)
```

Arguments

- `x`  
  The lcMethod object.

Value

- The number of arguments, as integer.
  - A character vector of argument names.

See Also

Other lcMethod functions: `[[,lcMethod-method, as.data.frame.lcMethods(), as.data.frame.lcMethod(), as.lcMethods(), as.list.lcMethod(), evaluate.lcMethod(), formula.lcMethod(), lcMethod-class, update.lcMethod()`

Examples

```r
method <- lcMethodLMKM(Y ~ Time)
names(method)
length(method)
```

nClusters  Number of clusters

Description

Get the number of clusters estimated by the given lcModel object.

Usage

```r
nClusters(object)
```
Arguments
object The lcModel object.

Value
An integer with the number of clusters identified by the lcModel.

See Also
nIds nobs

Examples
data(latrendData)
method <- lcMethodRandom("Y", id = "Id", time = "Time", nClusters = 3)
model <- latrend(method, latrendData)
nClusters(model) # 3

table

Description
Get the number of trajectories (strata) that were used for fitting the given lcModel object. The number of trajectories is determined from the number of unique identifiers in the training data. In case the trajectory ids were supplied using a factor column, the number of trajectories is determined by the number of levels instead.

Usage
nIds(object)

Arguments
object The lcModel object.

Value
An integer with the number of trajectories on which the lcModel was fitted.

See Also
nobs nClusters

Examples
data(latrendData)
method <- lcMethodRandom("Y", id = "Id", time = "Time")
model <- latrend(method, latrendData)
nIds(model)
nobs.lcModel

Number of observations used for the lcModel fit

Description

Extracts the number of observations that contributed information towards fitting the cluster trajectories of the respective lcModel object. Therefore, only non-missing response observations count towards the number of observations.

Usage

## S3 method for class 'lcModel'
nobs(object, ...)

Arguments

object The lcModel object.
...

Additional arguments.

See Also

nIds nClusters

Other model-specific methods: clusterTrajectories(), coef.lcModel(), converged(), deviance.lcModel(), df.residual.lcModel(), fitted.lcModel(), fittedTrajectories(), lcModel-class, logLik.lcModel(), model.frame.lcModel(), postprob(), predict.lcModel(), predictAssignments(), predictForCluster(), predictPostprob(), residuals.lcModel(), sigma.lcModel(), time.lcModel()

Examples

data(latrendData)
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
model <- latrend(method, latrendData)
nobs(model)

OCC

Odds of correct classification (OCC)

Description

Computes the odds of correct classification (OCC) for each cluster. In other words, it computes the proportion of trajectories that can be expected to be correctly classified by the model for each cluster.

Usage

OCC(object)
Arguments

object The model, of type lcModel.

Details

An OCC of 1 indicates that the cluster assignment is no better than by random chance.

Value

The OCC per cluster, as a numeric vector of length nClusters(object). Empty clusters will output NA.

References


See Also

confusionMatrix APPA

___

OSA.adherence Biweekly Mean Treatment Adherence of OSA Patients over 1 Year

Description

Deprecated, renamed to PAP.adh.

Usage

OSA.adherence

Format

An object of class data.frame with 13000 rows and 5 columns.
Description

A simulated longitudinal dataset comprising 500 patients with obstructive sleep apnea (OSA) during their first year on CPAP therapy. The dataset contains the patient usage hours, averaged over 2-week periods.

The daily usage data underlying the downsampled dataset was simulated based on 7 different adherence patterns. The defined adherence patterns were inspired by the adherence patterns identified by Aloia et al. (2008), with slight adjustments.

The PAP.adh1y dataset is a subset of PAP.adh, comprising only patients who used therapy for at least 1 year. The subset does not contain the Non-users and Early drop-out groups.

Usage

PAP.adh

PAP.adh1y

Format

A data.frame comprising longitudinal data of 500 patients, each having 26 observations over a period of 1 year. Each row represents a patient observation interval (two weeks), with columns:

- **Patient** factor: The patient identifier, where each level represents a simulated patient.
- **Biweek** integer: Two-week interval index. Starts from 1.
- **MaxDay** integer: The last day used for the aggregation of the respective interval, integer
- **UsageHours** numeric: The mean hours of usage in the respective week. Greater than or equal to zero, and typically around 4-6 hours.
- **Group** factor: The reference group (i.e., adherence pattern) from which this patient was generated.

An object of class data.frame with 9880 rows and 5 columns.

Note

This dataset is only intended for demonstration purposes. While the data format will remain the same, the data content is subject to change in future versions.

Source

This dataset was generated based on the cluster-specific descriptive statistics table provided in Aloia et al. (2008), with some adjustments made in order to improve cluster separation for demonstration purposes.

Examples

```r
data(PAP.adh)

if (require("ggplot2")) {
  plotTrajectories(PAP.adh, id = "Patient", time = "Biweek", response = "UsageHours")

  # plot according to cluster ground truth
  plotTrajectories(
    PAP.adh,
    id = "Patient",
    time = "Biweek",
    response = "UsageHours",
    cluster = "Group"
  )
}
```

### Description

Plot a lcModel object. By default, this plots the cluster trajectories of the model, along with the training data.

### Usage

```r
## S4 method for signature 'lcModel'
plot(x, y, ...)
```

### Arguments

- `x` The lcModel object.
- `y` Not used.
- `...` Arguments passed on to `plotClusterTrajectories` object. The (cluster) trajectory data.

### Value

A ggplot object.

### See Also

- `plotClusterTrajectories` `plotFittedTrajectories` `plotTrajectories` `ggplot2::ggplot`
Examples

```r
data(latrendData)
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
model <- latrend(method, latrendData, nClusters = 3)

if (require("ggplot2")) {
  plot(model)
}
```

plot-lcModels-method  
Grid plot for a list of models

Description

Grid plot for a list of models

Usage

```r
## S4 method for signature 'lcModels'
plot(x, y, ..., subset, gridArgs = list())
```

Arguments

- `x`  
  The lcModels object.
- `y`  
  Not used.
- `...`  
  Additional parameters passed to the plot() call for each lcModel object.
- `subset`  
  Logical expression based on the lcModel method arguments, indicating which lcModel objects to keep.
- `gridArgs`  
  Named list of parameters passed to gridExtra::arrangeGrob.

plotClusterTrajectories

Plot cluster trajectories

Description

Plot cluster trajectories

Plot the cluster trajectories of a lcModel
Usage

## S4 method for signature 'data.frame'
plotClusterTrajectories(
  object,
  response,
  cluster = "Cluster",
  time = getOption("latrend.time"),
  center = meanNA,
  trajectories = c(FALSE, "sd", "se", "80pct", "90pct", "95pct", "range"),
  facet = !isFALSE(as.logical(trajectories[1])),
  id = getOption("latrend.id"),
  ...
)

## S4 method for signature 'lcModel'
plotClusterTrajectories(
  object,
  what = "mu",
  at = time(object),
  clusterLabels = NULL,
  trajectories = FALSE,
  trajAssignments = trajectoryAssignments(object),
  ...
)

Arguments

- **object**: The (cluster) trajectory data.
- **response**: The response variable name.
- **cluster**: The cluster assignment column
- **time**: The time variable name.
- **center**: A function for aggregating multiple points at the same point in time.
- **trajectories**: Whether to additionally plot the original trajectories (TRUE), or to show the expected interval (standard deviation, standard error, range, or percentile range) of the observations at the respective moment in time. Note that visualizing the expected intervals is currently only supported for time-aligned trajectories, as the interval is computed at each unique moment in time. By default (FALSE), no information on the underlying trajectories is shown.
- **facet**: Whether to facet by cluster. This is done by default when trajectories is enabled.
- **id**: Id column. Only needed when trajectories = TRUE.
- **...**: Arguments passed to `clusterTrajectories()`, or `ggplot2::geom_line()` for plotting the cluster trajectory lines.
what

The distributional parameter to predict. By default, the mean response ‘mu’ is predicted. The cluster membership predictions can be obtained by specifying what = 'mb'.

at

An optional vector of the times at which to compute the cluster trajectory predictions.

clusterLabels

Cluster display names. By default it’s the cluster name with its proportion enclosed in parentheses.

trajAssignments

The cluster assignments for the fitted trajectories. Only used when trajectories = TRUE and facet = TRUE. See trajectoryAssignments.

Value

A ggplot object.

See Also

clusterTrajectories plotFittedTrajectories plotTrajectories plot

Examples

data(latrendData)
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
model <- latrend(method, latrendData, nClusters = 3)

if (require("ggplot2")) {
  plotClusterTrajectories(model)

  # show assigned trajectories
  plotClusterTrajectories(model, trajectories = TRUE)

  # show 95th percentile observation interval
  plotClusterTrajectories(model, trajectories = "95pct")

  # show observation standard deviation
  plotClusterTrajectories(model, trajectories = "sd")

  # show observation standard error
  plotClusterTrajectories(model, trajectories = "se")

  # show observation range
  plotClusterTrajectories(model, trajectories = "range")
}
Plot fitted trajectories of a lcModel

Description
Plot fitted trajectories of a lcModel

Usage
## S4 method for signature 'lcModel'
plotFittedTrajectories(object, ...)

Arguments
- **object**: The lcModel object.
- **...**: Arguments passed on to trajectories
  - **id**: The identifier variable name.
  - **time**: The time variable name.
  - **response**: The response variable name.

See Also
- fittedTrajectories
- plotClusterTrajectories
- plotTrajectories
- plot

Examples
```r
data(latrendData)
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
model <- latrend(method, latrendData, nClusters = 3)

if (require("ggplot2")) {
  plotFittedTrajectories(model)
}
```

Plot one or more internal metrics for all lcModels

Description
Plot one or more internal metrics for all lcModels

Usage
plotMetric(models, name, by = "nClusters", subset, group = character())
plotTrajectories

Plot the data trajectories

Description

Plots the output of trajectories for the given object.

Usage

plotTrajectories(object, ...)

## S4 method for signature 'data.frame'
plotTrajectories(
  object,
  response,
plotTrajectories

