

Package: latrend (via r-universe)

September 18, 2024

Type Package

Version 1.6.1

Date 2024-05-15

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. The 'akmedoids' package is available from <https://github.com/MAnalytics/akmedoids>.

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

URL <https://github.com/philips-software/latrend>,
<https://philips-software.github.io/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 (>= 1.15.4), magrittr, matrixStats,
rmarkdown (>= 1.18), rlang

Suggests testthat (>= 3.0.0), roxygen2 (>= 7.1.0), knitr (>= 1.24),
rcmdcheck, pkgdown, devtools, cluster, evaluate, lme4, covr,
lintr, tinytex, longitudinalData (>= 2.4.1), kml (>= 2.4.1),
lcmm (>= 1.9.3), mixtools, flexmix, fda, funFEM, gridExtra,
igraph, crimCV, dtwclust, mixAK, mclust, mclustcomp, clValid,

psych, qqplotr, doParallel, simTool, dplyr, ggplot2, caret,
tibble, clusterCrit (>= 1.3.0)

RoxygenNote 7.3.1

Roxygen list(markdown = TRUE)

RdMacros Rdpack

VignetteBuilder knitr

Collate 'assert.R' 'citation.R' 'compute.R' 'data.R' 'formula.R'
'generics.R' 'latrend.R' 'make.R' 'matrix.R' 'method.R'
'meta-method.R' 'meta-fit.R' 'meta-fit-converged.R'
'meta-fit-rep.R' 'methodMatrix.R' 'methodAKMedoids.R'
'methodCrimCV.R' 'methodDtwclust.R' 'trajectories.R' 'model.R'
'modelApprox.R' 'modelPartition.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' 'methodRandom.R' 'methodStratify.R'
'methods.R' 'metrics.R' 'metricsInternal.R' 'metricsExternal.R'
'model-evaluation.R' 'model-summary.R' 'model-transform.R'
'modelCrimCV.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' 'modelStratify.R'
'modelWeightedPartition.R' 'models.R' 'random.R' 'test.R'
'timing.R' 'variables.R' 'verbose.R' 'zzz.R'

Repository <https://philips-software.r-universe.dev>

RemoteUrl <https://github.com/philips-software/latrend>

RemoteRef HEAD

RemoteSha 4bd6cb73ecc99a70ed3ccbc8aa6d5208ed62187b

Contents

latrend-package	5
APPA	8
as.data.frame.lcMethod	8
as.data.frame.lcMethods	9
as.data.frame.lcModels	10
as.lcMethods	11
as.lcModels	11
as.list.lcMethod	12
clusterNames	13
clusterNames<-	14
clusterProportions	15
clusterSizes	16

clusterTrajectories	17
coef.lcModel	18
compose	19
confusionMatrix	20
converged	21
createTestDataFold	22
createTestDataFolds	23
createTrainDataFolds	24
defineExternalMetric	25
defineInternalMetric	26
deviance.lcModel	26
df.residual.lcModel	27
estimationTime	28
evaluate.lcMethod	29
externalMetric	30
fit	33
fitted.lcModel	35
fittedTrajectories	36
formula.lcMethod	37
formula.lcModel	38
generateLongData	39
getArgumentDefaults	40
getArgumentExclusions	41
getCitation	42
getExternalMetricDefinition	43
getExternalMetricNames	44
getInternalMetricDefinition	44
getInternalMetricNames	45
getLabel	45
getLcMethod	46
getName	47
ids	48
idVariable	49
initialize.lcMethod-method	50
interface-metaMethods	50
latrend	52
latrend-approaches	53
latrend-data	54
latrend-estimation	55
latrend-generics	56
latrend-methods	56
latrend-metrics	57
latrend-parallel	59
latrendBatch	61
latrendBoot	62
latrendCV	63
latrendData	65
latrendRep	66

lcApproxModel-class	67
lcFitMethods	68
lcMethod-class	69
lcMethod-estimation	71
lcMethodAkmedoids	72
lcMethodCrimCV	73
lcMethodDtwclust	74
lcMethodFeature	75
lcMethodFlexmix	77
lcMethodFlexmixGBTM	78
lcMethodFunction	79
lcMethodFunFEM	80
lcMethodGCKM	81
lcMethodKML	83
lcMethodLcmmGBTM	84
lcMethodLcmmGMM	85
lcMethodLMKM	87
lcMethodMclustLLPA	88
lcMethodMixAK_GLMM	89
lcMethodMixtoolsGMM	91
lcMethodMixtoolsNPRM	92
lcMethodMixTVEM	93
lcMethodRandom	94
lcMethods	96
lcMethodStratify	97
lcModel	98
lcModel-class	100
lcModelPartition	101
lcModels	103
lcModels-class	104
lcModelWeightedPartition	105
logLik.lcModel	105
max.lcModels	106
metric	108
min.lcModels	111
model.data.lcModel	112
model.frame.lcModel	113
names,lcMethod-method	114
nClusters	115
nIds	116
nobs.lcModel	117
OCC	117
PAP.adh	118
PAP.adh1y	119
plot-lcModel-method	121
plot-lcModels-method	122
plotClusterTrajectories	122
plotFittedTrajectories	125

plotMetric	126
plotTrajectories	127
postFit	129
postprob	130
postprobFromAssignments	132
predict.lcModel	132
predictAssignments	134
predictForCluster	135
predictPostprob	137
preFit	138
prepareData	139
print.lcMethod	141
print.lcModels	141
qqPlot	142
residuals.lcModel	143
responseVariable	144
sigma.lcModel	145
strip	146
subset.lcModels	147
summary.lcModel	148
test.latrend	149
time.lcModel	150
timeVariable	151
trajectories	152
trajectoryAssignments	154
transformFitted	155
transformPredict	157
tsframe	158
tsmatrix	159
update.lcMethod	161
update.lcModel	162
validate	163
which.weight	164
[[,lcMethod-method	165

Index**166**

latrend-package

*latrend: 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. The 'akmedoids' package is available from <https://github.com/MAnalytics/akmedoids>.

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**.

The supported types of longitudinal datasets are described [here](#).

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.

```
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.

```
km1Method <- lcMethodKML("Y", nClusters = 3)
km1Method
```

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

```
model <- latrend(km1Method, data = latrendData)
```

We can then investigate the fitted model using

```
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()`.

```
km1Methods <- lcMethods(km1Method, nClusters = 1:5)
models <- latrendBatch(km1Methods, data = latrendData)
```

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

```
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

Data requirements and datasets: [latrend-data](#) [latrendData](#) [PAP.adh](#)

High-level method recommendations and supported methods: [latrend-approaches](#) [latrend-methods](#)

Method specification: [lcMethod](#) [lcMethods](#)

Method estimation: [latrend](#) [latrendRep](#) [latrendBatch](#) [latrendBoot](#) [latrendCV](#) [latrend-parallel](#) [Steps performed during estimation](#)

Model functions: [lcModel](#) [clusterTrajectories](#) [plotClusterTrajectories](#) [postprob](#) [trajectoryAssignments](#) [predictPostprob](#) [predictAssignments](#) [predict.lcModel](#) [predictForCluster](#) [fitted.lcModel](#) [fitted-Trajectories](#)

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]
- Koninklijke Philips N.V. [copyright holder]

See Also

Useful links:

- <https://github.com/philips-software/latrend>
- <https://philips-software.github.io/latrend/>
- Report bugs at <https://github.com/philips-software/latrend/issues>

 APPA

Average posterior probability of assignment (APPA)

Description

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

Usage

```
APPA(object)
```

Arguments

object The model, of type lcModel.

Value

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

References

Nagin DS (2005). *Group-based modeling of development*. Harvard University Press. ISBN 9780674041318, doi:10.4159/9780674041318.

Klijn SL, Weijenberg MP, Lemmens P, van den Brandt PA, Passos VL (2017). “Introducing the fit-criteria assessment plot - A visualisation tool to assist class enumeration in group-based trajectory modelling.” *Statistical Methods in Medical Research*, **26**(5), 2424-2436.

van der Nest G, Lima Passos V, Candel MJ, van Breukelen GJ (2020). “An overview of mixture modelling for latent evolutions in longitudinal data: Modelling approaches, fit statistics and software.” *Advances in Life Course Research*, **43**, 100323. ISSN 1040-2608, doi:10.1016/j.alcr.2019.100323.

See Also

[confusionMatrix OCC](#)

 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\(\)](#)

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(x, ..., eval = TRUE, nullValue = NA, envir = parent.frame())
```

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](#), [names,lcMethod-method](#), [update.lcMethod\(\)](#)

as.data.frame.lcModels

Generate a data.frame containing the argument values per method per row

Description

Generate a data.frame containing the argument values per method per row

Usage

```
## S3 method for class 'lcModels'
as.data.frame(x, ..., excludeShared = FALSE, eval = TRUE)
```

Arguments

x	lcModels or a list of lcModel
...	Arguments passed to as.data.frame.lcMethod .
excludeShared	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.

Functionality

- [Print](#) an argument summary for each of the models.
- [Convert](#) to a data.frame of method arguments.
- [Subset](#) the list.
- Compute an [internal metric](#) or [external metric](#).
- Obtain the best model according to [minimizing](#) or [maximizing](#) a [metric](#).
- Obtain the summed [estimation time](#).
- [Plot a metric](#) across a variable.
- [Plot the cluster trajectories](#).

as.lcMethods	<i>Convert a list of lcMethod objects to a lcMethods list</i>
--------------	---

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.lcMethod\(\), as.data.frame.lcMethods\(\), as.list.lcMethod\(\), evaluate.lcMethod\(\), formula.lcMethod\(\), lcMethod-class, names, lcMethod-method, update.lcMethod\(\)\]](#)

as.lcModels	<i>Convert a list of lcModels to a lcModels list</i>
-------------	--

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.

Functionality

- **Print** an argument summary for each of the models.
- **Convert** to a `data.frame` of method arguments.
- **Subset** the list.
- Compute an **internal metric** or **external metric**.
- Obtain the best model according to **minimizing** or **maximizing** a **metric**.
- Obtain the summed **estimation time**.
- **Plot a metric** across a variable.
- **Plot the cluster trajectories**.

See Also

lcModels

Other lcModels functions: `lcModels`, `lcModels-class`, `max.lcModels()`, `min.lcModels()`, `plotMetric()`, `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

<code>x</code>	The <code>lcMethod</code> object.
<code>...</code>	Additional arguments.
<code>args</code>	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.
<code>eval</code>	Whether to evaluate the arguments.
<code>expand</code>	Whether to return all method arguments when <code>"..."</code> is present among the requested argument names.
<code>envir</code>	The environment in which to evaluate the arguments. If <code>NULL</code> , the environment associated with the object is used. If not available, the <code>parent.frame()</code> 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.lcMethod\(\)](#), [as.data.frame.lcMethods\(\)](#), [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.

Value

A character of the cluster names.

See Also

Other lcModel functions: `clusterProportions()`, `clusterSizes()`, `clusterTrajectories()`, `coef.lcModel()`, `converged()`, `deviance.lcModel()`, `df.residual.lcModel()`, `estimationTime()`, `externalMetric()`, `fitted.lcModel()`, `fittedTrajectories()`, `getCall.lcModel()`, `getLcMethod()`, `ids()`, `lcModel-class`, `metric()`, `model.frame.lcModel()`, `nClusters()`, `nIds()`, `nobs.lcModel()`, `plot-lcModel-method`, `plotClusterTrajectories()`, `plotFittedTrajectories()`, `postprob()`, `predict.lcModel()`, `predictAssignments()`, `predictForCluster()`, `predictPostprob()`, `qqPlot()`, `residuals.lcModel()`, `sigma.lcModel()`, `strip()`, `time.lcModel()`, `trajectoryAssignments()`

Examples

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

`clusterNames<-` *Update the cluster names*

Description

Update the cluster names

Usage

```
clusterNames(object) <- value
```

Arguments

<code>object</code>	The lcModel object to update.
<code>value</code>	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	<i>Proportional size of each cluster</i>
--------------------	--

Description

Obtain the proportional size per cluster, between 0 and 1.

Usage

```
clusterProportions(object, ...)

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

Arguments

object	The model.
...	For lcModel objects: Additional arguments passed to postprob() .

Value

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

lcModel

By default, the cluster proportions are determined from the cluster-averaged posterior probabilities of the fitted data (as computed by the [postprob\(\)](#) function).

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

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

See Also

[nClusters](#) [clusterNames](#)

[clusterSizes](#) [postprob](#)

Other `lcModel` functions: [clusterNames\(\)](#), [clusterSizes\(\)](#), [clusterTrajectories\(\)](#), [coef.lcModel\(\)](#), [converged\(\)](#), [deviance.lcModel\(\)](#), [df.residual.lcModel\(\)](#), [estimationTime\(\)](#), [externalMetric\(\)](#), [fitted.lcModel\(\)](#), [fittedTrajectories\(\)](#), [getCall.lcModel\(\)](#), [getLcMethod\(\)](#), [ids\(\)](#), [lcModel-class](#), [metric\(\)](#), [model.frame.lcModel\(\)](#), [nClusters\(\)](#), [nIds\(\)](#), [nobs.lcModel\(\)](#), [plot-lcModel-method](#), [plotClusterTrajectories\(\)](#), [plotFittedTrajectories\(\)](#), [postprob\(\)](#), [predict.lcModel\(\)](#), [predictAssignments\(\)](#), [predictForCluster\(\)](#), [predictPostprob\(\)](#), [qqPlot\(\)](#), [residuals.lcModel\(\)](#), [sigma.lcModel\(\)](#), [strip\(\)](#), [time.lcModel\(\)](#), [trajectoryAssignments\(\)](#)

Examples

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

clusterSizes	<i>Number of trajectories per cluster</i>
--------------	---

Description

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

Usage

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

Arguments

object	The lcModel object.
...	Additional arguments passed 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](#)

Other lcModel functions: [clusterNames\(\)](#), [clusterProportions\(\)](#), [clusterTrajectories\(\)](#), [coef.lcModel\(\)](#), [converged\(\)](#), [deviance.lcModel\(\)](#), [df.residual.lcModel\(\)](#), [estimationTime\(\)](#), [externalMetric\(\)](#), [fitted.lcModel\(\)](#), [fittedTrajectories\(\)](#), [getCall.lcModel\(\)](#), [getLcMethod\(\)](#), [ids\(\)](#), [lcModel-class](#), [metric\(\)](#), [model.frame.lcModel\(\)](#), [nClusters\(\)](#), [nIds\(\)](#), [nobs.lcModel\(\)](#), [plot-lcModel-method](#), [plotClusterTrajectories\(\)](#), [plotFittedTrajectories\(\)](#), [postprob\(\)](#), [predict.lcModel\(\)](#), [predictAssignments\(\)](#), [predictForCluster\(\)](#), [predictPostprob\(\)](#), [qqPlot\(\)](#), [residuals.lcModel\(\)](#), [sigma.lcModel\(\)](#), [strip\(\)](#), [time.lcModel\(\)](#), [trajectoryAssignments\(\)](#)

Examples

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

clusterTrajectories *Extract cluster trajectories*

Description

Extracts a data.frame of the cluster trajectories associated with the given object.

Usage

```
clusterTrajectories(object, ...)