```r

time = getOption("latrend.time"),
id = getOption("latrend.id"),
cluster = NULL,
facet = TRUE,
...
)

## S4 method for signature 'ANY'
plotTrajectories(object, ...)

## S4 method for signature 'lcModel'
plotTrajectories(object, ...)

Arguments

 object The data or model or extract the trajectories from.
...
 Arguments passed on to trajectories

 response Response variable character name or a call.
time The time variable name.
id The identifier variable name.
cluster Cluster variable name. If unspecified, trajectories are not grouped. Alternatively, cluster is a vector indicating cluster membership per id.
facet Whether to facet by cluster.

See Also

 trajectories plotFittedTrajectories plotClusterTrajectories

Examples

data(latrendData)

if (require("ggplot2")) {
  plotTrajectories(latrendData, response = "Y", id = "Id", time = "Time")
  plotTrajectories(
    latrendData,
    response = quote(exp(Y)),
    id = "Id",
    time = "Time"
  )
  plotTrajectories(
    latrendData,
    response = "Y",
    id = "Id",
    time = "Time",
    cluster = "Class"
  )
}
```
postFit

lcMethod fit process: logic for post-processing the fitted lcModel

Description

Note: this function should not be called directly, as it is part of the lcMethod fitting process. For fitting an lcMethod object to a dataset, see latrend().

The postFit() function of the lcMethod object defines how the lcModel object returned by fit() should be post-processed. This can be used, for example, to:

- Resolve label switching.
- Clean up the internal model representation.
- Correct estimation errors.
- Compute additional metrics.

By default, this method does not do anything. It merely returns the original lcModel object.

This is the last step in the lcMethod fitting procedure. The postFit method may be called again on fitted lcModel objects, allowing post-processing to be updated for existing models.

Usage

```r
## S4 method for signature 'lcMethod'
postFit(method, data, model, envir, verbose)
```
Arguments

- **method**: An object inheriting from `lcMethod` with all its arguments having been evaluated and finalized.
- **data**: A `data.frame` representing the transformed training data.
- **model**: The `lcModel` object returned by `fit()`.
- **envir**: The environment containing variables generated by `prepareData()` and `preFit()`.
- **verbose**: A `R.utils::Verbose` object indicating the level of verbosity.

Value

The updated `lcModel` object.

Implementation

The method is intended to be able to be called on previously fitted `lcModel` objects as well, allowing for potential bugfixes or additions to previously fitted models. Therefore, when implementing this method, ensure that you do not discard information from the model which would prevent the method from being run a second time on the object.

In this example, the `lcModelExample` class is assumed to be defined with a slot named "centers":

```r
setMethod("postFit", "lcMethodExample", function(method, data, model, envir, verbose) {
  # compute and store the cluster centers
  model@centers <- INTENSIVE_COMPUTATION
  return(model)
})
```

Fitting procedure

Each `lcMethod` subclass defines a type of methods in terms of a series of steps for estimating the method. These steps, as part of the fitting procedure, are executed by `latrend()` in the following order:

1. **compose()**: Evaluate and finalize the method argument values.
2. **validate()**: Check the validity of the method argument values in relation to the dataset.
3. **prepareData()**: Process the training data for fitting.
4. **preFit()**: Prepare environment for estimation, independent of training data.
5. **fit()**: Estimate the specified method on the training data, outputting an object inheriting from `lcModel`.
6. **postFit()**: Post-process the outputted `lcModel` object.

The result of the fitting procedure is an `lcModel` object that inherits from the `lcModel` class.
**Posterior probability per fitted trajectory**

### Description

Get the posterior probability matrix with element \((i, j)\) indicating the probability of trajectory \(i\) belonging to cluster \(j\).

### Usage

```r
## S4 method for signature 'lcModel'
postprob(object, ...)
```

### Arguments

- `object`: The `lcModel`.
- `...`: Additional arguments.

### Details

This method should be extended by `lcModel` implementations. The default implementation returns uniform probabilities for all observations.

### Value

A \(I\)-by-\(K\) matrix with \(I = nIds(object)\) and \(K = nClusters(object)\).

### Implementation

Classes extending `lcModel` should override this method.

```r
setMethod("postprob", "lcModelExt", function(object, ...) {
  # return trajectory-specific posterior probability matrix
})
```

### Troubleshooting

If you are getting errors about undefined model signatures when calling `postprob(model)`, check whether the `postprob()` function is still the one defined by the latrend package. It may have been overridden when attaching another package (e.g., lcmm). If you need to attach conflicting packages, load them first.

### See Also

- `trajectoryAssignments`
- `predictPostprob`
- `predictAssignments`

Other model-specific methods: `clusterTrajectories()`, `coef.lcModel()`, `converged()`, `deviance.lcModel()`, `df.residual.lcModel()`, `fitted.lcModel()`, `fittedTrajectories()`, `lcModel-class`, `logLik.lcModel()`, `model.frame.lcModel()`, `nobs.lcModel()`, `predict.lcModel()`, `predictAssignments()`, `predictForCluster()`, `predictPostprob()`, `residuals.lcModel()`, `sigma.lcModel()`, `time.lcModel()`
Examples

```r
data(latrendData)
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
model <- latrend(method, latrendData)
postprob(model)

if (rlang::is_installed("lcmm")) {
  gmmMethod = lcMethodLcmmGMM(
    fixed = Y ~ Time,
    mixture = ~ Time,
    id = "Id",
    time = "Time"
  )
  gmmModel <- latrend(gmmMethod, data = latrendData)
  postprob(gmmModel)
}
```

postprobFromAssignments

Create a posterior probability matrix from a vector of cluster assignments.

Description

For each trajectory, the probability of the assigned cluster is 1.

Usage

```r
postprobFromAssignments(assignments, k)
```

Arguments

- `assignments`: Integer vector indicating cluster assignment per trajectory
- `k`: The number of clusters.

predict.lcModel

lcModel predictions

Description

Predicts the expected trajectory observations at the given time for each cluster.

Usage

```r
## S3 method for class 'lcModel'
predict(object, newdata = NULL, what = "mu", ..., useCluster = NA)
```
predict.lcModel

Arguments

object The lcModel object.
newdata Optional data.frame for which to compute the model predictions. If omitted, the model training data is used. Cluster trajectory predictions are made when ids are not specified.
what The distributional parameter to predict. By default, the mean response 'mu' is predicted. The cluster membership predictions can be obtained by specifying what = 'mb'.
... Additional arguments.
useCluster Whether to use the "Cluster" column in the newdata argument for computing predictions conditional on the respective cluster. For useCluster = NA (the default), the feature is enabled if newdata contains the "Cluster" column.

Value

If newdata specifies the cluster membership; a data.frame of cluster-specific predictions. Otherwise, a list of data.frame of cluster-specific predictions is returned.

Implementation

Note: Subclasses of lcModel should preferably implement predictForCluster() instead of overriding predict.lcModel as that function is designed to be easier to implement because it is single-purpose.

The predict.lcModelExt function should be able to handle the case where newdata = NULL by returning the fitted values. After post-processing the non-NULL newdata input, the observation- and cluster-specific predictions can be computed. Lastly, the output logic is handled by the transformPredict() function. It converts the computed predictions (e.g., matrix or data.frame) to the appropriate output format.

predict.lcModelExt <- function(object, newdata = NULL, what = "mu", ...) {
  if (is.null(newdata)) {
    newdata = model.data(object)
    if (hasName(newdata, 'Cluster')) {
      # allowing the Cluster column to remain would break the fitted() output.
      newdata[['Cluster']] = NULL
    }
  }

  # compute cluster-specific predictions for the given newdata
  pred <- NEWDATA_COMPUTATIONS_HERE
  transformPredict(pred = pred, model = object, newdata = newdata)
}

See Also

predictForCluster stats::predict fitted.lcModel clusterTrajectories trajectories predictPostprob predictAssignments
predictAssignments

Other model-specific methods: `clusterTrajectories()`, `coef.lcModel()`, `converged()`, `deviance.lcModel()`, `df.residual.lcModel()`, `fitted.lcModel()`, `fittedTrajectories()`, `lcModel-class`, `logLik.lcModel()`, `model.frame.lcModel()`, `nobs.lcModel()`, `postprob()`, `predictAssignments()`, `predictForCluster()`, `predictPostprob()`, `residuals.lcModel()`, `sigma.lcModel()`, `time.lcModel()`

Examples

data(latrendData)
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
model <- latrend(method, latrendData)
predFitted <- predict(model) # same result as fitted(model)

# Cluster trajectory of cluster A
predCluster <- predict(model, newdata = data.frame(Cluster = "A", Time = time(model)))

# Prediction for id S1 given cluster A membership
predId <- predict(model, newdata = data.frame(Cluster = "A", Id = "S1", Time = time(model)))

# Prediction matrix for id S1 for all clusters
predIdAll <- predict(model, newdata = data.frame(Id = "S1", Time = time(model)))

predictAssignments

Predict the cluster assignments for new trajectories

Description
Computes the posterior probability based on the provided (observed) data.

Usage