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

Arguments

object	The model.
...	For lcModel objects: Arguments passed to predict.lcModel .
at	A numeric vector of the times at which to compute the cluster trajectories.
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'.

Value

A data.frame of the estimated values at the specified 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

[plotClusterTrajectories](#)

Other lcModel functions: [clusterNames\(\)](#), [clusterProportions\(\)](#), [clusterSizes\(\)](#), [coef.lcModel\(\)](#), [converged\(\)](#), [deviance.lcModel\(\)](#), [df.residual.lcModel\(\)](#), [estimationTime\(\)](#), [externalMetric\(\)](#), [fitted.lcModel\(\)](#), [fittedTrajectories\(\)](#), [getCall.lcModel\(\)](#), [getLcMethod\(\)](#), [ids\(\)](#), [lcModel-class](#), [metric\(\)](#), [model.frame.lcModel\(\)](#), [nClusters\(\)](#), [nIds\(\)](#), [nobs.lcModel\(\)](#), [plot-lcModel-method](#), [plotClusterTrajectories\(\)](#), [plotFittedTrajectories\(\)](#), [postprob\(\)](#), [predict.lcModel\(\)](#), [predictAssignments\(\)](#), [predictForCluster\(\)](#), [predictPostprob\(\)](#), [qqPlot\(\)](#), [residuals.lcModel\(\)](#), [sigma.lcModel\(\)](#), [strip\(\)](#), [time.lcModel\(\)](#), [trajectoryAssignments\(\)](#)

Examples

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

clusterTrajectories(model)

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

coef.lcModel	<i>Extract lcModel coefficients</i>
--------------	-------------------------------------

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

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

Arguments

<code>object</code>	The <code>lcModel</code> object.
<code>...</code>	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.

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

See Also

Other lcModel functions: `clusterNames()`, `clusterProportions()`, `clusterSizes()`, `clusterTrajectories()`, `converged()`, `deviance.lcModel()`, `df.residual.lcModel()`, `estimationTime()`, `externalMetric()`, `fitted.lcModel()`, `fittedTrajectories()`, `getCall.lcModel()`, `getLcMethod()`, `ids()`, `lcModel-class`, `metric()`, `model.frame.lcModel()`, `nClusters()`, `nIds()`, `nobs.lcModel()`, `plot-lcModel-method`, `plotClusterTrajectories()`, `plotFittedTrajectories()`, `postprob()`, `predict.lcModel()`, `predictAssignments()`, `predictForCluster()`, `predictPostprob()`, `qqPlot()`, `residuals.lcModel()`, `sigma.lcModel()`, `strip()`, `time.lcModel()`, `trajectoryAssignments()`

Other lcModel functions: `clusterNames()`, `clusterProportions()`, `clusterSizes()`, `clusterTrajectories()`, `converged()`, `deviance.lcModel()`, `df.residual.lcModel()`, `estimationTime()`, `externalMetric()`, `fitted.lcModel()`, `fittedTrajectories()`, `getCall.lcModel()`, `getLcMethod()`, `ids()`, `lcModel-class`, `metric()`, `model.frame.lcModel()`, `nClusters()`, `nIds()`, `nobs.lcModel()`, `plot-lcModel-method`, `plotClusterTrajectories()`, `plotFittedTrajectories()`, `postprob()`, `predict.lcModel()`, `predictAssignments()`, `predictForCluster()`, `predictPostprob()`, `qqPlot()`, `residuals.lcModel()`, `sigma.lcModel()`, `strip()`, `time.lcModel()`, `trajectoryAssignments()`

Examples

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

compose

lcMethod *estimation step: compose an lcMethod object*

Description

Note: this function should not be called directly, as it is part of the lcMethod [estimation procedure](#). For fitting an lcMethod object to a dataset, use the `latrend()` function or [one of the other standard estimation functions](#).

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

```
compose(method, envir, ...)

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

Arguments

<code>method</code>	The lcMethod object.
<code>envir</code>	The environment in which the lcMethod should be evaluated
<code>...</code>	Not used.

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.

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

Estimation procedure

The steps for estimating a lcMethod object are defined and executed as follows:

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

[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

```
confusionMatrix(object, strategy = which.max, scale = TRUE, ...)
```

Arguments

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

Value

A K-by-K confusion matrix with $K = nClusters(object)$.

See Also

[postprob](#) [clusterProportions](#) [trajectoryAssignments](#) [APPA](#) [OCC](#)

Examples

```
data(latrendData)

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

converged

Check model convergence

Description

Check whether the fitted object converged.

Usage

```
converged(object, ...)
```

S4 method for signature 'lcModel'

```
converged(object, ...)
```

Arguments

object	The model.
...	Not used.

Value

Either logical indicating convergence, or a numeric status code.

The default lcModel implementation returns NA.

Implementation

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

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

See Also

Other lcModel functions: [clusterNames\(\)](#), [clusterProportions\(\)](#), [clusterSizes\(\)](#), [clusterTrajectories\(\)](#), [coef.lcModel\(\)](#), [deviance.lcModel\(\)](#), [df.residual.lcModel\(\)](#), [estimationTime\(\)](#), [externalMetric\(\)](#), [fitted.lcModel\(\)](#), [fittedTrajectories\(\)](#), [getCall.lcModel\(\)](#), [getLcMethod\(\)](#), [ids\(\)](#), [lcModel-class](#), [metric\(\)](#), [model.frame.lcModel\(\)](#), [nClusters\(\)](#), [nIds\(\)](#), [nobs.lcModel\(\)](#), [plot-lcModel-method](#), [plotClusterTrajectories\(\)](#), [plotFittedTrajectories\(\)](#), [postprob\(\)](#), [predict.lcModel\(\)](#), [predictAssignments\(\)](#), [predictForCluster\(\)](#), [predictPostprob\(\)](#), [qqPlot\(\)](#), [residuals.lcModel\(\)](#), [sigma.lcModel\(\)](#), [strip\(\)](#), [time.lcModel\(\)](#), [trajectoryAssignments\(\)](#)

Examples

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

createTestDataFold *Create the test fold data for validation*

Description

Create the test fold data for validation

Usage

```
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

```
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

```
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
)
```

Arguments

<code>data</code>	A data.frame representing the complete dataset.
<code>folds</code>	The number of folds. By default, a 10-fold scheme is used.
<code>id</code>	The trajectory identifier variable.
<code>seed</code>	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: [createTestDataFold\(\)](#), [createTestDataFolds\(\)](#), [latrendBoot\(\)](#), [latrendCV\(\)](#), [lcModel-data-filters](#)

Examples

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

if (require("caret")) {
  trainFolds <- createTrainDataFolds(latrendData, folds = 5, id = "Id", seed = 1)

  foldModels <- latrendBatch(method, data = trainFolds)
  testDataFolds <- createTestDataFolds(latrendData, trainFolds)
}
```

defineExternalMetric *Define an external metric for lcModels*

Description

Define an external metric for lcModels

Usage

```
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.

See Also

Other metric functions: [defineInternalMetric\(\)](#), [externalMetric\(\)](#), [getExternalMetricDefinition\(\)](#), [getExternalMetricNames\(\)](#), [getInternalMetricDefinition\(\)](#), [getInternalMetricNames\(\)](#), [metric\(\)](#)

```
defineInternalMetric Define an internal metric for lcModels
```

Description

Define an internal metric for lcModels

Usage

```
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.

See Also

Other metric functions: [defineExternalMetric\(\)](#), [externalMetric\(\)](#), [getExternalMetricDefinition\(\)](#), [getExternalMetricNames\(\)](#), [getInternalMetricDefinition\(\)](#), [getInternalMetricNames\(\)](#), [metric\(\)](#)

Examples

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

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

```
deviance.lcModel lcModel deviance
```

Description

Get the deviance of the fitted lcModel object.

Usage

```
## 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 lcModel functions: [clusterNames\(\)](#), [clusterProportions\(\)](#), [clusterSizes\(\)](#), [clusterTrajectories\(\)](#), [coef.lcModel\(\)](#), [converged\(\)](#), [df.residual.lcModel\(\)](#), [estimationTime\(\)](#), [externalMetric\(\)](#), [fitted.lcModel\(\)](#), [fittedTrajectories\(\)](#), [getCall.lcModel\(\)](#), [getLcMethod\(\)](#), [ids\(\)](#), [lcModel-class](#), [metric\(\)](#), [model.frame.lcModel\(\)](#), [nClusters\(\)](#), [nIds\(\)](#), [nobs.lcModel\(\)](#), [plot-lcModel-method](#), [plotClusterTrajectories\(\)](#), [plotFittedTrajectories\(\)](#), [postprob\(\)](#), [predict.lcModel\(\)](#), [predictAssignments\(\)](#), [predictForCluster\(\)](#), [predictPostprob\(\)](#), [qqPlot\(\)](#), [residuals.lcModel\(\)](#), [sigma.lcModel\(\)](#), [strip\(\)](#), [time.lcModel\(\)](#), [trajectoryAssignments\(\)](#)

df.residual.lcModel *Extract the residual degrees of freedom from a lcModel*

Description

Extract the residual degrees of freedom from a lcModel

Usage

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

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 lcModel functions: [clusterNames\(\)](#), [clusterProportions\(\)](#), [clusterSizes\(\)](#), [clusterTrajectories\(\)](#), [coef.lcModel\(\)](#), [converged\(\)](#), [deviance.lcModel\(\)](#), [estimationTime\(\)](#), [externalMetric\(\)](#), [fitted.lcModel\(\)](#), [fittedTrajectories\(\)](#), [getCall.lcModel\(\)](#), [getLcMethod\(\)](#), [ids\(\)](#), [lcModel-class](#), [metric\(\)](#), [model.frame.lcModel\(\)](#), [nClusters\(\)](#), [nIds\(\)](#), [nobs.lcModel\(\)](#), [plot-lcModel-method](#), [plotClusterTrajectories\(\)](#), [plotFittedTrajectories\(\)](#), [postprob\(\)](#), [predict.lcModel\(\)](#), [predictAssignments\(\)](#), [predictForCluster\(\)](#), [predictPostprob\(\)](#), [qqPlot\(\)](#), [residuals.lcModel\(\)](#), [sigma.lcModel\(\)](#), [strip\(\)](#), [time.lcModel\(\)](#), [trajectoryAssignments\(\)](#)

estimationTime	<i>Estimation time</i>
----------------	------------------------

Description

Get the elapsed time for estimating the given model.

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

Usage

```
estimationTime(object, unit = "secs", ...)

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

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

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

Arguments

object	The model.
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 .
...	Not used.

Value

A non-negative scalar numeric representing the estimation time in the specified unit..

See Also

Other lcModel functions: `clusterNames()`, `clusterProportions()`, `clusterSizes()`, `clusterTrajectories()`, `coef.lcModel()`, `converged()`, `deviance.lcModel()`, `df.residual.lcModel()`, `externalMetric()`, `fitted.lcModel()`, `fittedTrajectories()`, `getCall.lcModel()`, `getLcMethod()`, `ids()`, `lcModel-class`, `metric()`, `model.frame.lcModel()`, `nClusters()`, `nIds()`, `nobs.lcModel()`, `plot-lcModel-method`, `plotClusterTrajectories()`, `plotFittedTrajectories()`, `postprob()`, `predict.lcModel()`, `predictAssignments()`, `predictForCluster()`, `predictPostprob()`, `qqPlot()`, `residuals.lcModel()`, `sigma.lcModel()`, `strip()`, `time.lcModel()`, `trajectoryAssignments()`

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

<code>object</code>	The lcMethod object.
<code>classes</code>	Substitute only arguments with specific class types. By default, all types are substituted.
<code>try</code>	Whether to try to evaluate arguments and ignore errors (the default), or to fail on any argument evaluation error.
<code>exclude</code>	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.
...	Not used.

Value

A new lcMethod object with the substituted arguments.

See Also

[compose](#)

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

externalMetric	<i>Compute external model metric(s)</i>
----------------	---

Description

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

Usage

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

## S4 method for signature 'lcModels,missing'
```

```

externalMetric(object, object2, name = "adjustedRand")

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

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

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

```

Arguments

object	The object to compare to the second object
object2	The second object
name	The name(s) of the external metric(s) to compute. If no names are given, the names specified in the <code>latrend.externalMetric</code> option (none by default) are used.
...	Additional arguments.
drop	Whether to return a numeric vector instead of a <code>data.frame</code> in case of a single metric.

Value

For `externalMetric(lcModel, lcModel)`: A numeric vector of the computed 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.

Supported external metrics

Metric name	Description
adjustedRand	Adjusted Rand index . Based on the Rand index, but adjusted for agreements occurring by chance. A score of 1 indicates a perfect agreement.
CohensKappa	Cohen's kappa . A partitioning agreement metric correcting for random chance. A score of 1 indicates a perfect agreement.
F	F-score
F1	F1-score , also referred to as the Sørensen–Dice Coefficient , or Dice similarity coefficient
FolkesMallows	Fowlkes-Mallows index
Hubert	Hubert index
Jaccard	Jaccard index
jointEntropy	Joint entropy between model assignments
Kulczynski	Kulczynski index

MaximumMatch	Maximum match measure
McNemar	McNemar statistic
MeilaHeckerman	Meila-Heckerman measure
Mirkin	Mirkin metric
MI	Mutual information
NMI	Normalized mutual information
NSJ	Normalized version of splitJoin. The proportion of edits relative to the maximum changes (twice the number of nodes).
NVI	Normalized variation of information
Overlap	Overlap coefficient , also referred to as the Szymkiewicz–Simpson coefficient
PD	Partition difference
Phi	Phi coefficient .
precision	precision
Rand	Rand index
recall	recall
RogersTanimoto	Rogers-Tanimoto dissimilarity
RusselRao	Russell-Rao dissimilarity
SMC	Simple matching coefficient
splitJoin	total split-join index
splitJoin.ref	Split-join index of the first model to the second model. In other words, it is the edit-distance between the two models.
SokalSneath1	Type-1 Sokal-Sneath dissimilarity
SokalSneath2	Type-2 Sokal-Sneath dissimilarity
VI	Variation of information
Wallace1	Type-1 Wallace criterion
Wallace2	Type-2 Wallace criterion
WMSSE	Weighted minimum sum of squared errors between cluster trajectories
WMMSE	Weighted minimum mean of squared errors between cluster trajectories
WMAE	Weighted minimum mean of absolute errors between cluster trajectories

Implementation

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

References

- Cohen J (1960). “A Coefficient of Agreement for Nominal Scales.” *Educational and Psychological Measurement*, **20**(1), 37-46.
- Csardi G, Nepusz T (2006). “The igraph software package for complex network research.” *InterJournal*, **Complex Systems**, 1695. <https://igraph.org>.
- Desgraupes B (2018). *clusterCrit: Clustering Indices*. R package version 1.2.8, <https://CRAN.R-project.org/package=clusterCrit>.
- Hubert L, Arabie P (1985). “Comparing Partitions.” *Journal of Classification*, **2**(1), 193–218. ISSN 1432-1343, [doi:10.1007/BF01908075](https://doi.org/10.1007/BF01908075).

M K V, K K (2016). "A Survey on Similarity Measures in Text Mining." *Machine Learning and Applications: An International Journal*, **3**, 19-28. doi:10.5121/mlaij.2016.3103.

Revelle W (2019). *psych: Procedures for Psychological, Psychometric, and Personality Research*. Northwestern University, Evanston, Illinois. R package version 1.9.12, <https://CRAN.R-project.org/package=psych>.

You K (2018). *mclustcomp: Measures for Comparing Clusters*. R package version 0.3.1, <https://CRAN.R-project.org/package=mclustcomp>.

See Also

[metric](#)

Other metric functions: [defineExternalMetric\(\)](#), [defineInternalMetric\(\)](#), [getExternalMetricDefinition\(\)](#), [getExternalMetricNames\(\)](#), [getInternalMetricDefinition\(\)](#), [getInternalMetricNames\(\)](#), [metric\(\)](#)

Other lcModel functions: [clusterNames\(\)](#), [clusterProportions\(\)](#), [clusterSizes\(\)](#), [clusterTrajectories\(\)](#), [coef.lcModel\(\)](#), [converged\(\)](#), [deviance.lcModel\(\)](#), [df.residual.lcModel\(\)](#), [estimationTime\(\)](#), [fitted.lcModel\(\)](#), [fittedTrajectories\(\)](#), [getCall.lcModel\(\)](#), [getLcMethod\(\)](#), [ids\(\)](#), [lcModel-class](#), [metric\(\)](#), [model.frame.lcModel\(\)](#), [nClusters\(\)](#), [nIds\(\)](#), [nobs.lcModel\(\)](#), [plot-lcModel-method](#), [plotClusterTrajectories\(\)](#), [plotFittedTrajectories\(\)](#), [postprob\(\)](#), [predict.lcModel\(\)](#), [predictAssignments\(\)](#), [predictForCluster\(\)](#), [predictPostprob\(\)](#), [qqPlot\(\)](#), [residuals.lcModel\(\)](#), [sigma.lcModel\(\)](#), [strip\(\)](#), [time.lcModel\(\)](#), [trajectoryAssignments\(\)](#)

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")
}
```

fit

lcMethod *estimation step: logic for fitting the method to the processed data*

Description

Note: this function should not be called directly, as it is part of the lcMethod [estimation procedure](#). For fitting an lcMethod object to a dataset, use the [latrend\(\)](#) function or [one of the other standard estimation functions](#).

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

Usage

```
fit(method, data, envir, verbose, ...)
```

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

Arguments

method	An object inheriting from <code>lcMethod</code> 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 <code>prepareData()</code> and <code>preFit()</code> .
verbose	A <code>R.utils::Verbose</code> object indicating the level of verbosity.
...	Not used.

Value

The fitted object, inheriting from `lcModel`.

Implementation

This method should be implemented for all `lcMethod` subclasses.

```
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)
  )
})
```

Estimation procedure

The steps for estimating a `lcMethod` object are defined and executed as follows:

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

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

Arguments

<code>object</code>	The <code>lcModel</code> object.
<code>...</code>	Additional arguments.
<code>clusters</code>	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 `lcModel` functions: [clusterNames\(\)](#), [clusterProportions\(\)](#), [clusterSizes\(\)](#), [clusterTrajectories\(\)](#), [coef.lcModel\(\)](#), [converged\(\)](#), [deviance.lcModel\(\)](#), [df.residual.lcModel\(\)](#), [estimationTime\(\)](#), [externalMetric\(\)](#), [fittedTrajectories\(\)](#), [getCall.lcModel\(\)](#), [getLcMethod\(\)](#), [ids\(\)](#), [lcModel-class](#), [metric\(\)](#), [model.frame.lcModel\(\)](#), [nClusters\(\)](#), [nIds\(\)](#), [nobs.lcModel\(\)](#), [plot-lcModel-method](#), [plotClusterTrajectories\(\)](#), [plotFittedTrajectories\(\)](#), [postprob\(\)](#), [predict.lcModel\(\)](#), [predictAssignments\(\)](#), [predictForCluster\(\)](#), [predictPostprob\(\)](#), [qqPlot\(\)](#), [residuals.lcModel\(\)](#), [sigma.lcModel\(\)](#), [strip\(\)](#), [time.lcModel\(\)](#), [trajectoryAssignments\(\)](#)

Examples

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

`fittedTrajectories` *Extract the fitted trajectories*

Description

Extract the fitted trajectories

Usage

```
fittedTrajectories(object, ...)

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

Arguments

<code>object</code>	The model.
<code>...</code>	For <code>lcModel</code> : Additional arguments passed to fitted.lcModel .
<code>at</code>	The time points at which to compute the id-specific trajectories. The default implementation merely filters the output, i.e., fitted values can only be outputted for times at which the model was trained.
<code>what</code>	The distributional parameter to compute the response for.
<code>clusters</code>	The cluster assignments for the strata to base the trajectories on.

Details

The default lcModel 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.

For lcModel: A data.frame with columns id, time, response, and "Cluster".

See Also

[plotFittedTrajectories](#)

Other lcModel functions: [clusterNames\(\)](#), [clusterProportions\(\)](#), [clusterSizes\(\)](#), [clusterTrajectories\(\)](#), [coef.lcModel\(\)](#), [converged\(\)](#), [deviance.lcModel\(\)](#), [df.residual.lcModel\(\)](#), [estimationTime\(\)](#), [externalMetric\(\)](#), [fitted.lcModel\(\)](#), [getCall.lcModel\(\)](#), [getLcMethod\(\)](#), [ids\(\)](#), [lcModel-class](#), [metric\(\)](#), [model.frame.lcModel\(\)](#), [nClusters\(\)](#), [nIds\(\)](#), [nobs.lcModel\(\)](#), [plot-lcModel-method](#), [plotClusterTrajectories\(\)](#), [plotFittedTrajectories\(\)](#), [postprob\(\)](#), [predict.lcModel\(\)](#), [predictAssignments\(\)](#), [predictForCluster\(\)](#), [predictPostprob\(\)](#), [qqPlot\(\)](#), [residuals.lcModel\(\)](#), [sigma.lcModel\(\)](#), [strip\(\)](#), [time.lcModel\(\)](#), [trajectoryAssignments\(\)](#)

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)])
```

formula.lcMethod	<i>Extract formula</i>
------------------	------------------------

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.lcMethod\(\), as.data.frame.lcMethods\(\), 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	<i>Extract the formula of a lcModel</i>
-----------------	---

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

```
## 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**

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

generateLongData	<i>Generate longitudinal test data</i>
------------------	--

Description

Generate longitudinal test data

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.

See Also

[latrend-data](#)

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 the given method specification*

Description

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

Usage

```
getArgumentDefaults(object, ...)

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

Arguments

object	The method specification object.
...	Not used.

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.

```
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.

See Also

[getArgumentExclusions](#)

[lcMethod](#)

Other `lcMethod` implementations: [getArgumentExclusions\(\)](#), [lcMethod-class](#), [lcMethodAkmedoids](#), [lcMethodCrimCV](#), [lcMethodDtwclust](#), [lcMethodFeature](#), [lcMethodFunFEM](#), [lcMethodFunction](#), [lcMethodGCKM](#), [lcMethodKML](#), [lcMethodLMKM](#), [lcMethodLcmmGBTM](#), [lcMethodLcmmGMM](#), [lcMethodMcLustLLPA](#), [lcMethodMixAK_GLMM](#), [lcMethodMixtoolsGMM](#), [lcMethodMixtoolsNPRM](#), [lcMethodRandom](#), [lcMethodStratify](#)

`getArgumentExclusions` *Arguments to be excluded from the specification*

Description

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

Usage

```
getArgumentExclusions(object, ...)

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

Arguments

object	The object.
...	Not used.

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 superclass

See Also

[getArgumentDefaults](#)

[lcMethod getArgumentExclusions](#)

Other `lcMethod` implementations: [getArgumentDefaults\(\)](#), [lcMethod-class](#), [lcMethodAkmedoids](#), [lcMethodCrimCV](#), [lcMethodDtwclust](#), [lcMethodFeature](#), [lcMethodFunFEM](#), [lcMethodFunction](#), [lcMethodGCKM](#), [lcMethodKML](#), [lcMethodLMKM](#), [lcMethodLcmmGBTM](#), [lcMethodLcmmGMM](#), [lcMethodMclustLLPA](#), [lcMethodMixAK_GLMM](#), [lcMethodMixtoolsGMM](#), [lcMethodMixtoolsNPRM](#), [lcMethodRandom](#), [lcMethodStratify](#)

getCitation

Get citation info

Description

Get a citation object indicating how to cite the underlying R packages used for estimating or representing the given method or model.

Usage

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

Arguments

object	The object
...	Not used.

Value

A [utils::citation](#) object.

See Also

[utils::citation](#)

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\(\)](#), [getExternalMetricNames\(\)](#), [getInternalMetricDefinition\(\)](#), [getInternalMetricNames\(\)](#), [metric\(\)](#)

`getExternalMetricNames`*Get the names of the available external metrics*

Description

Get the names of the available external metrics

Usage

```
getExternalMetricNames()
```

See Also

Other metric functions: [defineExternalMetric\(\)](#), [defineInternalMetric\(\)](#), [externalMetric\(\)](#), [getExternalMetricDefinition\(\)](#), [getInternalMetricDefinition\(\)](#), [getInternalMetricNames\(\)](#), [metric\(\)](#)

`getInternalMetricDefinition`*Get the internal metric definition*

Description

Get the internal metric definition

Usage

```
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\(\)](#), [getExternalMetricDefinition\(\)](#), [getExternalMetricNames\(\)](#), [getInternalMetricNames\(\)](#), [metric\(\)](#)

`getInternalMetricNames`*Get the names of the available internal metrics*

Description

Get the names of the available internal metrics

Usage

```
getInternalMetricNames()
```

See Also

Other metric functions: [defineExternalMetric\(\)](#), [defineInternalMetric\(\)](#), [externalMetric\(\)](#), [getExternalMetricDefinition\(\)](#), [getExternalMetricNames\(\)](#), [getInternalMetricDefinition\(\)](#), [metric\(\)](#)

`getLabel`*Object label*

Description

Get the object label, if any.

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

```
getLabel(object, ...)
```

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

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

Arguments

<code>object</code>	The object.
<code>...</code>	Not used.

Value

A scalar character. The empty string is returned if there is no label.

See Also[getName](#)[getName](#) [getShortName](#)**Examples**

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

getLabel(update(method, label = "v2")) # "v2"
```

`getLcMethod`*Get the method specification*

Description

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

Usage

```
getLcMethod(object, ...)

## S4 method for signature 'lcModel'
getLcMethod(object)
```

Arguments

<code>object</code>	The model.
<code>...</code>	Not used.

Value

An `lcMethod` object.

See Also[getCall.lcModel](#)

Other `lcModel` functions: [clusterNames\(\)](#), [clusterProportions\(\)](#), [clusterSizes\(\)](#), [clusterTrajectories\(\)](#), [coef.lcModel\(\)](#), [converged\(\)](#), [deviance.lcModel\(\)](#), [df.residual.lcModel\(\)](#), [estimationTime\(\)](#), [externalMetric\(\)](#), [fitted.lcModel\(\)](#), [fittedTrajectories\(\)](#), [getCall.lcModel\(\)](#), [ids\(\)](#), [lcModel-class](#), [metric\(\)](#), [model.frame.lcModel\(\)](#), [nClusters\(\)](#), [nIds\(\)](#), [nobs.lcModel\(\)](#), [plot-lcModel-method](#), [plotClusterTrajectories\(\)](#), [plotFittedTrajectories\(\)](#), [postprob\(\)](#), [predict.lcModel\(\)](#), [predictAssignments\(\)](#), [predictForCluster\(\)](#), [predictPostprob\(\)](#), [qqPlot\(\)](#), [residuals.lcModel\(\)](#), [sigma.lcModel\(\)](#), [strip\(\)](#), [time.lcModel\(\)](#), [trajectoryAssignments\(\)](#)

Examples

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

getName	<i>Object name</i>
---------	--------------------

Description

Get the name associated with the given object.

getShortName(): Extracts the short object name

Usage

```
getName(object, ...)
```

```
getShortName(object, ...)
```

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

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

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

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

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

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

Arguments

object The object.

... Not used.

Details

For lcModel: The name is determined by its associated lcMethod name and label, unless specified otherwise.

Value

A nonempty string, as character.

Implementation

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

```
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

[getShortName](#) [getLabel](#)

Examples

```
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

```
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.

See Also

Other lcModel functions: `clusterNames()`, `clusterProportions()`, `clusterSizes()`, `clusterTrajectories()`, `coef.lcModel()`, `converged()`, `deviance.lcModel()`, `df.residual.lcModel()`, `estimationTime()`, `externalMetric()`, `fitted.lcModel()`, `fittedTrajectories()`, `getCall.lcModel()`, `getLcMethod()`, `lcModel-class`, `metric()`, `model.frame.lcModel()`, `nClusters()`, `nIds()`, `nobs.lcModel()`, `plot-lcModel-method`, `plotClusterTrajectories()`, `plotFittedTrajectories()`, `postprob()`, `predict.lcModel()`, `predictAssignments()`, `predictForCluster()`, `predictPostprob()`, `qqPlot()`, `residuals.lcModel()`, `sigma.lcModel()`, `strip()`, `time.lcModel()`, `trajectoryAssignments()`

Examples

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

<code>idVariable</code>	<i>Extract the trajectory identifier variable</i>
-------------------------	---

Description

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

Usage

```
idVariable(object, ...)
```

S4 method for signature 'lcMethod'

```
idVariable(object, ...)
```

S4 method for signature 'lcModel'

```
idVariable(object)
```

S4 method for signature 'ANY'

```
idVariable(object)
```

Arguments

<code>object</code>	The object.
<code>...</code>	Not used.