```r
## S4 method for signature 'lcModel'
predictAssignments(object, newdata = NULL, strategy = which.max, ...)
```

Arguments

- `object` The lcModel object.
- `newdata` Optional data.frame for which to compute the model predictions. If omitted, the model training data is used. Cluster trajectory predictions are made when ids are not specified.
- `strategy` A function returning the cluster index based on the given vector of membership probabilities. By default (strategy = which.max), trajectories are assigned to the most likely cluster.
- `...` Additional arguments.

Details
The default implementation uses `predictPostprob` to determine the cluster membership.
predictForCluster

Value

A factor of length nrow(newdata) that indicates the assigned cluster per trajectory per observation.

See Also

predictPostprob predict.lcModel

Other model-specific methods: clusterTrajectories(), coef.lcModel(), converged(), deviance.lcModel(), df.residual.lcModel(), fitted.lcModel(), fittedTrajectories(), lcModel-class, logLik.lcModel(), model.frame.lcModel(), nobs.lcModel(), postprob(), predict.lcModel(), predictForCluster(), predictPostprob(), residuals.lcModel(), sigma.lcModel(), time.lcModel()

Examples

## Not run:
data(latrendData)
if (require("kml")) {
  model <- latrend(method = lcMethodKML("Y", id = "Id", time = "Time"), latrendData)
  predictAssignments(model, newdata = data.frame(Id = 999, Y = 0, Time = 0))
}

## End(Not run)

---

predictForCluster lcModel prediction conditional on a cluster

Description

Predicts the expected trajectory observations at the given time under the assumption that the trajectory belongs to the specified cluster.

The same result can be obtained by calling predict() with the newdata data.frame having a "Cluster" assignment column. The main purpose of this function is to make it easier to implement the prediction computations for custom lcModel classes.

Usage

```r
## S4 method for signature 'lcModel'
predictForCluster(object, newdata = NULL, cluster, ..., what = "mu")
```

Arguments

- **object**: The lcModel object.
- **newdata**: Optional data.frame for which to compute the model predictions. If omitted, the model training data is used. Cluster trajectory predictions are made when ids are not specified.
- **cluster**: The cluster name (as character) to predict for.
Additional arguments.

what

The distributional parameter to predict. By default, the mean response \('\text{mu}'\) is predicted. The cluster membership predictions can be obtained by specifying what = 'mb'.

Details

The default `predictForCluster()` method makes use of `predict.lcModel()`, and vice versa. For this to work, any extending `lcModel` classes, e.g., `lcModelExample`, should implement either `predictForCluster(lcModelExample)` or `predict.lcModelExample()`. When implementing new models, it is advisable to implement `predictForCluster` as the cluster-specific computation generally results in shorter and simpler code.

Value

A vector with the predictions per `newdata` observation, or a `data.frame` with the predictions and `newdata` alongside.

Implementation

Classes extending `lcModel` should override this method, unless `predict.lcModel()` is preferred.

```r
setMethod("predictForCluster", "lcModelExt",
  function(object, newdata = NULL, cluster, ..., what = "mu") {
    # return model predictions for the given data under the
    # assumption of the data belonging to the given cluster
  })
```

See Also

`predict.lcModel`

Other model-specific methods: `clusterTrajectories()`, `coef.lcModel()`, `converged()`, `deviance.lcModel()`, `df.residual.lcModel()`, `fitted.lcModel()`, `fittedTrajectories()`, `lcModel-class`, `logLik.lcModel()`, `model.frame.lcModel()`, `nobs.lcModel()`, `postprob()`, `predict.lcModel()`, `predictAssignments()`, `predictPostprob()`, `residuals.lcModel()`,

Examples

```r
data(latrendData)
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
model <- latrend(method, latrendData)

predictForCluster(model,
  newdata = data.frame(Time = c(0, 1)),
  cluster = "B"
)
```

# all fitted values under cluster B
`predictForCluster(model, cluster = "B")`
predictPostprob

lcModel posterior probability prediction

Description

Returns the observation-specific posterior probabilities for the given data. The default implementation returns a uniform probability matrix.

Usage

```r
## S4 method for signature 'lcModel'
predictPostprob(object, newdata = NULL, ...)
```

Arguments

- `object`: The `lcModel` to predict the posterior probabilities with.
- `newdata`: Optional data frame for which to compute the posterior probability. If omitted, the model training data is used.
- `...`: Additional arguments.

Value

A N-by-K matrix indicating the posterior probability per trajectory per measurement on each row, for each cluster (the columns). Here, \( N = \text{row}(newdata) \) and \( K = \text{nClusters}(object) \).

Implementation

Classes extending `lcModel` should override this method to enable posterior probability predictions for new data.

```r
setMethod("predictPostprob", "lcModelExt", function(object, newdata = NULL, ...){
  # return observation-specific posterior probability matrix
})
```

See Also

Other model-specific methods: `clusterTrajectories()`, `coef.lcModel()`, `converged()`, `deviance.lcModel()`, `df.residual.lcModel()`, `fitted.lcModel()`, `fittedTrajectories()`, `lcModel-class`, `logLik.lcModel()`, `model.frame.lcModel()`, `nobs.lcModel()`, `postprob()`, `predict.lcModel()`, `predictAssignments()`, `predictForCluster()`, `residuals.lcModel()`, `sigma.lcModel()`, `time.lcModel()`
Description

Note: this function should not be called directly, as it is part of the lcMethod fitting process. For fitting an lcMethod object to a dataset, see \texttt{latrend()}. The \texttt{preFit()} function of the lcMethod object performs preparatory work that is needed for fitting the method but should not be counted towards the method estimation time. The work is added to the provided environment, allowing the \texttt{fit()} function to make use of the prepared work.

Usage

## S4 method for signature 'lcMethod'
\texttt{preFit(method, data, envir, verbose)}

Arguments

- \texttt{method}: An object inheriting from \texttt{lcMethod} with all its arguments having been evaluated and finalized.
- \texttt{data}: A \texttt{data.frame} representing the transformed training data.
- \texttt{envir}: The environment containing additional data variables returned by \texttt{prepareData()}.
- \texttt{verbose}: A \texttt{R.utils::Verbose} object indicating the level of verbosity.

Value

The updated environment that will be passed to \texttt{fit()}.

Implementation

\begin{verbatim}
setMethod("preFit", "lcMethodExample", function(method, data, envir, verbose) {
  # update envir with additional computed work
  envir$x <- INTENSIVE_OPERATION
  return(envir)
})
\end{verbatim}

Fitting procedure

Each lcMethod subclass defines a type of methods in terms of a series of steps for estimating the method. These steps, as part of the fitting procedure, are executed by \texttt{latrend()} in the following order:

1. \texttt{compose()}: Evaluate and finalize the method argument values.
2. \texttt{validate()}: Check the validity of the method argument values in relation to the dataset.
3. \texttt{prepareData()}: Process the training data for fitting.
4. \texttt{preFit()}: Prepare environment for estimation, independent of training data.
5. **fit()**: Estimate the specified method on the training data, outputting an object inheriting from `lcModel`.

6. **postFit()**: Post-process the outputted `lcModel` object.

The result of the fitting procedure is an `lcModel` object that inherits from the `lcModel` class.

---

**prepareData**

### lcMethod fit process: logic for preparing the training data

#### Description

Note: this function should not be called directly, as it is part of the lcMethod fitting process. For fitting an lcMethod object to a dataset, see `latrend()`.

The `prepareData()` function of the lcMethod object processes the training data prior to fitting the method. Example uses:

- Transforming the data to another format, e.g., a matrix.
- Truncating the response variable.
- Computing derived covariates.
- Creating additional data objects.

The computed variables are stored in an environment which is passed to the `preFit()` function for further processing.

By default, this method does not do anything.

#### Usage

```r
## S4 method for signature 'lcMethod'
prepareData(method, data, verbose)
```

#### Arguments

- `method` An object inheriting from lcMethod with all its arguments having been evaluated and finalized.
- `data` A `data.frame` representing the transformed training data.
- `verbose` A `R.utils::Verbose` object indicating the level of verbosity.

#### Value

An environment with the prepared data variable(s) that will be passed to `preFit()`.
Implementation

A common use case for this method is when the internal method fitting procedure expects the data in a different format. In this example, the method converts the training data `data.frame` to a `matrix` of repeated and aligned trajectory measurements.