Value

A nonempty string, as character.

See Also

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

Examples

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

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

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

Arguments

<code>.Object</code>	The newly allocated <code>lcMethod</code> object.
<code>...</code>	Other method arguments.

Examples

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

```
interface-metaMethods lcMetaMethod abstract class
```

Description

Virtual class for internal use. Do not use.

Usage

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

## S4 method for signature 'lcMetaMethod'
getLcMethod(object, ...)

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

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

## S4 method for signature 'lcMetaMethod'
idVariable(object, ...)

## S4 method for signature 'lcMetaMethod'
preFit(method, data, envir, verbose)

## S4 method for signature 'lcMetaMethod'
prepareData(method, data, verbose)

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

## S4 method for signature 'lcMetaMethod'
postFit(method, data, model, envir, verbose)

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

## S4 method for signature 'lcMetaMethod'
timeVariable(object, ...)

## S4 method for signature 'lcMetaMethod'
validate(method, data, envir = NULL, ...)

## S3 method for class 'lcMetaMethod'
update(object, ...)

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

## S4 method for signature 'lcFitConverged'
validate(method, data, envir = NULL, ...)

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

```
## S4 method for signature 'lcFitRep'
validate(method, data, envir = NULL, ...)
```

Arguments

method	The lcMethod object.
envir	The environment in which the lcMethod should be evaluated
object	The model.
...	Not used.
data	A data.frame representing the transformed training data.
verbose	A R.utils::Verbose object indicating the level of verbosity.
model	The lcModel object returned by fit() .

latrend

Cluster longitudinal data using the specified method

Description

An overview of the [latrend](#) package and its capabilities can be found [here](#).

The `latrend()` function fits a specified longitudinal cluster [method](#) to the given data comprising the trajectories.

This function runs all steps of the standardized [method estimation procedure](#), as implemented by the given `lcMethod` object. The result of this procedure is the estimated `lcModel`.

Usage

```
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 the <code>lcMethod</code> subclass to instantiate.
data	The data of the trajectories to which to estimate the method for. Any inputs supported by trajectories() can be used, including <code>data.frame</code> and <code>matrix</code> .
...	Any other arguments to update the <code>lcMethod</code> definition with.
envir	The environment in which to evaluate the method arguments via compose() . If the data argument is of type <code>call</code> then this environment is also used to evaluate the data argument.

`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 [preFit](#) step.

Value

A [lcModel](#) object representing the fitted solution.

See Also

Other longitudinal cluster fit functions: [latrendBatch\(\)](#), [latrendBoot\(\)](#), [latrendCV\(\)](#), [latrendRep\(\)](#)

Examples

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

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

model <- latrend(method, data = latrendData, nClusters = 3, seed = 1)
```

latrend-approaches *High-level approaches to longitudinal clustering*

Description

This page provides high-level guidelines on which methods are applicable to your dataset. Note that this is intended as a quick-start.

Recommended overview and comparison papers:

- (Den Teuling et al. 2021): A tutorial and overview on methods for longitudinal clustering.
- Den Teuling et al. (2021) compared KmL, MixTVEM, GBTM, GMM, and GCKM.
- Twisk and Hoekstra (2012) compared KmL, GCKM, LLCA, GBTM and GMM.
- Verboon and Pat-El (2022) compared the *kml*, *traj* and *lcmm* packages in R.
- Martin and von Oertzen (2015) compared KmL, LCA, and GMM.

Approaches

Disclaimer: The table below has been adapted from a pre-print of (Den Teuling et al. 2021).

Approach	Strengths
Cross-sectional clustering	Suitable for large datasets — Many available algorithms — Non-parametric cluster trajectory re
Distance-based clustering	Suitable for medium-sized datasets — Many distance metrics — Distance matrix only needs to l
Feature-based clustering	Suitable for large datasets — Configurable — Features only needs to be computed once — Com
Model-based clustering	Parametric cluster trajectory — Incorporate (domain) assumptions — Low sample size requirem

It is strongly encouraged to [evaluate and compare](#) several candidate methods in order to identify the most suitable method.

References

Den Teuling N, Pauws S, Heuvel Evd (2021). “Clustering of longitudinal data: A tutorial on a variety of approaches.” doi:10.48550/ARXIV.2111.05469, <https://arxiv.org/abs/2111.05469>.

Den Teuling NGP, Pauws SC, van den Heuvel ER (2021). “A comparison of methods for clustering longitudinal data with slowly changing trends.” *Communications in Statistics - Simulation and Computation*. doi:10.1080/03610918.2020.1861464.

Martin DP, von Oertzen T (2015). “Growth mixture models outperform simpler clustering algorithms when detecting longitudinal heterogeneity, even with small sample sizes.” *Struct. Equ. Model.*, **22**(2), 264–275. ISSN 1070-5511, doi:10.1080/10705511.2014.936340.

Twisk J, Hoekstra T (2012). “Classifying developmental trajectories over time should be done with great caution: A comparison between methods.” *Journal of Clinical Epidemiology*, **65**(10), 1078–1087. ISSN 0895-4356, doi:10.1016/j.jclinepi.2012.04.010.

Verboon P, Pat-EI R (2022). “Clustering Longitudinal Data Using R: A Monte Carlo Study.” *Methodology*, **18**(2), 144-163. doi:10.5964/meth.7143.

See Also

[latrend-methods](#) [latrend-estimation](#) [latrend-metrics](#)

latrend-data

Longitudinal dataset representation

Description

The [latrend estimation functions](#) expect univariate longitudinal data that can be represented in a data.frame with one row per trajectory observation:

- Trajectory identifier: numeric, character, or factor
- Observation time: numeric
- Observation value: numeric

In principle, any type of longitudinal data structure is supported, given that it can be transformed to the required `data.frame` format using the generic `trajectories` function. Support can be added by implementing the `trajectories` function for the respective signature. This means that users can implement their own data adapters as needed.

Included longitudinal datasets

The following datasets are included with the package:

- `latrendData`
- `PAP.adh`
- `PAP.adh1y`

latrend-estimation *Overview of lcMethod estimation functions*

Description

This page presents an overview of the different functions that are available for estimating one or more [longitudinal cluster methods](#). All functions are prefixed by "*latrend*".

latrend estimation functions

- `latrend()`: estimate a [method](#) on a [longitudinal dataset](#), returning the resulting [model](#).
- `latrendBatch()`: estimate multiple [methods](#) on multiple [longitudinal datasets](#), returning a [list of models](#).
- `latrendRep()`: repeatedly estimate a [method](#) on a [longitudinal dataset](#), returning a [list of models](#).
- `latrendBoot()`: repeatedly estimate a [method](#) on bootstrapped [longitudinal dataset](#), returning a [list of models](#).
- `latrendCV()`: repeatedly estimate a [method](#) using cross-validation on a [longitudinal dataset](#), returning a [list of models](#).

Parallel estimation

The functions involving repeated estimation support parallel computation. [See here](#).

See Also

[latrend-package lcMethod-estimation](#)

latrend-generics	<i>Generics used by latrend for different classes</i>
------------------	---

Description

Generics used by latrend for different classes

latrend-methods	<i>Supported methods for longitudinal clustering</i>
-----------------	--

Description

This page provides an overview of the currently supported methods for longitudinal clustering. For general recommendations on which method to apply to your dataset, [see here](#).

Supported methods

Method	Description
lcMethodAkmedoids	Anchored k -medoids (Adepeju et al. 2020)
lcMethodCrimCV	Group-based trajectory modeling of count data (Nielsen 2018)
lcMethodDtwclust	Methods for distance-based clustering, including dynamic time warping (Sardá-Espinosa 2019)
lcMethodFeature	Feature-based clustering
lcMethodFlexmix	Interface to the FlexMix framework (Grün and Leisch 2008)
lcMethodFlexmixGBTM	Group-based trajectory modeling
lcMethodFunFEM	Model-based clustering using funFEM (Bouveyron 2015)
lcMethodGCKM	Growth-curve modeling and k -means
lcMethodKML	Longitudinal k -means (Genolini et al. 2015)
lcMethodLcmmGBTM	Group-based trajectory modeling (Proust-Lima et al. 2017)
lcMethodLcmmGMM	Growth mixture modeling (Proust-Lima et al. 2017)
lcMethodLMKM	Feature-based clustering using linear regression and k -means
lcMethodMclustLLPA	Longitudinal latent profile analysis (Scrucca et al. 2016)
lcMethodMixAK_GLMM	Mixture of generalized linear mixed models
lcMethodMixtoolsGMM	Growth mixture modeling
lcMethodMixtoolsNPRM	Non-parametric repeated measures clustering (Benaglia et al. 2009)
lcMethodMixTVEM	Mixture of time-varying effects models
lcMethodRandom	Random partitioning
lcMethodStratify	Stratification rule

In addition, the functionality of any method can be extended via [meta methods](#). This is used for extending the estimation procedure of a method, such as [repeated fitting](#) and selecting the best result, or [fitting until convergence](#).

It is strongly encouraged to [evaluate and compare](#) several candidate methods in order to identify the most suitable method.

References

- Adepeju M, Langton S, Bannister J (2020). *akmedoids: Anchored Kmedoids for Longitudinal Data Clustering*. R package version 0.1.5, <https://CRAN.R-project.org/package=akmedoids>.
- Benaglia T, Chauveau D, Hunter DR, Young D (2009). “mixtools: An R Package for Analyzing Finite Mixture Models.” *Journal of Statistical Software*, **32**(6), 1–29. doi:10.18637/jss.v032.i06.
- Bouveyron C (2015). *funFEM: Clustering in the Discriminative Functional Subspace*. R package version 1.1, <https://CRAN.R-project.org/package=funFEM>.
- Genolini C, Alacoque X, Sentenac M, Arnaud C (2015). “kml and kml3d: R Packages to Cluster Longitudinal Data.” *Journal of Statistical Software*, **65**(4), 1–34. doi:10.18637/jss.v065.i04.
- Grün B, Leisch F (2008). “FlexMix Version 2: Finite Mixtures with Concomitant Variables and Varying and Constant Parameters.” *Journal of Statistical Software*, **28**(4), 1–35. doi:10.18637/jss.v028.i04.
- Nielsen JD (2018). *crimCV: Group-Based Modelling of Longitudinal Data*. R package version 0.9.6, <https://CRAN.R-project.org/package=crimCV>.
- Proust-Lima C, Philipps V, Liqueur B (2017). “Estimation of Extended Mixed Models Using Latent Classes and Latent Processes: The R Package lcmm.” *Journal of Statistical Software*, **78**(2), 1–56. doi:10.18637/jss.v078.i02.
- Sardá-Espinosa A (2019). “Time-Series Clustering in R Using the dtwclust Package.” *The R Journal*. doi:10.32614/RJ2019023.
- Scrucca L, Fop M, Murphy TB, Raftery AE (2016). “mclust 5: clustering, classification and density estimation using Gaussian finite mixture models.” *The R Journal*, **8**(1), 205–233.

See Also

[latrend-approaches](#) [latrend-estimation](#) [latrend-metrics](#)

Examples

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

latrend-metrics

Metrics

Description

The package supports a variety of metrics that help to evaluate and compare [estimated models](#).

- [Internal metrics](#): metrics that assess the adequacy of the model with respect to the data.

- **External metrics:** metrics that compare two models.

Users can implement new metrics through `defineInternalMetric()` and `defineExternalMetric()`. Custom-defined metrics are accessible using the same by-name mechanism as the other metrics.

Supported internal metrics

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

Supported external metrics

Metric name	Description
adjustedRand	Adjusted Rand index. Based on the Rand index, but adjusted for agreements occurring by chance. A score
CohensKappa	Cohen's kappa. A partitioning agreement metric correcting for random chance. A score of 1 indicates a p

F	F-score
F1	F1-score , also referred to as the Sørensen–Dice Coefficient , or Dice similarity coefficient
FolkesMallows	Fowlkes-Mallows index
Hubert	Hubert index
Jaccard	Jaccard index
jointEntropy	Joint entropy between model assignments
Kulczynski	Kulczynski index
MaximumMatch	Maximum match measure
McNemar	McNemar statistic
MeilaHeckerman	Meila-Heckerman measure
Mirkin	Mirkin metric
MI	Mutual information
NMI	Normalized mutual information
NSJ	Normalized version of splitJoin. The proportion of edits relative to the maximum changes (twice the number of clusters)
NVI	Normalized variation of information
Overlap	Overlap coefficient , also referred to as the Szymkiewicz–Simpson coefficient
PD	Partition difference
Phi	Phi coefficient .
precision	precision
Rand	Rand index
recall	recall
RogersTanimoto	Rogers-Tanimoto dissimilarity
RusselRao	Russell-Rao dissimilarity
SMC	Simple matching coefficient
splitJoin	total split-join index
splitJoin.ref	Split-join index of the first model to the second model. In other words, it is the edit-distance between the two models
SokalSneath1	Type-1 Sokal-Sneath dissimilarity
SokalSneath2	Type-2 Sokal-Sneath dissimilarity
VI	Variation of information
Wallace1	Type-1 Wallace criterion
Wallace2	Type-2 Wallace criterion
WMSSE	Weighted minimum sum of squared errors between cluster trajectories
WMMSE	Weighted minimum mean of squared errors between cluster trajectories
WMAE	Weighted minimum mean of absolute errors between cluster trajectories

See Also

[metric externalMetric](#)

Description

The model estimation functions support parallel computation through the use of the [foreach](#) mechanism. In order to make use of parallel execution, a parallel back-end must be registered.

Windows

On Windows, the [parallel-package](#) can be used to define parallel socket workers.

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

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

```
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:

```
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:

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

See Also

[latrendRep](#), [latrendBatch](#), [latrendBoot](#), [latrendCV](#)

Examples

```
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	<i>Cluster longitudinal data for a list of method specifications</i>
--------------	--

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 a seed number for the methods. Seeds are only set for methods without a seed argument or NULL seed.
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: [latrend\(\)](#), [latrendBoot\(\)](#), [latrendCV\(\)](#), [latrendRep\(\)](#)

Examples

```
data(latrendData)
refMethod <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
methods <- lcMethods(refMethod, nClusters = 1:2)
models <- latrendBatch(methods, data = latrendData)

# different dataset per method
models <- latrendBatch(
  methods,
  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 the lcMethod subclass to instantiate.
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 via compose() . If the data argument is of type call then this environment is also used to evaluate the data argument.
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: [latrend\(\)](#), [latrendBatch\(\)](#), [latrendCV\(\)](#), [latrendRep\(\)](#)

Other validation methods: [createTestDataFold\(\)](#), [createTestDataFolds\(\)](#), [createTrainDataFolds\(\)](#), [latrendCV\(\)](#), [lcModel-data-filters](#)

Examples

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

bootMAE <- metric(bootModels, name = "MAE")
mean(bootMAE)
sd(bootMAE)
```

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.

Usage

```
latrendCV(
  method,
  data,
  folds = 10,
  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 the lcMethod subclass to instantiate.
data	A <code>data.frame</code> .
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 via compose() . If the data argument is of type <code>call</code> then this environment is also used to evaluate the data argument.
verbose	The level of verbosity. Either an object of class <code>Verbose</code> (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 <code>c('info', 'fine', 'finest')</code> .

Value

A `lcModels` object of containing the folds training models.

See Also

Other longitudinal cluster fit functions: [latrend\(\)](#), [latrendBatch\(\)](#), [latrendBoot\(\)](#), [latrendRep\(\)](#)

Other validation methods: [createTestDataFold\(\)](#), [createTestDataFolds\(\)](#), [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, seed = 1)
```



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

latrendData	<i>Artificial longitudinal dataset comprising three classes</i>
-------------	---

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.

```
data(latrendData)
head(latrendData)
#>   Id      Time      Y      Class
#> 1  1 0.0000000 -1.08049205 Class 1
#> 2  1 0.2222222 -0.68024151 Class 1
#> 3  1 0.4444444 -0.65148373 Class 1
#> 4  1 0.6666667 -0.39115398 Class 1
#> 5  1 0.8888889 -0.19407876 Class 1
#> 6  1 1.1111111 -0.02991783 Class 1
```

Source

This dataset was generated using [generateLongData](#).

See Also

[latrend-data 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 the lcMethod subclass to instantiate.
data	The data of the trajectories to which to estimate the method for. Any inputs supported by trajectories() can be used, including <code>data.frame</code> and <code>matrix</code> .
.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 via compose() . If the data argument is of type <code>call</code> then this environment is also used to evaluate the data argument.

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: [latrend\(\)](#), [latrendBatch\(\)](#), [latrendBoot\(\)](#), [latrendCV\(\)](#)

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)
```

lcApproxModel-class *lcApproxModel class*

Description

approx models have defined cluster trajectories at fixed moments in time, which should be interpolated. For a correct implementation, `lcApproxModel` requires the extending class to implement `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

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.
newdata	A data.frame of trajectory data for which to compute trajectory assignments.
cluster	The cluster name (as character) to predict for.
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'.
approxFun	Function to interpolate between measurement moments, approx() by default.

lcFitMethods

*Method fit modifiers***Description**

A collection of special methods that adapt the fitting procedure of the underlying longitudinal cluster method.

NOTE: the underlying implementation is experimental and may change in the future.

Supported fit methods:

- `lcFitConverged`: Fit a method until a converged result is obtained.
- `lcFitRep`: Repeatedly fit a method and return the best result based on a given internal metric.
- `lcFitRepMin`: Repeatedly fit a method and return the best result that minimizes the given internal metric.
- `lcFitRepMax`: Repeatedly fit a method and return the best result that maximizes the given internal metric.

Usage

```
lcFitConverged(method, maxRep = Inf)
```

```
lcFitRep(method, rep = 10, metric, maximize)
```

```
lcFitRepMin(method, rep = 10, metric)
```

```
lcFitRepMax(method, rep = 10, metric)
```

Arguments

method	The lcMethod to use for fitting.
maxRep	The maximum number of fit attempts
rep	The number of fits
metric	The internal metric to assess the fit.
maximize	Whether to maximize the metric. Otherwise, it is minimized.

Details

Meta methods are immutable and cannot be updated after instantiation. Calling `update()` on a meta method is only used to update arguments of the underlying `lcMethod` object.

Examples

```
data(latrendData)
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time", nClusters = 2)
metaMethod <- lcFitConverged(method, maxRep = 10)
metaMethod
model <- latrend(metaMethod, latrendData)

data(latrendData)
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time", nClusters = 2)
repMethod <- lcFitRep(method, rep = 10, metric = "RSS", maximize = FALSE)
repMethod
model <- latrend(repMethod, latrendData)

minMethod <- lcFitRepMin(method, rep = 10, metric = "RSS")

maxMethod <- lcFitRepMax(method, rep = 10, metric = "ASW")
```

lcMethod-class

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.

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](#).

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](#) will be of use.

See Also

[environment](#)

Other lcMethod implementations: `getArgumentDefaults()`, `getArgumentExclusions()`, `lcMethodAkmedoids`, `lcMethodCrimCV`, `lcMethodDtwclust`, `lcMethodFeature`, `lcMethodFunFEM`, `lcMethodFunction`, `lcMethodGCKM`, `lcMethodKML`, `lcMethodLMKM`, `lcMethodLcmmGBTM`, `lcMethodLcmmGMM`, `lcMethodMcIustLLPA`, `lcMethodMixAK_GLMM`, `lcMethodMixtoolsGMM`, `lcMethodMixtoolsNPRM`, `lcMethodRandom`, `lcMethodStratify`

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

Examples

```
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)
```

lcMethod-estimation *Longitudinal cluster method (lcMethod) estimation procedure*

Description

Each longitudinal cluster method represented by a [lcMethod class](#) implements a series of standardized steps that produce the estimated method as its output. These steps, as part of the estimation procedure, are executed by the [latrend\(\)](#) function and other functions prefixed by "latrend" (e.g., [latrendRep\(\)](#), [latrendBoot\(\)](#), [latrendCV\(\)](#)).

Estimation procedure

The steps for estimating a [lcMethod](#) object are defined and executed as follows:

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

[lcMethod](#) [latrend](#)

Examples

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

lcMethodAkmedoids *Specify AKMedoids method*

Description

Specify AKMedoids method

Usage

```
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 <code>akmedoids::akclustr</code> . The following external arguments are ignored: <code>traj</code> , <code>id_field</code> , <code>k</code>

References

Adepeju M, Langton S, Bannister J (2020). *akmedoids: Anchored Kmedoids for Longitudinal Data Clustering*. R package version 0.1.5, <https://CRAN.R-project.org/package=akmedoids>.

See Also

Other lcMethod implementations: [getArgumentDefaults\(\)](#), [getArgumentExclusions\(\)](#), [lcMethod-class](#), [lcMethodCrimCV](#), [lcMethodDtwclust](#), [lcMethodFeature](#), [lcMethodFunFEM](#), [lcMethodFunction](#), [lcMethodGCKM](#), [lcMethodKML](#), [lcMethodLMKM](#), [lcMethodLcmmGBTM](#), [lcMethodLcmmGMM](#), [lcMethodMclustLLPA](#), [lcMethodMixAK_GLMM](#), [lcMethodMixtoolsGMM](#), [lcMethodMixtoolsNPRM](#), [lcMethodRandom](#), [lcMethodStratify](#)

Examples

```

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

```

lcMethodCrimCV	<i>Specify a zero-inflated repeated-measures GBTM method</i>
----------------	--

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 <code>crimCV::crimCV</code> . The following external arguments are ignored: <code>Dat</code> , <code>ng</code> .

References

Nielsen JD (2018). *crimCV: Group-Based Modelling of Longitudinal Data*. R package version 0.9.6, <https://CRAN.R-project.org/package=crimCV>.

See Also

Other lcMethod implementations: `getArgumentDefaults()`, `getArgumentExclusions()`, `lcMethod-class`, `lcMethodAkmedoids`, `lcMethodDtwclust`, `lcMethodFeature`, `lcMethodFunFEM`, `lcMethodFunction`, `lcMethodGCKM`, `lcMethodKML`, `lcMethodLMKM`, `lcMethodLcmmGBTM`, `lcMethodLcmmGMM`, `lcMethodMcLustLLPA`, `lcMethodMixAK_GLMM`, `lcMethodMixtoolsGMM`, `lcMethodMixtoolsNPRM`, `lcMethodRandom`, `lcMethodStratify`

Examples

```

# This example is not tested because crimCV sometimes fails
# to converge and throws the error "object 'Frtr' not found"
## Not run:
data(latrendData)
if (require("crimCV")) {
  method <- lcMethodCrimCV("Y", id = "Id", time = "Time", nClusters = 3, dpolyp = 1, init = 2)
  model <- latrend(method, data = subset(latrendData, Time > .5))

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

  data(T01adj)
  method <- lcMethodCrimCV(response = "Offenses", time = "Offense", id = "Subject",
    nClusters = 2, dpolyp = 1, init = 2)
  model <- latrend(method, data = T01adj[1:100, ])
}

## End(Not run)

```

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,
  ...
)

```

Arguments

response	The name of the response variable.
time	The name of the time variable.
id	The name of the trajectory identifier variable.
nClusters	Number of clusters.
...	Arguments passed to dtwclust::tsclust . The following arguments are ignored: series, k, trace.

References

Sardá-Espinosa A (2019). “Time-Series Clustering in R Using the dtwclust Package.” *The R Journal*. doi:10.32614/RJ2019023.

See Also

Other lcMethod implementations: [getArgumentDefaults\(\)](#), [getArgumentExclusions\(\)](#), [lcMethod-class](#), [lcMethodAkmedoids](#), [lcMethodCrimCV](#), [lcMethodFeature](#), [lcMethodFunFEM](#), [lcMethodFunction](#), [lcMethodGCKM](#), [lcMethodKML](#), [lcMethodLMKM](#), [lcMethodLcmmGBTM](#), [lcMethodLcmmGMM](#), [lcMethodMcLustLLPA](#), [lcMethodMixAK_GLMM](#), [lcMethodMixtoolsGMM](#), [lcMethodMixtoolsNPRM](#), [lcMethodRandom](#), [lcMethodStratify](#)

Examples

```
data(latrendData)

if (require("dtwclust")) {
  method <- lcMethodDtwclust("Y", id = "Id", time = "Time", nClusters = 3)
  model <- latrend(method, latrendData)
}
```

lcMethodFeature	<i>Feature-based clustering</i>
-----------------	---------------------------------

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 <code>function(method, data)</code> that computes the representation per strata, returned as a matrix. Alternatively, <code>representationStep</code> is a pre-computed representation matrix.

clusterStep	A function with signature <code>function(repdata)</code> that outputs a <code>lcModel</code> .
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

Other lcMethod implementations: [getArgumentDefaults\(\)](#), [getArgumentExclusions\(\)](#), [lcMethod-class](#), [lcMethodAkmedoids](#), [lcMethodCrimCV](#), [lcMethodDtwclust](#), [lcMethodFunFEM](#), [lcMethodFunction](#), [lcMethodGCKM](#), [lcMethodKML](#), [lcMethodLMKM](#), [lcMethodLcmmGBTM](#), [lcMethodLcmmGMM](#), [lcMethodMclustLLPA](#), [lcMethodMixAK_GLMM](#), [lcMethodMixtoolsGMM](#), [lcMethodMixtoolsNPRM](#), [lcMethodRandom](#), [lcMethodStratify](#)

lcMethodFlexmix	<i>Method interface to flexmix()</i>
-----------------	--------------------------------------

Description

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

Usage

```

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

```

Arguments

<code>formula</code>	A formula specifying the model.
<code>formula.mb</code>	A formula specifying the class membership model. By default, an intercept-only model is used.
<code>time</code>	The name of the time variable.
<code>id</code>	The name of the trajectory identifier variable.
<code>nClusters</code>	The number of clusters to estimate.
<code>...</code>	Arguments passed to <code>flexmix::flexmix</code> . The following arguments are ignored: <code>data</code> , <code>concomitant</code> , <code>k</code> .

References

Grün B, Leisch F (2008). “FlexMix Version 2: Finite Mixtures with Concomitant Variables and Varying and Constant Parameters.” *Journal of Statistical Software*, **28**(4), 1–35. doi:[10.18637/jss.v028.i04](https://doi.org/10.18637/jss.v028.i04).

See Also

Other lcMethod package interfaces: [lcMethodFlexmixGBTM](#)

Examples

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

lcMethodFlexmixGBTM *Group-based trajectory modeling using flexmix*

Description

Fits a GBTM based on the [flexmix::FLXMRglm](#) driver.

Usage

```
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.
nClusters	The number of clusters to estimate.
...	Arguments passed to flexmix::flexmix or flexmix::FLXMRglm . The following arguments are ignored: data, k, trace.

References

Grün B, Leisch F (2008). “FlexMix Version 2: Finite Mixtures with Concomitant Variables and Varying and Constant Parameters.” *Journal of Statistical Software*, **28**(4), 1–35. doi:10.18637/jss.v028.i04.

See Also

Other lcMethod package interfaces: [lcMethodFlexmix](#)

Examples

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

lcMethodFunction	<i>Specify a custom method based on a function</i>
------------------	--

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).
time	The name of the time variable.
id	The name of the trajectory identification variable.
name	The name of the method.

See Also

Other lcMethod implementations: [getArgumentDefaults\(\)](#), [getArgumentExclusions\(\)](#), [lcMethod-class](#), [lcMethodAkmedoids](#), [lcMethodCrimCV](#), [lcMethodDtwclust](#), [lcMethodFeature](#), [lcMethodFunFEM](#), [lcMethodGCKM](#), [lcMethodKML](#), [lcMethodLMKM](#), [lcMethodLcmmGBTM](#), [lcMethodLcmmGMM](#), [lcMethodMcLustLLPA](#), [lcMethodMixAK_GLMM](#), [lcMethodMixtoolsGMM](#), [lcMethodMixtoolsNPRM](#), [lcMethodRandom](#), [lcMethodStratify](#)

Examples

```
data(latrendData)
# Stratification based on the mean response level
clusfun <- function(data, response, id, time, ...) {
  clusters <- data.table::as.data.table(data)[, mean(Y) > 0, by = Id]$V1
  lcModelPartition(
    data = data,
    trajectoryAssignments = factor(
      clusters,
      levels = c(FALSE, TRUE),
      labels = c("Low", "High")
    ),
    response = response,
    time = time,
    id = id
  )
}
method <- lcMethodFunction(response = "Y", fun = clusfun, id = "Id", time = "Time")
model <- latrend(method, data = latrendData)
```

lcMethodFunFEM

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),
  ...
)
```


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 <code>funFEM::funFEM</code> . The following external arguments are ignored: <code>fd</code> , <code>K</code> , <code>disp</code> , <code>graph</code> .

References

Bouveyron C (2015). *funFEM: Clustering in the Discriminative Functional Subspace*. R package version 1.1, <https://CRAN.R-project.org/package=funFEM>.