```r
setMethod("prepareData", "lcMethodExample", function(method, data, verbose) {
  envir = new.env()
  # transform the data to matrix
  envir$dataMat = tsmatrix(data,
    id = idColumn, time = timeColumn, response = valueColumn)
  return(envir)
})
```

Fitting procedure

Each `lcMethod` subclass defines a type of methods in terms of a series of steps for estimating the method. These steps, as part of the fitting procedure, are executed by `latrend()` in the following order:

1. `compose()`: Evaluate and finalize the method argument values.
2. `validate()`: Check the validity of the method argument values in relation to the dataset.
3. `prepareData()`: Process the training data for fitting.
4. `preFit()`: Prepare environment for estimation, independent of training data.
5. `fit()`: Estimate the specified method on the training data, outputting an object inheriting from `lcModel`.
6. `postFit()`: Post-process the outputted `lcModel` object.

The result of the fitting procedure is an `lcModel` object that inherits from the `lcModel` class.

---

**print.lcMethod**

*Print the arguments of an lcMethod object*

**Description**

Print the arguments of an `lcMethod` object

**Usage**

```r
## S3 method for class 'lcMethod'
print(x, ..., eval = FALSE, width = 40, envir = NULL)
```

**Arguments**

- `x` The `lcMethod` object.
- `...` Not used.
- `eval` Whether to print the evaluated argument values.
- `width` Maximum number of characters per argument.
- `envir` The environment in which to evaluate the arguments when `eval = TRUE`. 
print.lcModels  

**Print lcModels list concisely**

**Description**

Print lcModels list concisely

**Usage**

```r
## S3 method for class 'lcModels'
print(
x, 
..., 
summary = FALSE,
excludeShared = !getOption("latrend.printSharedModelArgs")
)
```

**Arguments**

- `x` The lcModels object.
- `...` Not used.
- `summary` Whether to print the complete summary per model. This may be slow for long lists!
- `excludeShared` Whether to exclude model arguments which are identical across all models.

**See Also**

Other lcModel list functions: `as.lcModels()`, `lcModels`, `subset.lcModels()`

---

qqPlot  

**Quantile-quantile plot**

**Description**

Plot the quantile-quantile (Q-Q) plot for the fitted lcModel object. This function is based on the `qqplotr` package.

**Usage**

```r
## S4 method for signature 'lcModel'
qqPlot(object, byCluster = FALSE, ...)
```
Arguments

- **object**: The `lcModel` object.
- **byCluster**: Whether to plot the Q-Q line per cluster.
- **...**: Additional arguments passed to `qqplotr::geom_qq_band()`, `qqplotr::stat_qq_line()`, and `qqplotr::stat_qq_point()`.

Value

A `ggplot` object.

See Also

- `residuals.lcModel`
- `metric`  
- `plotClusterTrajectories`

Examples

```r
data(latrendData)
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time", nClusters = 3)
model <- latrend(method, latrendData)

if (require("ggplot2") && require("qqplotr")) {
  qqPlot(model)
}
```

---

**residuals.lcModel**

*Extract lcModel residuals*

Description

Extract the residuals for a fitted `lcModel` object. By default, residuals are computed under the most likely cluster assignment for each trajectory.

Usage

```r
## S3 method for class 'lcModel'
residuals(object, ..., clusters = trajectoryAssignments(object))
```

Arguments

- **object**: The `lcModel` object.
- **...**: Additional arguments.
- **clusters**: Optional cluster assignments per id. If unspecified, a `matrix` is returned containing the cluster-specific predictions per column.

Value

A numeric vector of residuals for the cluster assignments specified by `clusters`. If the `clusters` argument is unspecified, a `matrix` of cluster-specific residuals per observations is returned.
See Also

fitted.lcModel trajectories

Other model-specific methods: clusterTrajectories(), coef.lcModel(), converged(), deviance.lcModel(), df.residual.lcModel(), fitted.lcModel(), fittedTrajectories(), lcModel-class, logLik.lcModel(), model.frame.lcModel(), nobs.lcModel(), postprob(), predict.lcModel(), predictAssignments(), predictForCluster(), predictPostprob(), sigma.lcModel(), time.lcModel()

Description

Extracts the response variable from the given object.

Usage

## S4 method for signature 'lcMethod'
responseVariable(object, ...)

## S4 method for signature 'lcModel'
responseVariable(object, ...)

Arguments

object The object to extract the response variable from.

... Additional arguments.

Details

If the lcMethod object specifies a formula argument, then the response is extracted from the response term of the formula.

Value

The response variable name as a character.

See Also

Other lcModel variables: idVariable(), timeVariable()

Examples

```r
method <- lcMethodLMKM(Y ~ Time)
responseVariable(method) # "Y"
data(latrendData)
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
model <- latrend(method, latrendData)
responseVariable(model) # "Y"
```
sigma.lcModel

Extract residual standard deviation from a lcModel

Description

Extracts or estimates the residual standard deviation. If sigma() is not defined for a model, it is estimated from the residual error vector.

Usage

## S3 method for class 'lcModel'
sigma(object, ...)

Arguments

object The lcModel object.
...
Additional arguments.

Value

A numeric indicating the residual standard deviation.

See Also

calc.lcModel metric

Other model-specific methods: clusterTrajectories().coef.lcModel().converged().deviance.lcModel().
df.residual.lcModel().fitted.lcModel().fittedTrajectories().lcModel-class.logLik.lcModel().
model.frame.lcModel().nobs.lcModel().postprob().predict.lcModel().predictAssignments().
predictForCluster().predictPostprob().residuals.lcModel().time.lcModel()

strip

Reduce the lcModel memory footprint for serialization

Description

Strip a lcModel of non-essential variables and environments in order to reduce the model size for serialization.

Usage

## S4 method for signature 'lcMethod'
strip(object, ..., classes = "formula")

## S4 method for signature 'ANY'
strip(object, ..., classes = "formula")

## S4 method for signature 'lcModel'
strip(object, ..., classes = "formula")
Arguments

- **object**: The lcModel object.
- **...**: Additional arguments.
- **classes**: The object classes for which to remove their assigned environment. By default, only environments from formula are removed.

Value

An lcModel object of the same type as the object argument.

Implementation

Classes extending lcModel can override this method to remove additional non-essentials.

```r
setMethod("strip", "lcModelExt", function(object, ..., classes = "formula") {
  object <- callNextMethod()
  # further process the object
  return(object)
})
```

Examples

```r
data(latrendData)
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
model <- latrend(method, latrendData)
newModel <- strip(model)
```

---

**subset.lcModels**

Subsetting a lcModels list based on method arguments

Description

Subsetting a lcModels list based on method arguments

Usage

```r
## S3 method for class 'lcModels'
subset(x, subset, drop = FALSE, ...)
```

Arguments

- **x**: The lcModels or list of lcModel to be subsettted.
- **subset**: Logical expression based on the lcModel method arguments, indicating which lcModel objects to keep.
- **drop**: Whether to return a lcModel object if the result is length 1.
- **...**: Not used.
Value

A \texttt{lcModels} list with the subset of \texttt{lcModel} objects.

See Also

Other \texttt{lcModel} list functions: \texttt{as.lcModels()}, \texttt{lcModels}, \texttt{print.lcModels()}

Examples

data(latrendData)
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")

model1 <- latrend(method, latrendData, nClusters = 1)
model2 <- latrend(method, latrendData, nClusters = 2)
model3 <- latrend(method, latrendData, nClusters = 3)

rngMethod <- lcMethodRandom("Y", id = "Id", time = "Time")
rngModel <- latrend(rngMethod, latrendData)

models <- lcModels(model1, model2, model3, rngModel)
subset(models, nClusters > 1 & .method == "lmkm")
**test.latrend**

*Test the implementation of an lcMethod and associated lcModel subclasses*

**Description**

Test a lcMethod subclass implementation and its resulting lcModel implementation.

**Usage**