See Also

Other lcMethod implementations: `getArgumentDefaults()`, `getArgumentExclusions()`, `lcMethod-class`, `lcMethodAkmedoids`, `lcMethodCrimCV`, `lcMethodDtwclust`, `lcMethodFeature`, `lcMethodFunction`, `lcMethodGCKM`, `lcMethodKML`, `lcMethodLMKM`, `lcMethodLcmmGBTM`, `lcMethodLcmmGMM`, `lcMethodMcLustLLPA`, `lcMethodMixAK_GLMM`, `lcMethodMixtoolsGMM`, `lcMethodMixtoolsNPRM`, `lcMethodRandom`, `lcMethodStratify`

Examples

```
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

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

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.
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 lme4::lmer . The following external arguments are ignored: data, centers, trace.

See Also

Other lcMethod implementations: [getArgumentDefaults\(\)](#), [getArgumentExclusions\(\)](#), [lcMethod-class](#), [lcMethodAkmedoids](#), [lcMethodCrimCV](#), [lcMethodDtwclust](#), [lcMethodFeature](#), [lcMethodFunFEM](#), [lcMethodFunction](#), [lcMethodKML](#), [lcMethodLMKM](#), [lcMethodLcmmGBTM](#), [lcMethodLcmmGMM](#), [lcMethodMclustLLPA](#), [lcMethodMixAK_GLMM](#), [lcMethodMixtoolsGMM](#), [lcMethodMixtoolsNPRM](#), [lcMethodRandom](#), [lcMethodStratify](#)

Examples

```
data(latrendData)

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

lcMethodKML	<i>Specify a longitudinal k-means (KML) method</i>
-------------	--

Description

Specify a longitudinal k-means (KML) method

Usage

```
lcMethodKML(
  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 kml::parALGO and kml::kml . The following external arguments are ignored: object, nbClusters, parAlgo, toPlot, saveFreq

References

Genolini C, Alacoque X, Sentenac M, Arnaud C (2015). “kml and kml3d: R Packages to Cluster Longitudinal Data.” *Journal of Statistical Software*, **65**(4), 1–34. doi:10.18637/jss.v065.i04.

See Also

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

Examples

```
data(latrendData)

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

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

Arguments

<code>fixed</code>	The fixed effects formula.
<code>mixture</code>	The mixture-specific effects formula. See lcmm::hlme for details.
<code>classmb</code>	The cluster membership formula for the multinomial logistic model. See lcmm::hlme for details.
<code>time</code>	The name of the time variable.
<code>id</code>	The name of the trajectory identifier variable. This replaces the subject argument of lcmm::hlme .
<code>nClusters</code>	The number of clusters to fit. This replaces the <code>ng</code> argument of lcmm::hlme .
<code>init</code>	Alternative for the <code>B</code> argument of lcmm::hlme , for initializing the <code>hlme</code> fitting procedure. This is only applicable for <code>nClusters > 1</code> . Options: <ul style="list-style-type: none"> "<code>lme.random</code>" (default): random initialization through a standard linear mixed model. Assigns a fitted standard linear mixed model enclosed in a call to <code>random()</code> to the <code>B</code> argument. "<code>lme</code>", fits a standard linear mixed model and passes this to the <code>B</code> argument. "<code>gridsearch</code>", a gridsearch is used with initialization from "<code>lme.random</code>", following the approach used by lcmm::gridsearch. To use this initialization, specify arguments <code>gridsearch.maxiter</code> (max number of iterations during search), <code>gridsearch.rep</code> (number of fits during search), and <code>gridsearch.parallel</code> (whether to enable parallel computation). NULL or "<code>default</code>", the default lcmm::hlme input for <code>B</code> is used. The argument is ignored if the <code>B</code> argument is specified, or <code>nClusters = 1</code> .
<code>...</code>	Arguments passed to lcmm::hlme . The following arguments are ignored: <code>data</code> , <code>fixed</code> , <code>random</code> , <code>mixture</code> , <code>subject</code> , <code>classmb</code> , <code>returndata</code> , <code>ng</code> , <code>verbose</code> , <code>subset</code> .

References

Proust-Lima C, Philipps V, Lique B (2017). “Estimation of Extended Mixed Models Using Latent Classes and Latent Processes: The R Package lcmm.” *Journal of Statistical Software*, **78**(2), 1–56. doi:10.18637/jss.v078.i02.

Proust-Lima C, Philipps V, Diakite A, Lique B (2019). *lcmm: Extended Mixed Models Using Latent Classes and Latent Processes*. R package version: 1.8.1, <https://cran.r-project.org/package=lcmm>.

See Also

Other lcMethod implementations: [getArgumentDefaults\(\)](#), [getArgumentExclusions\(\)](#), [lcMethod-class](#), [lcMethodAkmedoids](#), [lcMethodCrimCV](#), [lcMethodDtwclust](#), [lcMethodFeature](#), [lcMethodFunFEM](#), [lcMethodFunction](#), [lcMethodGCKM](#), [lcMethodKML](#), [lcMethodLMKM](#), [lcMethodLcmmGMM](#), [lcMethodMcLustLLPA](#), [lcMethodMixAK_GLMM](#), [lcMethodMixtoolsGMM](#), [lcMethodMixtoolsNPRM](#), [lcMethodRandom](#), [lcMethodStratify](#)

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

<code>fixed</code>	The fixed effects formula.
<code>mixture</code>	The mixture-specific effects formula. See lcmm::hlme for details.
<code>random</code>	The random effects formula. See lcmm::hlme for details.
<code>classmb</code>	The cluster membership formula for the multinomial logistic model. See lcmm::hlme for details.
<code>time</code>	The name of the time variable.
<code>id</code>	The name of the trajectory identifier variable. This replaces the subject argument of lcmm::hlme .
<code>init</code>	Alternative for the B argument of lcmm::hlme , for initializing the hlme fitting procedure. This is only applicable for <code>nClusters > 1</code> . Options: <ul style="list-style-type: none"> "lme.random" (default): random initialization through a standard linear mixed model. Assigns a fitted standard linear mixed model enclosed in a call to <code>random()</code> to the B argument. "lme", fits a standard linear mixed model and passes this to the B argument. "gridsearch", a gridsearch is used with initialization from "lme.random", following the approach used by lcmm::gridsearch. To use this initialization, specify arguments <code>gridsearch.maxiter</code> (max number of iterations during search), <code>gridsearch.rep</code> (number of fits during search), and <code>gridsearch.parallel</code> (whether to enable parallel computation). NULL or "default", the default lcmm::hlme input for B is used. The argument is ignored if the B argument is specified, or <code>nClusters = 1</code> .
<code>nClusters</code>	The number of clusters to fit. This replaces the <code>ng</code> argument of lcmm::hlme .
<code>...</code>	Arguments passed to lcmm::hlme . The following arguments are ignored: <code>data</code> , <code>fixed</code> , <code>random</code> , <code>mixture</code> , <code>subject</code> , <code>classmb</code> , <code>returndata</code> , <code>ng</code> , <code>verbose</code> , <code>subset</code> .

References

Proust-Lima C, Philipps V, Lique B (2017). "Estimation of Extended Mixed Models Using Latent Classes and Latent Processes: The R Package lcmm." *Journal of Statistical Software*, **78**(2), 1–56. [doi:10.18637/jss.v078.i02](https://doi.org/10.18637/jss.v078.i02).

Proust-Lima C, Philipps V, Diakite A, Lique B (2019). *lcmm: Extended Mixed Models Using Latent Classes and Latent Processes*. R package version: 1.8.1, <https://cran.r-project.org/package=lcmm>.

See Also

Other lcMethod implementations: `getArgumentDefaults()`, `getArgumentExclusions()`, `lcMethod-class`, `lcMethodAkmedoids`, `lcMethodCrimCV`, `lcMethodDtwclust`, `lcMethodFeature`, `lcMethodFunFEM`, `lcMethodFunction`, `lcMethodGCKM`, `lcMethodKML`, `lcMethodLMKM`, `lcMethodLcmmGBTM`, `lcMethodMclustLLPA`, `lcMethodMixAK_GLMM`, `lcMethodMixtoolsGMM`, `lcMethodMixtoolsNPRM`, `lcMethodRandom`, `lcMethodStratify`

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)

  # define method with gridsearch
  method <- lcMethodLcmmGMM(
    fixed = Y ~ Time,
    mixture = ~ Time,
    random = ~ 1,
    id = "Id",
    time = "Time",
    nClusters = 3,
    init = "gridsearch",
    gridsearch.maxiter = 10,
    gridsearch.rep = 50,
    gridsearch.parallel = TRUE
  )
}
```

 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 <code>stats::lm</code> . The following external arguments are ignored: <code>x</code> , <code>data</code> , <code>control</code> , <code>centers</code> , <code>trace</code> .

See Also

Other lcMethod implementations: [getArgumentDefaults\(\)](#), [getArgumentExclusions\(\)](#), [lcMethod-class](#), [lcMethodAkmedoids](#), [lcMethodCrimCV](#), [lcMethodDtwclust](#), [lcMethodFeature](#), [lcMethodFunFEM](#), [lcMethodFunction](#), [lcMethodGCKM](#), [lcMethodKML](#), [lcMethodLcmmGBTM](#), [lcMethodLcmmGMM](#), [lcMethodMclustLLPA](#), [lcMethodMixAK_GLMM](#), [lcMethodMixtoolsGMM](#), [lcMethodMixtoolsNPRM](#), [lcMethodRandom](#), [lcMethodStratify](#)

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 <code>mclust::Mclust</code> . The following external arguments are ignored: data, G, verbose.

References

Scrucca L, Fop M, Murphy TB, Raftery AE (2016). “mclust 5: clustering, classification and density estimation using Gaussian finite mixture models.” *The R Journal*, **8**(1), 205–233.

See Also

Other lcMethod implementations: [getArgumentDefaults\(\)](#), [getArgumentExclusions\(\)](#), [lcMethod-class](#), [lcMethodAkmedoids](#), [lcMethodCrimCV](#), [lcMethodDtwclust](#), [lcMethodFeature](#), [lcMethodFunFEM](#), [lcMethodFunction](#), [lcMethodGCKM](#), [lcMethodKML](#), [lcMethodLMKM](#), [lcMethodLcmmGBTM](#), [lcMethodLcmmGMM](#), [lcMethodMixAK_GLMM](#), [lcMethodMixtoolsGMM](#), [lcMethodMixtoolsNPRM](#), [lcMethodRandom](#), [lcMethodStratify](#)

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

<code>fixed</code>	A formula specifying the fixed effects of the model, including the response. Creates the <code>y</code> and <code>x</code> arguments for the call to mixAK::GLMM_MCMC .
<code>random</code>	A formula specifying the random effects of the model, including the random intercept. Creates the <code>z</code> and <code>random.intercept</code> arguments for the call to mixAK::GLMM_MCMC .
<code>time</code>	The name of the time variable.
<code>id</code>	The name of the trajectory identifier variable. This is used to generate the <code>id</code> vector argument for the call to mixAK::GLMM_MCMC .
<code>nClusters</code>	The number of clusters.
<code>...</code>	Arguments passed to mixAK::GLMM_MCMC . The following external arguments are ignored: <code>y</code> , <code>x</code> , <code>z</code> , <code>random.intercept</code> , <code>silent</code> .

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

Komárek A (2009). "A New R Package for Bayesian Estimation of Multivariate Normal Mixtures Allowing for Selection of the Number of Components and Interval-Censored Data." *Computational Statistics and Data Analysis*, **53**(12), 3932–3947. doi:10.1016/j.csda.2009.05.006.

See Also

Other `lcMethod` implementations: [getArgumentDefaults\(\)](#), [getArgumentExclusions\(\)](#), [lcMethod-class](#), [lcMethodAkmedoids](#), [lcMethodCrimCV](#), [lcMethodDtwclust](#), [lcMethodFeature](#), [lcMethodFunFEM](#), [lcMethodFunction](#), [lcMethodGCKM](#), [lcMethodKML](#), [lcMethodLMKM](#), [lcMethodLcmmGBTM](#), [lcMethodLcmmGMM](#), [lcMethodMclustLLPA](#), [lcMethodMixtoolsGMM](#), [lcMethodMixtoolsNPRM](#), [lcMethodRandom](#), [lcMethodStratify](#)

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,
  ...
)

```

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

Benaglia T, Chauveau D, Hunter DR, Young D (2009). “mixtools: An R Package for Analyzing Finite Mixture Models.” *Journal of Statistical Software*, **32**(6), 1–29. doi:[10.18637/jss.v032.i06](https://doi.org/10.18637/jss.v032.i06).

See Also

Other lcMethod implementations: [getArgumentDefaults\(\)](#), [getArgumentExclusions\(\)](#), [lcMethod-class](#), [lcMethodAkmedoids](#), [lcMethodCrimCV](#), [lcMethodDtwclust](#), [lcMethodFeature](#), [lcMethodFunFEM](#), [lcMethodFunction](#), [lcMethodGCKM](#), [lcMethodKML](#), [lcMethodLMKM](#), [lcMethodLcmmGBTM](#), [lcMethodLcmmGMM](#), [lcMethodMclustLLPA](#), [lcMethodMixAK_GLMM](#), [lcMethodMixtoolsNPRM](#), [lcMethodRandom](#), [lcMethodStratify](#)

Examples

```

data(latrendData)

if (require("mixtools")) {
  method <- lcMethodMixtoolsGMM(
    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

```

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

Benaglia T, Chauveau D, Hunter DR, Young D (2009). “mixtools: An R Package for Analyzing Finite Mixture Models.” *Journal of Statistical Software*, **32**(6), 1–29. doi:10.18637/jss.v032.i06.