```r
test.latrend(
  class = "lcMethodKML",
  instantiator = NULL,
  data = NULL,
  args = list(),
  tests = c("method", "basic", "fitted", "predict", "cluster-single", "cluster-three"),
  maxFails = 5L,
  errorOnFail = FALSE,
  clusterRecovery = c("warn", "ignore", "fail"),
  verbose = TRUE
)
```

**Arguments**

- `class`: The name of the lcMethod subclass to test. The class should inherit from lcMethod.
- `instantiator`: A function with signature `(id, time, response, ...)`, returning an object inheriting from the lcMethod specified by the class argument.
- `data`: An optional dataset comprising three highly distinct constant clusters that will be used for testing, represented by a data.frame. The data.frame must contain the columns "Id", "Time", "Value", "Cluster" of types character, numeric, numeric, and character, respectively. All trajectories should be of equal length and have observations at the same moments in time. Trajectory observations are assumed to be independent of time, i.e., all trajectories are constant. This enables tests to insert additional observations as needed by sampling from the available observations.
- `args`: Other arguments passed to the instantiator function.
- `tests`: A character vector indicating the type of tests to run, as defined in the *.Rraw files inside the /test/ folder.
- `maxFails`: The maximum number of allowed test condition failures before testing is ended prematurely.
- `errorOnFail`: Whether to throw the test errors as an error. This is always enabled while running package tests.
- `clusterRecovery`: Whether to test for correct recovery/identification of the original clusters in the test data. By default, a warning is outputted.
time.lcModel

verbose Whether the output testing results. This is always disabled while running package tests.

Note

This is an experimental function that is subject to large changes in the future. The default dataset used for testing is subject to change.

Examples

test.latrend("lcMethodRandom", clusterRecovery = "skip")

description

Extract the sampling times on which the lcModel was fitted.

Usage

## S3 method for class 'lcModel'
time(x, ...)

Arguments

x The lcModel object.

... Not used.

Value

A numeric vector of the unique times at which observations occur, in increasing order.

See Also

timeVariable model.data

Other model-specific methods: clusterTrajectories(), coef.lcModel(), converged(), deviance.lcModel(), df.residual.lcModel(), fitted.lcModel(), fittedTrajectories(), lcModel-class, logLik.lcModel(), model.frame.lcModel(), nobs.lcModel(), postprob(), predict.lcModel(), predictAssignments(), predictForCluster(), predictPostprob(), residuals.lcModel(), sigma.lcModel()
### timeVariable

**Extract the time variable**

**Description**

Extracts the time variable (i.e., column name) from the given object.

**Usage**

```r
## S4 method for signature 'lcMethod'
timeVariable(object, ...)
```

```r
## S4 method for signature 'lcModel'
timeVariable(object)
```

**Arguments**

- `object`  
  The object to extract the variable from.

- `...`  
  Additional arguments.

**Value**

The time variable name, as character.

**See Also**

Other lcModel variables: `idVariable()`, `responseVariable()`

**Examples**

```r
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
timeVariable(method) # "Time"
data(latrendData)
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
model <- latrend(method, latrendData)
idVariable(model) # "Id"
```

### trajectories

**Extract the trajectories**

**Description**

Transform or extract the trajectories from the given object to a standardized format. The standardized data format is for method estimation by `latrend`, and for plotting functions.
Usage

```r
trajectories(
  object,
  id = idVariable(object),
  time = timeVariable(object),
  response = responseVariable(object),
  ...
)
```

## S4 method for signature 'data.frame'
```r
trajectories(
  object,
  id = idVariable(object),
  time = timeVariable(object),
  response = responseVariable(object),
  ...
)
```

## S4 method for signature 'matrix'
```r
trajectories(
  object,
  id = idVariable(object),
  time = timeVariable(object),
  response = responseVariable(object),
  ...
)
```

## S4 method for signature 'call'
```r
trajectories(object, ..., envir)
```

## S4 method for signature 'lcModel'
```r
trajectories(
  object,
  id = idVariable(object),
  time = timeVariable(object),
  response = responseVariable(object),
  ...
)
```

Arguments

- `object` The data or model or extract the trajectories from.
- `id` The identifier variable name.
- `time` The time variable name.
- `response` The response variable name.
- `...` Additional arguments.
The environment used to evaluate the data object in (e.g., in case object is of type call).

Details
The generic function removes unused factor levels in the Id column, and any trajectories which are only comprised of NAs in the response.

Value
A data.frame with columns matching the id, time, and response name arguments.

See Also
plotTrajectories latrend

### S4 method for signature 'matrix'
trajectoryAssignments(
    object,
    strategy = which.max,
    clusterNames = colnames(object),
    ...
)

### S4 method for signature 'lcModel'
trajectoryAssignments(object, strategy = which.max, ...)

Get the cluster membership of each trajectory

Description
Classify the fitted trajectories based on the posterior probabilities computed by postprob(), according to a given classification strategy.

By default, trajectories are assigned based on the highest posterior probability using which.max(). In cases where identical probabilities are expected between clusters, it is preferable to use which.is.max instead, as this function breaks ties at random. Another strategy to consider is the function which.weight(), which enables weighted sampling of cluster assignments based on the trajectory-specific probabilities.
transformFitted

Arguments

- **object**: The object to obtain the cluster assignments from.
- **strategy**: A function returning the cluster index based on the given vector of membership probabilities. By default, ids are assigned to the cluster with the highest probability.
- **clusterNames**: Optional character vector with the cluster names. If `clusterNames = NULL`, `make.clusterNames()` is used.
- **...**: Any additional arguments passed to the strategy function.

Details

In case `object` is a matrix: the posterior probability matrix, with the kth column containing the observation- or trajectory-specific probability for cluster k.

Value

A factor indicating the cluster membership for each trajectory.

See Also

`postprob`, `clusterSizes`, `predictAssignments`

Examples

```r
data(latrendData)
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
model <- latrend(method, latrendData)
trajectoryAssignments(model)
# assign trajectories at random using weighted sampling
trajectoryAssignments(model, strategy = which.weight)
```

Description

A helper function for implementing the `fitted.lcModel()` method as part of your own `lcModel` class, ensuring the correct output type and format (see the Value section). Note that this function has no use outside of implementing `fitted.lcModel`.

The function makes it easier to implement `fitted.lcModel` based on existing implementations that may output their results in different data formats. Furthermore, the function checks whether the input data is valid.

The prediction ordering depends on the ordering of the data observations that was used for fitting the `lcModel`.

By default, `transformFitted()` accepts one of the following inputs:
data.frame A data.frame in long format providing a cluster-specific prediction for each observation per row, with column names "Fit" and "Cluster". This data.frame therefore has \text{nobs}(\text{object}) \times \text{nClusters}(\text{object}) rows.

matrix An N-by-K matrix where each row provides the cluster-specific predictions for the respective observation. Here, N = nrow(\text{model.data}(\text{object})) and K = nClusters(\text{object}).

list A list of cluster-specific prediction vectors. Each prediction vector should be of length nrow(\text{model.data}(\text{object})). The overall (named) list of cluster-specific prediction vectors is of length nClusters(\text{object}).

Users can implement support for other prediction formats by defining the \text{transformFitted} method with other signatures.

Usage

\text{transformFitted}(\text{pred}, \text{model}, \text{clusters})

## S4 method for signature '\text{\textbackslash NULL}',\text{lcModel}'
\text{transformFitted}(\text{pred}, \text{model}, \text{clusters} = \text{\textbackslash NULL})

## S4 method for signature 'matrix,\text{lcModel}'
\text{transformFitted}(\text{pred}, \text{model}, \text{clusters} = \text{\textbackslash NULL})

## S4 method for signature 'list,\text{lcModel}'
\text{transformFitted}(\text{pred}, \text{model}, \text{clusters} = \text{\textbackslash NULL})

## S4 method for signature '\text{data.frame},\text{lcModel}'
\text{transformFitted}(\text{pred}, \text{model}, \text{clusters} = \text{\textbackslash NULL})

Arguments

\begin{itemize}
\item \text{pred} The cluster-specific predictions for each observation
\item \text{model} The \text{lcModel} by which the prediction was made.
\item \text{clusters} The trajectory cluster assignment per observation. Optional.
\end{itemize}

Value

If the \text{clusters} argument was specified, a vector of fitted values conditional on the given cluster assignment. Else, a matrix with the fitted values per cluster per column.

Example implementation

A typical implementation of \text{fitted.lcModel()} for your own \text{lcModel} class would have the following format:

\begin{verbatim}
fitted.lcModelExample <- function(object,
               clusters = trajectoryAssignments(object)) {
    # computations of the fitted values per cluster here
    predictionMatrix <- CODE_HERE
    \text{transformFitted}(\text{pred} = \text{predictionMatrix}, \text{model} = \text{object}, \text{clusters} = \text{clusters})
\end{verbatim}
transformPredict

}  

For a complete and runnable example, see the custom models vignette accessible via vignette("custom",  
package = "latrend").

---

**transformPredict**  
*Helper function for custom lcModel classes implementing predict.lcModel()*

### Description

A helper function for implementing the `predict.lcModel()` method as part of your own `lcModel`  
class, ensuring the correct output type and format (see the Value section). Note that this function  
has no use outside of ensuring valid output for `predict.lcModel`. For implementing `lcModel`  
predictions from scratch, it is advisable to implement `predictForCluster` instead of  
`predict.lcModel`.

The prediction ordering corresponds to the observation ordering of the `newdata` argument.

By default, `transformPredict()` accepts one of the following inputs:

- **data.frame**  
  A data.frame in long format providing a cluster-specific prediction for each observation per row,  
  with column names "Fit" and "Cluster". This data.frame therefore has  
  `nrow(model.data(object)) * nClusters(object)` rows.

- **matrix**  
  An N-by-K matrix where each row provides the cluster-specific predictions for the respective  
  observations in `newdata`. Here, N = `nrow(newdata)` and K = `nClusters(object)`.

- **vector**  
  A vector of length `nrow(newdata)` with predictions corresponding to the rows of `newdata`.

Users can implement support for other prediction formats by defining the `transformPredict()`  
method with other signatures.

### Usage