See Also

Other lcMethod implementations: [getArgumentDefaults\(\)](#), [getArgumentExclusions\(\)](#), [lcMethod-class](#), [lcMethodAkmedoids](#), [lcMethodCrimCV](#), [lcMethodDtwclust](#), [lcMethodFeature](#), [lcMethodFunFEM](#), [lcMethodFunction](#), [lcMethodGCKM](#), [lcMethodKML](#), [lcMethodLMKM](#), [lcMethodLcmmGBTM](#), [lcMethodLcmmGMM](#), [lcMethodMclustLLPA](#), [lcMethodMixAK_GLMM](#), [lcMethodMixtoolsGMM](#), [lcMethodRandom](#), [lcMethodStratify](#)

Examples

```
data(latrendData)

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

lcMethodMixTVEM

Specify a MixTVEM

Description

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>

Dziak JJ, Li R, Tan X, Shiffman S, Shiyko MP (2015). “Modeling intensive longitudinal data with mixtures of nonparametric trajectories and time-varying effects.” *Psychological Methods*, **20**(4), 444–469. ISSN 1939-1463.

Examples

```
# this example only runs if you download and place MixTVEM.R in your wd
try({
  source("MixTVEM.R")
  method = lcMethodMixTVEM(
    Value ~ time(1) - 1,
    time = 'Assessment',
    id = "Id",
    nClusters = 3
  )
})
```

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

Frigyik BA, Kapila A, Gupta MR (2010). "Introduction to the Dirichlet distribution and related processes." Technical Report UWEETR-2010-0006, Department of Electrical Engineering, University of Washington.

See Also

Other lcMethod implementations: [getArgumentDefaults\(\)](#), [getArgumentExclusions\(\)](#), [lcMethod-class](#), [lcMethodAkmedoids](#), [lcMethodCrimCV](#), [lcMethodDtwclust](#), [lcMethodFeature](#), [lcMethodFunFEM](#), [lcMethodFunction](#), [lcMethodGCKM](#), [lcMethodKML](#), [lcMethodLMKM](#), [lcMethodLcmmGBTM](#), [lcMethodLcmmGMM](#), [lcMethodMclustLLPA](#), [lcMethodMixAK_GLMM](#), [lcMethodMixtoolsGMM](#), [lcMethodMixtoolsNPRM](#), [lcMethodStratify](#)

Examples

```
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

```
lcMethods(method, ..., 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
```

lcMethodStratify	<i>Specify a stratification method</i>
------------------	--

Description

Specify a stratification method

Usage

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

Other lcMethod implementations: [getArgumentDefaults\(\)](#), [getArgumentExclusions\(\)](#), [lcMethod-class](#), [lcMethodAkmedoids](#), [lcMethodCrimCV](#), [lcMethodDtwclust](#), [lcMethodFeature](#), [lcMethodFunFEM](#), [lcMethodFunction](#), [lcMethodGCKM](#), [lcMethodKML](#), [lcMethodLMKM](#), [lcMethodLcmmGBTM](#), [lcMethodLcmmGMM](#), [lcMethodMclustLLPA](#), [lcMethodMixAK_GLMM](#), [lcMethodMixtoolsGMM](#), [lcMethodMixtoolsNPRM](#), [lcMethodRandom](#)

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

Longitudinal cluster result (lcModel)

Description

A longitudinal cluster model (`[lcModel][lcModel-class]`) describes the clustered representation of a certain longitudinal dataset.

A `lcModel` is obtained by estimating a specified [longitudinal cluster method](#) on a [longitudinal dataset](#). The estimation is done via one of the [latrend estimation functions](#).

A longitudinal cluster result represents the dataset in terms of a partitioning of the trajectories into a number of clusters. The `trajectoryAssignments()` function outputs the most likely membership for the respective trajectories. Each cluster has a longitudinal representation, obtained via `clusterTrajectories()`, and can be plotted via `plotClusterTrajectories()`.

Functionality

Clusters and partitioning:

- `nClusters()`: The number of clusters this model represents.
- `clusterNames()`: The names of the clusters.
- `clusterSizes()`: The respective number of trajectories assigned to each cluster.
- `clusterProportions()`: The respective proportional size of each cluster.
- `trajectoryAssignments()`: The most likely cluster membership of each trajectory.
- `postprob()`: The posterior probability of each trajectory to each cluster.

Longitudinal cluster representation (i.e., trends):

- `clusterTrajectories()`: A data.frame containing the longitudinal representation of each cluster.
- `plotClusterTrajectories()`: Plots the longitudinal representation of each cluster.
- `fittedTrajectories()`: A data.frame containing the longitudinal representation of each trajectory. For many methods, this is the cluster center.
- `plotFittedTrajectories()`: Plot the trajectory representation.

Training data:

- `nIds()`: The number of trajectories used for estimation.
- `ids()`: A vector of identifiers of the trajectories that were used for estimation.
- `nobs()`: The number of observations used for estimation, across trajectories.
- `time()`: Moments in time on which observations are present.
- `trajectories()`: The trajectories that were used for estimation.
- `plotTrajectories()`: Plot the trajectories that were used for estimation.

Model evaluation:

- `summary()`: Obtain a summary of the model.
- `metric()`: Compute an internal metric.
- `externalMetric()`: Compute an external metric in relation to a second `IcModel`.
- `converged()`: Whether the estimation procedure converged.
- `estimationTime()`: Total time that was needed for the fitting steps.
- `sigma()`: Residual error scale.
- `qqPlot()`: QQ plot of the model residuals.

Model prediction:

- `predictForCluster()`: Cluster-specific prediction on new data. Not supported for all methods.
- `predictPostprob()`: Predict posterior probability for new data. Not supported for all methods.

- `predictAssignments()`: Predict cluster membership for new data. Not supported for all methods.

Other functionality:

- `getLcMethod()`: Get the [method specification](#) by which this model was estimated.
- `update()`: Retrain a model with altered method arguments.
- `strip()`: Removes non-essential (meta) data and environments from the model to facilitate efficient serialization.

See Also

[lcModel](#)

Examples

```
data(latrendData)
# define the method
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
# estimate the method, giving the model
model <- latrend(method, data = latrendData)

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

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 that 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 `lcModel` functions: `clusterNames()`, `clusterProportions()`, `clusterSizes()`, `clusterTrajectories()`, `coef.lcModel()`, `converged()`, `deviance.lcModel()`, `df.residual.lcModel()`, `estimationTime()`, `externalMetric()`, `fitted.lcModel()`, `fittedTrajectories()`, `getCall.lcModel()`, `getLcMethod()`, `ids()`, `metric()`, `model.frame.lcModel()`, `nClusters()`, `nIds()`, `nobs.lcModel()`, `plot-lcModel-method`, `plotClusterTrajectories()`, `plotFittedTrajectories()`, `postprob()`, `predict.lcModel()`, `predictAssignments()`, `predictForCluster()`, `predictPostprob()`, `qqPlot()`, `residuals.lcModel()`, `sigma.lcModel()`, `strip()`, `time.lcModel()`, `trajectoryAssignments()`

 lcModelPartition

Create a lcModel with pre-defined partitioning

Description

Represents an arbitrary partitioning of a set of trajectories. As such, this model has no predictive capabilities. The cluster trajectories are represented by the specified center function (mean by default).

Usage

```
lcModelPartition(
  data,
  response,
  trajectoryAssignments,
  nClusters = NA,
  clusterNames = character(),
  time = getOption("latrend.time"),
  id = getOption("latrend.id"),
  name = "part",
  center = meanNA,
  method = NULL,
  converged = TRUE,
  model = NULL,
  envir = parent.frame()
)
```

Arguments

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

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")
}
```

lcModels

Construct a list of lcModel objects

Description

[A general overview of the lcModels class can be found here.](#)

The `lcModels()` function creates a flat (named) list of `lcModel` objects. 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.

Functionality

- [Print](#) an argument summary for each of the models.
- [Convert](#) to a `data.frame` of method arguments.
- [Subset](#) the list.
- Compute an [internal metric](#) or [external metric](#).
- Obtain the best model according to [minimizing](#) or [maximizing](#) a [metric](#).
- Obtain the summed [estimation time](#).
- [Plot a metric](#) across a variable.
- [Plot the cluster trajectories](#).

See Also

Other lcModels functions: [as.lcModels\(\)](#), [lcModels-class](#), [max.lcModels\(\)](#), [min.lcModels\(\)](#), [plotMetric\(\)](#), [print.lcModels\(\)](#), [subset.lcModels\(\)](#)

Examples

```
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)

lcModels(defaults = c(lmkmModel, rngModel))
```

lcModels-class

lcModels: *a list of lcModel objects*

Description

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

A list of models is outputted from the repeated estimation functions such as [latrendRep\(\)](#), [latrendBatch\(\)](#), and [others](#). You can construct a list of models using the [lcModels\(\)](#) function.

Functionality

- [Print](#) an argument summary for each of the models.
- [Convert](#) to a data.frame of method arguments.
- [Subset](#) the list.
- Compute an [internal metric](#) or [external metric](#).
- Obtain the best model according to [minimizing](#) or [maximizing](#) a [metric](#).
- Obtain the summed [estimation time](#).
- [Plot a metric](#) across a variable.
- [Plot the cluster trajectories](#).

See Also

Other lcModels functions: [as.lcModels\(\)](#), [lcModels](#), [max.lcModels\(\)](#), [min.lcModels\(\)](#), [plotMetric\(\)](#), [print.lcModels\(\)](#), [subset.lcModels\(\)](#)

Examples

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

bestModel <- min(models, "MAE")
```

 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 metric](#)

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)
  logLik(gbtm)
}
```

max.lcModels

Select the lcModel with the highest metric value

Description

Select the lcModel with the highest metric value

Usage

```
## 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

Functionality

- [Print](#) an argument summary for each of the models.
- [Convert](#) to a data.frame of method arguments.
- [Subset](#) the list.
- Compute an [internal metric](#) or [external metric](#).
- Obtain the best model according to [minimizing](#) or [maximizing](#) a [metric](#).
- Obtain the summed [estimation time](#).
- [Plot a metric](#) across a variable.
- [Plot the cluster trajectories](#).

See Also

[min.lcModels externalMetric](#)

Other lcModels functions: [as.lcModels\(\)](#), [lcModels](#), [lcModels-class](#), [min.lcModels\(\)](#), [plotMetric\(\)](#), [print.lcModels\(\)](#), [subset.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)

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

```
metric(object, name = getOption("latrend.metric", c("WRSS", "APPA.mean")), ...)
```

```
## 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

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

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.

Supported internal metrics

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

Implementation

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

References

- Akaike H (1974). "A new look at the statistical model identification." *IEEE Transactions on Automatic Control*, **19**(6), 716-723. doi:10.1109/TAC.1974.1100705.
- Arbelaitz O, Gurrutxaga I, Muguerza J, Pérez JM, Perona I (2013). "An extensive comparative study

of cluster validity indices.” *Pattern recognition*, **46**(1), 243–256. ISSN 0031-3203, doi:10.1016/j.patcog.2012.07.021.

Biernacki C, Celeux G, Govaert G (2000). “Assessing a mixture model for clustering with the integrated completed likelihood.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **22**(7), 719–725. doi:10.1109/34.865189.

Bozdogan H (1987). “Model Selection and Akaike’s Information Criterion (AIC): The General Theory and Its Analytical Extensions.” *Psychometrika*, **52**, 345–370. doi:10.1007/BF02294361.

Dunn JC (1974). “Well-Separated Clusters and Optimal Fuzzy Partitions.” *Journal of Cybernetics*, **4**(1), 95–104. doi:10.1080/01969727408546059.

Henson JM, Reise SP, Kim KH (2007). “Detecting Mixtures From Structural Model Differences Using Latent Variable Mixture Modeling: A Comparison of Relative Model Fit Statistics.” *Structural Equation Modeling: A Multidisciplinary Journal*, **14**(2), 202–226. doi:10.1080/10705510709336744.

Mahalanobis PC (1936). “On the generalized distance in statistics.” *Proceedings of the National Institute of Sciences (Calcutta)*, **2**(1), 49–55.

McLachlan G, Peel D (2000). *Finite Mixture Models*. John Wiley & Sons, Inc. ISBN 9780471006268.

Muthén B (2004). “Latent variable analysis: Growth mixture modeling and related techniques for longitudinal data.” In *The SAGE Handbook of Quantitative Methodology for the Social Sciences*, 346–369. SAGE Publications, Inc. doi:10.4135/9781412986311.n19.

Nagin DS (2005). *Group-based modeling of development*. Harvard University Press. ISBN 9780674041318, doi:10.4159/9780674041318.

Ramaswamy V, Desarbo W, Reibstein D, Robinson W (1993). “An Empirical Pooling Approach for Estimating Marketing Mix Elasticities with PIMS Data.” *Marketing Science*, **12**(1), 103–124. doi:10.1287/mksc.12.1.103.

Rousseeuw PJ (1987). “Silhouettes: A graphical aid to the interpretation and validation of cluster analysis.” *Journal of Computational and Applied Mathematics*, **20**, 53–65. ISSN 0377-0427, doi:10.1016/03770427(87)901257.

Schwarz G (1978). “Estimating the Dimension of a Model.” *The Annals of Statistics*, **6**(2), 461–464.

Sclove SL (1987). “Application of model-selection criteria to some problems in multivariate analysis.” *Psychometrika*, **52**(3), 333–343. doi:10.1007/BF02294360.

van der Nest G, Lima Passos V, Candel MJ, van Breukelen GJ (2020). “An overview of mixture modelling for latent evolutions in longitudinal data: Modelling approaches, fit statistics and software.” *Advances in Life Course Research*, **43**, 100323. ISSN 1040-2608, doi:10.1016/j.alcr.2019.100323.

See Also

[externalMetric](#) [min.lcModels](#) [max.lcModels](#)

Other metric functions: [defineExternalMetric\(\)](#), [defineInternalMetric\(\)](#), [externalMetric\(\)](#), [getExternalMetricDefinition\(\)](#), [getExternalMetricNames\(\)](#), [getInternalMetricDefinition\(\)](#), [getInternalMetricNames\(\)](#)

Other lcModel functions: [clusterNames\(\)](#), [clusterProportions\(\)](#), [clusterSizes\(\)](#), [clusterTrajectories\(\)](#), [coef.lcModel\(\)](#), [converged\(\)](#), [deviance.lcModel\(\)](#), [df.residual.lcModel\(\)](#), [estimationTime\(\)](#), [externalMetric\(\)](#), [fitted.lcModel\(\)](#), [fittedTrajectories\(\)](#), [getCall.lcModel\(\)](#), [getLcMethod\(\)](#), [ids\(\)](#), [lcModel-class](#), [model.frame.lcModel\(\)](#), [nClusters\(\)](#), [nIds\(\)](#), [nobs.lcModel\(\)](#), [plot-lcModel-method](#), [plotClusterTrajectories\(\)](#), [plotFittedTrajectories\(\)](#), [postprob\(\)](#), [predict.lcModel\(\)](#), [predictAssignments\(\)](#), [predictForCluster\(\)](#), [predictPostprob\(\)](#), [qqPlot\(\)](#), [residuals.lcModel\(\)](#), [sigma.lcModel\(\)](#), [strip\(\)](#), [time.lcModel\(\)](#), [trajectoryAssignments\(\)](#)

Examples

```
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

```
## 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

Functionality

- [Print](#) an argument summary for each of the models.
- [Convert](#) to a data.frame of method arguments.
- [Subset](#) the list.
- Compute an [internal metric](#) or [external metric](#).
- Obtain the best model according to [minimizing](#) or [maximizing](#) a [metric](#).
- Obtain the summed [estimation time](#).
- [Plot a metric](#) across a variable.
- [Plot the cluster trajectories](#).