```r
transformPredict(pred, model, newdata)
```

## S4 method for signature 'NULL',lcModel
```r
transformPredict(pred, model, newdata)
```

## S4 method for signature 'vector',lcModel
```r
transformPredict(pred, model, newdata)
```

## S4 method for signature 'matrix',lcModel
```r
transformPredict(pred, model, newdata)
```

## S4 method for signature 'data.frame',lcModel
```r
transformPredict(pred, model, newdata)
```
Arguments

pred   The (per-cluster) predictions for newdata.
model The lcModel for which the prediction was made.
newdata A data.frame containing the input data to predict for.

Value

A data.frame with the predictions, or a list of cluster-specific prediction data.frames.

Example implementation

In case we have a custom lcModel class based on an existing internal model representation with a predict() function, we can use transformPredict() to easily transform the internal model predictions to the right format. A common output is a matrix with the cluster-specific predictions.

```r
predict.lcModelExample <- function(object, newdata) {
  predictionMatrix <- predict(object@model, newdata)
  transformPredict(
    pred = predictionMatrix,
    model = object,
    newdata = newdata
  )
}
```

However, for ease of implementation it is generally advisable to implement predictForCluster instead of predict.lcModel.

For a complete and runnable example, see the custom models vignette accessible via vignette("custom", package = "latrend").

See Also

predictForCluster, predict.lcModel

---

**tsframe**

*Convert a multiple time series matrix to a data.frame*

**Description**

Convert a multiple time series matrix to a data.frame
tsframe

Usage

tsframe(
  data,
  response,
  id = getOption("latrend.id"),
  time = getOption("latrend.time"),
  ids = rownames(data),
  times = colnames(data),
  as.data.table = FALSE
)

meltRepeatedMeasures(
  data,
  response,
  id = getOption("latrend.id"),
  time = getOption("latrend.time"),
  ids = rownames(data),
  times = colnames(data),
  as.data.table = FALSE
)

Arguments

data The matrix containing a trajectory on each row.
response The response column name.
id The id column name.
time The time column name.
ids A vector specifying the id names. Should match the number of rows of data.
times A numeric vector specifying the times of the measurements. Should match the number of columns of data.

as.data.table Whether to return the result as a data.table, or a data.frame otherwise.

Value

A data.table or data.frame containing the repeated measures.

Note

The meltRepeatedMeasures() function is deprecated and will be removed in a future version, please use tsframe() instead.

See Also

 tsmatrix
tsmatrix  
*Convert a longitudinal data.frame to a matrix*

**Description**

Converts a longitudinal data.frame comprising trajectories with an equal number of observations, measured at identical moments in time, to a matrix. Each row of the matrix represents a trajectory.

**Usage**

```r
tsmatrix(
  data,
  response,
  id = getOption("latrend.id"),
  time = getOption("latrend.time"),
  fill = NA
)

dcastRepeatedMeasures(
  data,
  response,
  id = getOption("latrend.id"),
  time = getOption("latrend.time"),
  fill = NA
)
```

**Arguments**

- `data`  
  The matrix containing a trajectory on each row.
- `response`  
  The response column name.
- `id`  
  The id column name.
- `time`  
  The time column name.
- `fill`  
  A scalar value. If FALSE, an error is thrown when time series observations are missing in the data frame. Otherwise, the value used for representing missing observations.

**Value**

A matrix with a trajectory per row.

**Note**

The `dcastRepeatedMeasures()` function is deprecated and will be removed in a future version. Please use `tsmatrix()` instead.
update.lcMethod

See Also

tsframe

update.lcMethod  Update a method specification

Description

Update a method specification

Usage

## S3 method for class 'lcMethod'
update(object, ..., .eval = FALSE, .remove = character(), envir = NULL)

Arguments

object The lcMethod object.
...
The new or updated method argument values.
.eval Whether to assign the evaluated argument values to the method. By default
(FALSE), the argument expression is preserved.
.remove Names of arguments that should be removed.
envir The environment in which to evaluate the arguments. If NULL, the environment
associated with the object is used. If not available, the parent.frame() is used.

Details

Updates or adds arguments to a lcMethod object. The inputs are evaluated in order to determine
the presence of formula objects, which are updated accordingly.

Value

The new lcMethod object with the additional or updated arguments.

See Also

Other lcMethod functions: [,lcMethod-method, as.data.frame.lcMethods(), as.data.frame.lcMethod(),
as.lcMethods(), as.list.lcMethod(), evaluate.lcMethod(), formula.lcMethod(), lcMethod-class,
names,lcMethod-method
Examples

```r
method <- lcMethodLMKM(Y ~ 1, nClusters = 2)
method2 <- update(method, formula = ~ . + Time)

method3 <- update(method2, nClusters = 3)

k <- 2
method4 <- update(method, nClusters = k) # nClusters: k

method5 <- update(method, nClusters = k, .eval = TRUE) # nClusters: 2
```

update.lcModel

Update a lcModel

Description

Fit a new model with modified arguments from the current model.

Usage

```r
## S3 method for class 'lcModel'
update(object, ...)
```

Arguments

- `object` The lcModel object.
- `...` Arguments passed on to `latrend`
- `method` An lcMethod object specifying the longitudinal cluster method to apply, or the name (as character) of an lcMethod subclass. See `lcMethod` for details.
- `data` The data.frame to which to apply the method. Inputs supported by `trajectories()` can also be used.
- `envir` The environment in which to evaluate the method arguments (by `compose()`). This environment is also used to evaluate the data argument if it is of type `call`.
- `verbose` The level of verbosity. Either an object of class `Verbose` (see `R.utils::Verbose` for details), a logical indicating whether to show basic computation information, a numeric indicating the verbosity level (see `Verbose`), or one of `c('info', 'fine', 'finest')`.

Value

The refitted lcModel object, of the same type as the object argument.

See Also

`latrend` `getCall`
validate

Examples
data(latrendData)
method <- lcMethodLMM(Y ~ Time, id = "Id", time = "Time")
model2 <- latrend(method, latrendData, nClusters = 2)

# fit for a different number of clusters
model3 <- update(model2, nClusters = 3)

---

validate

**lcMethod fit process: method argument validation logic**

Description

Note: this function should not be called directly, as it is part of the lcMethod fitting process. For fitting an lcMethod object to a dataset, see `latrend()`.

The `validate()` function of the lcMethod object validates the method with respect to the training data. This enables a method to verify, for example:

- whether the formula covariates are present.
- whether the argument combination settings are valid.
- whether the data is suitable for training.

By default, the `validate()` function checks whether the id, time, and response variables are present as columns in the training data.

Usage

```r
## S4 method for signature 'lcMethod'
validate(method, data, envir = NULL, ...)
```

Arguments

- `method`: An object inheriting from lcMethod with all its arguments having been evaluated and finalized.
- `data`: A data.frame representing the transformed training data.
- `envir`: The environment in which the lcMethod should be evaluated.
- `...`: Not used.

Value

Either `TRUE` if all validation checks passed, or a character containing a description of the failed validation checks.
Implementation

An example implementation checking for the existence of specific arguments and type:

```r
library(assertthat)
setMethod("validate", "lcMethodExample", function(method, data, envir = NULL, ...) {
  validate_that(
    hasName(method, "myArgument"),
    hasName(method, "anotherArgument"),
    is.numeric(method$myArgument)
  )
})
```

Fitting procedure

Each lcMethod subclass defines a type of methods in terms of a series of steps for estimating the method. These steps, as part of the fitting procedure, are executed by `latrend()` in the following order:

1. `compose()`: Evaluate and finalize the method argument values.
2. `validate()`: Check the validity of the method argument values in relation to the dataset.
3. `prepareData()`: Process the training data for fitting.
4. `prefit()`: Prepare environment for estimation, independent of training data.
5. `fit()`: Estimate the specified method on the training data, outputting an object inheriting from `lcModel`.
6. `postfit()`: Post-process the outputted `lcModel` object.

The result of the fitting procedure is an `lcModel` object that inherits from the `lcModel` class.

See Also

`assertthat::validate_that`

---

`which.weight` | *Sample an index of a vector weighted by the elements*

**Description**

Returns a random index, weighted by the element magnitudes. This function is intended to be used as an optional strategy for `trajectoryAssignments`, resulting in randomly sampled cluster membership.

**Usage**

`which.weight(x)`
Arguments

  x  A positive numeric vector.

Value

  An integer giving the index of the sampled element.

Examples

  x = c(.01, .69, .3)
  which.weight(x) #1, 2, or 3

[[,lcMethod-method Retrieve and evaluate a lcMethod argument by name

Description

  Retrieve and evaluate a lcMethod argument by name

Usage

  ## S4 method for signature 'lcMethod'
  x$name

  ## S4 method for signature 'lcMethod'
  x[[i, eval = TRUE, envir = NULL]]

Arguments

  x  The lcMethod object.
  name  The argument name, as character.
  i  Name or index of the argument to retrieve.
  eval  Whether to evaluate the call argument (enabled by default).
  envir  The environment in which to evaluate the argument. This argument is only applicable when eval = TRUE.

Value

  The argument call or evaluation result.

See Also

  Other lcMethod functions: as.data.frame.lcMethods(), as.data.frame.lcMethod(), as.lcMethods(), as.list.lcMethod(), evaluate.lcMethod(), formula.lcMethod(), lcMethod-class, names, lcMethod-method, update.lcMethod()
Examples

```r
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time", nClusters = 3)
m$\text{nClusters} # 3
m = lcMethodLMKM(Y ~ Time, id = "Id", time = "Time", nClusters = 5)
m[["nClusters"]][[5]] # 5

k = 2
m = lcMethodLMKM(Y ~ Time, id = "Id", time = "Time", nClusters = k)
m[["nClusters", eval=FALSE]] # k
```
Index

* datasets
  latrendData, 53
  OSA.adherence, 99
  PAP.adh, 100

* lcMethod functions
  [[,lcMethod-method, 141
  as.data.frame.lcMethod, 8
  as.data.frame.lcMethods, 9
  as.lcMethods, 10
  as.list.lcMethod, 11
  evaluate.lcMethod, 27
  formula.lcMethod, 35
  lcMethod-class, 56
  names,lcMethod-method, 96
  update.lcMethod, 137

* lcMethod implementations
  getArgumentDefaults, 38
  getArgumentExclusions, 39
  lcMethod-class, 56
  lcMethodAkmedoids, 58
  lcMethodCrimCV, 59
  lcMethodDtwc1ust, 60
  lcMethodFeature, 61
  lcMethodFunction, 65
  lcMethodFunFEM, 66
  lcMethodGCKM, 67
  lcMethodKML, 69
  lcMethodLcmmGBTM, 70
  lcMethodLcmmGMM, 71
  lcMethodLMKM, 73
  lcMethodMclustLLPA, 74
  lcMethodMixAK_GLMM, 75
  lcMethodMixtoolsGMM, 76
  lcMethodMixtoolsNPRM, 77
  lcMethodRandom, 80
  lcMethodStratify, 82

* lcMethod package interfaces
  lcMethodFlexmix, 63
  lcMethodFlexmixGBTM, 64

* lcModel list functions
  as.lcModels, 11
  lcModels, 87
  print.lcModels, 120
  subset.lcModels, 124

* lcModel variables
  idVariable, 45
  responseVariable, 122
  timeVariable, 128

* longitudinal cluster fit functions
  latrend, 46
  latrendBatch, 49
  latrendBoot, 50
  latrendCV, 51
  latrendRep, 54

* metric functions
  defineExternalMetric, 23
  defineInternalMetric, 24
  externalMetric,lcModel,lcModel-method, 28
  getExternalMetricDefinition, 40
  getExternalMetricNames, 40
  getInternalMetricDefinition, 41
  getInternalMetricNames, 41
  metric, 90

* model-specific methods
  clusterTrajectories, 16
  coef.lcModel, 17
  converged, 20
  deviance.lcModel, 25
  df.residual.lcModel, 25
  fitted.lcModel, 32
  fittedTrajectories, 34
  lcModel-class, 84
  logLik.lcModel, 88
  model.frame.lcModel, 95
  nobs.lcModel, 98
  postprob, 110
  predict.lcModel, 111
predictAssignments, 113
predictForCluster, 114
predictPostprob, 116
residuals.lcModel, 121
sigma.lcModel, 123
time.lcModel, 127

* validation methods
createTestDataFold, 21
createTestDataFolds, 22
createTrainDataFolds, 22
latrendBoot, 50
latrendCV, 51
[[,lcMethod-method, 141
$,lcMethod-method ([[,lcMethod-method), 141
_.PACKAGE (latrend-package), 5

akmedoids::akclustr, 58
APPA, 7, 19, 99
APPA(), 91
approx, 56
as.data.frame.lcMethod, 8, 9, 10, 12, 27, 35, 57, 96, 137, 141
as.data.frame.lcMethods, 8, 9, 10, 12, 27, 35, 57, 96, 137, 141
as.data.frame.lcModels, 9
as.lcMethods, 8, 9, 10, 12, 27, 35, 57, 96, 137, 141
as.lcModels, 11, 87, 120, 125
as.list.lcMethod, 8–10, 11, 27, 35, 57, 96, 137, 141
assertthat::validate_that, 140
atomic, 8
base::difftime, 26
clusterCrit::extCriteria(), 29
clusterNames, 12
clusterNames<-, 13
clusterProportions, 14, 15, 19
clusterProportions,lcModel-method (clusterProportions), 14
clusterSizes, 14, 15, 131
clusterTrajectories, 7, 16, 17, 20, 25, 26, 33, 34, 85, 89, 95, 98, 104, 110, 112–116, 122, 123, 127
clusterTrajectories(), 103
clusterTrajectories,lcModel-method (clusterTrajectories), 16
coe.lcModel, 16, 17, 20, 25, 26, 33, 34, 85, 89, 95, 98, 110, 113–116, 122, 123, 127
compose, 18, 27
compose(), 18, 32, 47, 51, 52, 54, 57, 109, 117, 119, 138, 140
compose,lcModel-method (compose), 18
confusionMatrix, 8, 19, 99
converged, 16, 17, 20, 25, 26, 33, 34, 85, 89, 95, 98, 110, 113–116, 122, 123, 127
converged(), 91
converged,lcModel-method (converged), 20
createTestDataFold, 21, 22, 23, 51, 52
createTestDataFolds, 21, 22, 23, 51, 52
createTrainDataFolds, 21, 22, 23, 51, 52
crimCV::crimCV, 59
dcastRepeatedMeasures (tsmatrix), 136
defineExternalMetric, 23, 24, 31, 40, 41, 93
defineExternalMetric(), 30
defineInternalMetric, 23, 24, 31, 40, 41, 93
defineInternalMetric(), 92
deviance.lcModel, 16, 17, 20, 25, 26, 33, 34, 85, 89, 95, 98, 110, 113–116, 122, 123, 127
df.residual.lcModel, 16, 17, 20, 25, 26, 33, 34, 85, 89, 95, 98, 110, 113–116, 122, 123, 127
dist, 30
dtwclust::tsclust, 61
diffTime, 26
diffTime(), 91
diffTime,lcModel-method (diffTime), 26
diffTime,lcModels-method (diffTime), 26
diffTime,list-method (diffTime), 26
evaluate.lcMethod, 8–10, 12, 19, 27, 35, 57, 96, 137, 141
evaluationMetric, 90, 93, 94
evaluationMetric (evaluationMetric,lcModel,lcModel-method), 28
evaluationMetric,lcModel,lcModel-method, 28
externalMetric, lcModels, character-method

getArgumentExclusions, lcMethod-method

(getArgumentExclusions, lcModel-method), 39

call, 138

generateLongData, 36, 53

getArgumentDefaults, 38, 40, 57, 59–61, 63, 66–69, 71, 72, 74–78, 81, 83

getArgumentDefault(), 57

getArgumentDefault, lcMethod-method

(getArgumentDefault, lcMethod-method), 38

getArgumentExclusions, 39, 39, 40, 57, 59–61, 63, 66–69, 71, 72, 74–78, 81, 83

fit, 31

fit(), 19, 26, 32, 57, 108, 109, 117–119, 140

fit, lcMethod-method (fit), 31

fitted.lcApproxModel

(lcApproxModel-class), 55

fitted.lcModel, 7, 16, 17, 20, 25, 26, 32, 34, 85, 89, 95, 98, 110, 112–116, 122, 123, 127

fitted.lcModel(), 131, 132

fittedTrajectories, 7, 16, 17, 20, 25, 26, 33, 34, 85, 89, 95, 98, 105, 110, 113–116, 122, 123, 127

fittedTrajectories, lcModel-method (fittedTrajectories), 34

flexmix:::flexmix, 64, 65

flexmix:::FLXMRglm, 64, 65

foreach, 47

formals, 38

formula.lcMethod, 8–10, 12, 27, 35, 57, 96, 137, 141

formula.lcModel, 36

funFEM:::FunFEM, 67

generateLongData, 36, 53

getArgumentDefaults, 38, 40, 57, 59–61, 63, 66–69, 71, 72, 74–78, 81, 83

getArgumentDefault(), 57

getArgumentDefault, lcMethod-method

(getArgumentDefault, lcMethod-method), 38

getArgumentExclusions, 39, 39, 40, 57, 59–61, 63, 66–69, 71, 72, 74–78, 81, 83

getExternalMetricNames, 24, 31, 40, 41

getExternalMetricNames(), 28

getInternalMetricDefinition, 24, 31, 40, 41, 93

getInternalMetricNames, 24, 31, 40, 41, 93

ggridExtra::arrangeGrob, 102

igraph::compare(), 29

igraph::split_join_distance(), 29

initialize, lcMethod-method, 46

kml::kml, 69

kml::parALGO, 69

latrend, 7, 46, 50–52, 54, 55, 128, 130, 138

latrend(), 6, 18, 31, 32, 42, 57, 108, 109, 117–119, 139, 140
latrend-package, 5
latrend-parallel, 7, 47, 49, 51, 52, 54
latrendBatch, 7, 47, 48, 49, 51, 52, 55
latrendBatch(), 6
latrendBoot, 7, 21–23, 47, 48, 50, 50, 52, 55
latrendCV, 7, 21–23, 47, 48, 50, 51, 51, 55
latrendData, 6, 53
latrendRep, 7, 47, 48, 50–52, 54
lcApproxModel (lcApproxModel-class), 55
lcApproxModel-class, 55
lcMethod, 7, 39, 40, 47, 51, 52, 54, 138
lcMethod (lcMethod-class), 56
lcMethod-class, 56, 84
lcMethodAkaeodids, 39, 40, 57, 58, 60, 61, 63, 66–69, 71, 72, 74–78, 81, 83
lcMethodCrimCV, 39, 40, 57, 59, 59, 61, 63, 66–69, 71, 72, 74–78, 81, 83
lcMethodDtwclust, 39, 40, 57, 59, 60, 60, 63, 66–69, 71, 72, 74–78, 81, 83
lcMethodDtwclust(), 56
lcMethodFeature, 39, 40, 57, 59–61, 61, 66–69, 71, 72, 74–78, 81, 83
lcMethodFlexmix, 63, 65
lcMethodFlexmixGBTM, 64, 64
lcMethodFunction, 39, 40, 57, 59–61, 63, 65, 67–69, 71, 72, 74–78, 81, 83
lcMethodFunFEM, 39, 40, 57, 59–61, 63, 66, 66, 68, 69, 71, 72, 74–78, 81, 83
lcMethodGCKM, 39, 40, 57, 59–61, 63, 66, 67, 67, 69, 71, 72, 74–78, 81, 83
lcMethodKML, 39, 40, 57, 59–61, 63, 66–68, 69, 71, 72, 74–78, 81, 83
lcMethodKML(), 56
lcMethodLcmmGBTM, 39, 40, 57, 59–61, 63, 66–69, 70, 72, 74–78, 81, 83
lcMethodLcmmGBTM(), 56
lcMethodLcmmGMM, 39, 40, 57, 59–61, 63, 66–69, 71, 71, 74–78, 81, 83
lcMethodLMKM, 39, 40, 57, 59–61, 63, 66–69, 71, 72, 73, 75–78, 81, 83
lcMethodLMKM(), 62
lcMethodMclustLPA, 39, 40, 57, 59–61, 63, 66–69, 71, 72, 74, 76–78, 81, 83
lcMethodMixAK_GLCMM, 39, 40, 57, 59–61, 63, 66–69, 71, 72, 74, 75, 75, 77, 78, 81, 83
lcMethodMixtoolsGMM, 39, 40, 57, 59–61, 63, 66–69, 71, 72, 74–76, 76, 78, 81, 83
lcMethodMixtoolsNPRM, 39, 40, 57, 59–61, 63, 66–69, 71, 72, 74–77, 77, 81, 83
lcMethodMixTVEM, 79
lcMethodRandom, 39, 40, 57, 59–61, 63, 66–69, 71, 72, 74–78, 80, 83
lcMethods, 7, 81
lcMethods(), 6
lcMethodStratify, 39, 40, 57, 59–61, 63, 66–69, 71, 72, 74–78, 81, 82
lcmc::hmae, 70, 72
lcModel, 7, 19, 32, 57, 109, 118, 119, 140
lcModel-class, 84
lcModelPartition, 85
lcModels, 11, 87, 120, 125
lcModels-class (lcModels), 87
lcModelWeightedPartition, 88
length (lcMethod-method)
(names, lcMethod-method), 96
lme4::lmer, 68, 77
logLik, 91
logLik(lcModel), 16, 17, 20, 25, 26, 33, 34, 85, 88, 95, 98, 110, 113–116, 122, 123, 127
make.clusteringNames(), 131
max (lcModels), 89, 93, 94
mclust::Mclust, 74
mclustcomp::mclustcomp(), 28–30
meltRepeatedMeasures (tsframe), 134
method fitting procedure, 46
methods::new(), 56
metric, 24, 25, 31, 40, 41, 89, 90, 121, 123
metric, lcModel-method (metric), 90
metric, lcModels-method (metric), 90
metric, list-method (metric), 90
min (lcModels), 90, 93, 93
mixAK::GLMM_MCMC, 75, 76
mixtools::npEM, 78
mixtools::regmixEM.mixed, 77
model.data, 127
model.data.lcModel, 86, 94, 95
model.frame.lcModel, 16, 17, 20, 25, 26, 33, 34, 85, 89, 95, 98, 110, 113–116, 122, 123, 127
names, lcMethod-method, 96
nClusters, 96, 97, 98
nIds, 97, 97, 98
nobs, 26, 97
INDEX

responseVariable, lcMethod-method (responseVariable), 122
responseVariable, lcModel-method (responseVariable), 122

sigma(), 123
sigma.lcModel, , 16, 17, 20, 25, 26, 33, 34, 85, 89, 95, 98, 110, 113–116, 122, 123, 127
stats::AIC(), 91
stats::BIC(), 91
stats::deviance, , 25
stats::deviance(), 91
stats::df.residual, 26
stats::fitted, 33
stats::formula, 36
stats::lm, 74
stats::logLik, 89
stats::logLik(), 91
stats::model.frame, 95
stats::model.frame(), 95
stats::predict, 112
stats::sigma(), 91
strip, 123
strip, ANY-method (strip), 123
strip, lcModel-method (strip), 123
subset.lcModels, , 11, 87, 120, 124
summary.lcModel, 125

test.latrend, 126
time.lcModel, , 16, 17, 20, 25, 26, 33, 34, 85, 89, 95, 98, 110, 113–116, 122, 123, 127
timeVariable, , 45, 122, 127, 128
timeVariable, lcMethod-method (timeVariable), 128
timeVariable, lcModel-method (timeVariable), 128
trajectories, , 105–107, 112, 122, 128
trajectories(), 47, 54, 138
trajectories, call-method (trajectories), 128
trajectories, data.frame-method (trajectories), 128
trajectories, lcModel-method (trajectories), 128
trajectories, matrix-method (trajectories), 128

trajectoryAssignments, , 7, 15, 19, 33, 104, 110, 130, 140
trajectoryAssignments(), 15, 19, 44
trajectoryAssignments, lcModel-method (trajectoryAssignments), 130
trajectoryAssignments, matrix-method (trajectoryAssignments), 130
transformFitted, 33, 131
transformFitted(), 33
transformFitted, data.frame, lcModel-method (transformFitted), 131
transformFitted, list, lcModel-method (transformFitted), 131
transformFitted, matrix, lcModel-method (transformFitted), 131
transformFitted, NULL, lcModel-method (transformFitted), 131
transformPredict, 133
transformPredict(), 112
transformPredict, data.frame, lcModel-method (transformPredict), 133
transformPredict, matrix, lcModel-method (transformPredict), 133
transformPredict, NULL, lcModel-method (transformPredict), 133
transformPredict, vector, lcModel-method (transformPredict), 133
tsf, 134, 137
tsmatrix, 135, 136

update.lcMethod, 8–10, 12, 27, 35, 56, 57, 96, 137, 141
update.lcModel, 138

validate, 139
validate(), 18, 32, 56, 57, 109, 117, 119, 140
validate, lcMethod-method (validate), 139
Verbose, 47, 49, 51, 52, 54, 138

which.is.max, 130
which.max(), 130
which.weight, 140
which.weight(), 130