See Also

[max.lcModels externalMetric](#)

Other lcModels functions: [as.lcModels\(\)](#), [lcModels](#), [lcModels-class](#), [max.lcModels\(\)](#), [plotMetric\(\)](#), [print.lcModels\(\)](#), [subset.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)

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.

See Also

[model.frame.lcModel](#) [time.lcModel](#)

Examples

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

model.frame.lcModel	<i>Extract model training data</i>
---------------------	------------------------------------

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 lcModel functions: [clusterNames\(\)](#), [clusterProportions\(\)](#), [clusterSizes\(\)](#), [clusterTrajectories\(\)](#), [coef.lcModel\(\)](#), [converged\(\)](#), [deviance.lcModel\(\)](#), [df.residual.lcModel\(\)](#), [estimationTime\(\)](#), [externalMetric\(\)](#), [fitted.lcModel\(\)](#), [fittedTrajectories\(\)](#), [getCall.lcModel\(\)](#), [getLcMethod\(\)](#), [ids\(\)](#), [lcModel-class](#), [metric\(\)](#), [nClusters\(\)](#), [nIds\(\)](#), [nobs.lcModel\(\)](#), [plot-lcModel-method](#), [plotClusterTrajectories\(\)](#), [plotFittedTrajectories\(\)](#), [postprob\(\)](#), [predict.lcModel\(\)](#), [predictAssignments\(\)](#), [predictForCluster\(\)](#), [predictPostprob\(\)](#), [qqPlot\(\)](#), [residuals.lcModel\(\)](#), [sigma.lcModel\(\)](#), [strip\(\)](#), [time.lcModel\(\)](#), [trajectoryAssignments\(\)](#)

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

```
## 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 scalar integer.
A character vector of argument names.

See Also

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

Examples

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

nClusters	<i>Number of clusters</i>
-----------	---------------------------

Description

Get the number of clusters estimated by the given object.

Usage

```
nClusters(object, ...)

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

Arguments

object	The object
...	Not used.

Value

The number of clusters: a scalar numeric non-zero count.

See Also

[nIds](#) [nobs](#)

Other lcModel functions: [clusterNames\(\)](#), [clusterProportions\(\)](#), [clusterSizes\(\)](#), [clusterTrajectories\(\)](#), [coef.lcModel\(\)](#), [converged\(\)](#), [deviance.lcModel\(\)](#), [df.residual.lcModel\(\)](#), [estimationTime\(\)](#), [externalMetric\(\)](#), [fitted.lcModel\(\)](#), [fittedTrajectories\(\)](#), [getCall.lcModel\(\)](#), [getLcMethod\(\)](#), [ids\(\)](#), [lcModel-class](#), [metric\(\)](#), [model.frame.lcModel\(\)](#), [nIds\(\)](#), [nobs.lcModel\(\)](#), [plot-lcModel-method](#), [plotClusterTrajectories\(\)](#), [plotFittedTrajectories\(\)](#), [postprob\(\)](#), [predict.lcModel\(\)](#), [predictAssignments\(\)](#), [predictForCluster\(\)](#), [predictPostprob\(\)](#), [qqPlot\(\)](#), [residuals.lcModel\(\)](#), [sigma.lcModel\(\)](#), [strip\(\)](#), [time.lcModel\(\)](#), [trajectoryAssignments\(\)](#)

Examples

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

nIds	<i>Number of trajectories</i>
------	-------------------------------

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](#)

Other `lcModel` functions: [clusterNames\(\)](#), [clusterProportions\(\)](#), [clusterSizes\(\)](#), [clusterTrajectories\(\)](#), [coef.lcModel\(\)](#), [converged\(\)](#), [deviance.lcModel\(\)](#), [df.residual.lcModel\(\)](#), [estimationTime\(\)](#), [externalMetric\(\)](#), [fitted.lcModel\(\)](#), [fittedTrajectories\(\)](#), [getCall.lcModel\(\)](#), [getLcMethod\(\)](#), [ids\(\)](#), [lcModel-class](#), [metric\(\)](#), [model.frame.lcModel\(\)](#), [nClusters\(\)](#), [nobs.lcModel\(\)](#), [plot-lcModel-method](#), [plotClusterTrajectories\(\)](#), [plotFittedTrajectories\(\)](#), [postprob\(\)](#), [predict.lcModel\(\)](#), [predictAssignments\(\)](#), [predictForCluster\(\)](#), [predictPostprob\(\)](#), [qqPlot\(\)](#), [residuals.lcModel\(\)](#), [sigma.lcModel\(\)](#), [strip\(\)](#), [time.lcModel\(\)](#), [trajectoryAssignments\(\)](#)

Examples

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

nobs.lcModel	<i>Number of observations used for the lcModel fit</i>
--------------	--

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 lcModel functions: [clusterNames\(\)](#), [clusterProportions\(\)](#), [clusterSizes\(\)](#), [clusterTrajectories\(\)](#), [coef.lcModel\(\)](#), [converged\(\)](#), [deviance.lcModel\(\)](#), [df.residual.lcModel\(\)](#), [estimationTime\(\)](#), [externalMetric\(\)](#), [fitted.lcModel\(\)](#), [fittedTrajectories\(\)](#), [getCall.lcModel\(\)](#), [getLcMethod\(\)](#), [ids\(\)](#), [lcModel-class](#), [metric\(\)](#), [model.frame.lcModel\(\)](#), [nClusters\(\)](#), [nIds\(\)](#), [plot-lcModel-method](#), [plotClusterTrajectories\(\)](#), [plotFittedTrajectories\(\)](#), [postprob\(\)](#), [predict.lcModel\(\)](#), [predictAssignments\(\)](#), [predictForCluster\(\)](#), [predictPostprob\(\)](#), [qqPlot\(\)](#), [residuals.lcModel\(\)](#), [sigma.lcModel\(\)](#), [strip\(\)](#), [time.lcModel\(\)](#), [trajectoryAssignments\(\)](#)

Examples

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

OCC	<i>Odds of correct classification (OCC)</i>
-----	---

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

Nagin DS (2005). *Group-based modeling of development*. Harvard University Press. ISBN 9780674041318, doi:10.4159/9780674041318. Klijn SL, Weijnenberg MP, Lemmens P, van den Brandt PA, Passos VL (2017). “Introducing the fit-criteria assessment plot - A visualisation tool to assist class enumeration in group-based trajectory modelling.” *Statistical Methods in Medical Research*, **26**(5), 2424-2436. van der Nest G, Lima Passos V, Candel MJ, van Breukelen GJ (2020). “An overview of mixture modelling for latent evolutions in longitudinal data: Modelling approaches, fit statistics and software.” *Advances in Life Course Research*, **43**, 100323. ISSN 1040-2608, doi:10.1016/j.alcr.2019.100323.

See Also

[confusionMatrix APPA](#)

PAP.adh

Weekly Mean PAP Therapy Usage of OSA Patients in the First 3 Months

Description

A simulated longitudinal dataset comprising 301 patients with obstructive sleep apnea (OSA) during their first 91 days (13 weeks) of PAP therapy. The longitudinal patterns were inspired by the adherence patterns reported by Yi et al. (2022), interpolated to weekly hours of usage.

Usage

PAP.adh

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 integer: The patient identifier, where each level represents a simulated patient.

Week integer: The week number, starting from 1.

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.

Yi H, Dong X, Shang S, Zhang C, Xu L, Han F (2022). "Identifying longitudinal patterns of CPAP treatment in OSA using growth mixture modeling: Disease characteristics and psychological determinants." *Frontiers in Neurology*, **13**, 1063461. doi:10.3389/fneur.2022.1063461.

See Also

[latrend-data PAP.adh1y](#)

Examples

```
data(PAP.adh)

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

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

PAP.adh1y

Biweekly Mean PAP Therapy Adherence of OSA Patients over 1 Year

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

Usage

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.

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.

Aloia MS, Goodwin MS, Velicer WF, Arnedt JT, Zimmerman M, Skrekas J, Harris S, Millman RP (2008). "Time series analysis of treatment adherence patterns in individuals with obstructive sleep apnea." *Annals of Behavioral Medicine*, **36**(1), 44–53. ISSN 0883-6612, doi:10.1007/s12160008-90529.

See Also

[latrend-data](#)

Examples

```
data(PAP.adh1y)

if (require("ggplot2")) {
  plotTrajectories(PAP.adh1y, id = "Patient", time = "Biweek", response = "UsageHours")

  # plot according to cluster ground truth
  plotTrajectories(
    PAP.adh1y,
    id = "Patient",
    time = "Biweek",
    response = "UsageHours",
    cluster = "Group"
```



```

  )
}

```

plot-lcModel-method *Plot a lcModel*

Description

Plot a `lcModel` object. By default, this plots the cluster trajectories of the model, along with the trajectories used for estimation.

Usage

```

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

```

Arguments

<code>x</code>	The <code>lcModel</code> object.
<code>y</code>	Not used.
<code>...</code>	Arguments passed on to plotClusterTrajectories object The (cluster) trajectory data.

Value

A `ggplot` object.

See Also

[plotClusterTrajectories](#) [plotFittedTrajectories](#) [plotTrajectories](#) [ggplot2::ggplot](#)

Other `lcModel` functions: [clusterNames\(\)](#), [clusterProportions\(\)](#), [clusterSizes\(\)](#), [clusterTrajectories\(\)](#), [coef.lcModel\(\)](#), [converged\(\)](#), [deviance.lcModel\(\)](#), [df.residual.lcModel\(\)](#), [estimationTime\(\)](#), [externalMetric\(\)](#), [fitted.lcModel\(\)](#), [fittedTrajectories\(\)](#), [getCall.lcModel\(\)](#), [getLcMethod\(\)](#), [ids\(\)](#), [lcModel-class](#), [metric\(\)](#), [model.frame.lcModel\(\)](#), [nClusters\(\)](#), [nIds\(\)](#), [nobs.lcModel\(\)](#), [plotClusterTrajectories\(\)](#), [plotFittedTrajectories\(\)](#), [postprob\(\)](#), [predict.lcModel\(\)](#), [predictAssignments\(\)](#), [predictForCluster\(\)](#), [predictPostprob\(\)](#), [qqPlot\(\)](#), [residuals.lcModel\(\)](#), [sigma.lcModel\(\)](#), [strip\(\)](#), [time.lcModel\(\)](#), [trajectoryAssignments\(\)](#)

Examples

```

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

```
## S4 method for signature 'lcModels,ANY'
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 the cluster trajectories associated with the given model.

Usage

```
plotClusterTrajectories(object, ...)

## S4 method for signature 'data.frame'
plotClusterTrajectories(
  object,
  response,
  cluster = "Cluster",
  clusterOrder = character(),
  clusterLabeler = make.clusterPropLabels,
  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),
  clusterOrder = character(),
  clusterLabeler = make.clusterPropLabels,
  trajectories = FALSE,
  facet = !isFALSE(as.logical(trajectories[1])),
  ...
)

```

Arguments

object	The (cluster) trajectory data.
...	Additional arguments passed to clusterTrajectories .
response	The response variable name, see responseVariable .
cluster	The cluster assignment column
clusterOrder	Specify which clusters to plot and the order. Can be the cluster names or index. By default, all clusters are shown.
clusterLabeler	A function(clusterNames, clusterSizes) that generates plot labels for the clusters. By default the cluster name with the proportional size is shown, see make.clusterPropLabels .
time	The time variable name, see timeVariable .
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.
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	A numeric vector of the times at which to compute the cluster trajectories.

Value

A ggplot object.

See Also[clusterTrajectories](#)[plotTrajectories](#) plot

Other lcModel functions: [clusterNames\(\)](#), [clusterProportions\(\)](#), [clusterSizes\(\)](#), [clusterTrajectories\(\)](#), [coef.lcModel\(\)](#), [converged\(\)](#), [deviance.lcModel\(\)](#), [df.residual.lcModel\(\)](#), [estimationTime\(\)](#), [externalMetric\(\)](#), [fitted.lcModel\(\)](#), [fittedTrajectories\(\)](#), [getCall.lcModel\(\)](#), [getLcMethod\(\)](#), [ids\(\)](#), [lcModel-class](#), [metric\(\)](#), [model.frame.lcModel\(\)](#), [nClusters\(\)](#), [nIds\(\)](#), [nobs.lcModel\(\)](#), [plot-lcModel-method](#), [plotFittedTrajectories\(\)](#), [postprob\(\)](#), [predict.lcModel\(\)](#), [predictAssignments\(\)](#), [predictForCluster\(\)](#), [predictPostprob\(\)](#), [qqPlot\(\)](#), [residuals.lcModel\(\)](#), [sigma.lcModel\(\)](#), [strip\(\)](#), [time.lcModel\(\)](#), [trajectoryAssignments\(\)](#)

Examples

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

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

  # show cluster sizes in labels
  plotClusterTrajectories(model, clusterLabeler = make.clusterSizeLabels)

  # change cluster order
  plotClusterTrajectories(model, clusterOrder = c('B', 'C', 'A'))

  # sort clusters by decreasing size
  plotClusterTrajectories(model, clusterOrder = order(-clusterSizes(model)))

  # show only specific clusters
  plotClusterTrajectories(model, clusterOrder = c('B', 'C'))

  # 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")
}
```

```
plotFittedTrajectories
```

Plot the fitted trajectories

Description

Plot the fitted trajectories as represented by the given model

Usage

```
plotFittedTrajectories(object, ...)
```

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

Arguments

`object` The model.
`...` Arguments passed to [fittedTrajectories\(\)](#) and [plotTrajectories](#).

Value

A ggplot object.

See Also

[fittedTrajectories](#)

[plotClusterTrajectories](#) [plotTrajectories](#) [plot](#)

Other `lcModel` functions: [clusterNames\(\)](#), [clusterProportions\(\)](#), [clusterSizes\(\)](#), [clusterTrajectories\(\)](#), [coef.lcModel\(\)](#), [converged\(\)](#), [deviance.lcModel\(\)](#), [df.residual.lcModel\(\)](#), [estimationTime\(\)](#), [externalMetric\(\)](#), [fitted.lcModel\(\)](#), [fittedTrajectories\(\)](#), [getCall.lcModel\(\)](#), [getLcMethod\(\)](#), [ids\(\)](#), [lcModel-class](#), [metric\(\)](#), [model.frame.lcModel\(\)](#), [nClusters\(\)](#), [nIds\(\)](#), [nobs.lcModel\(\)](#), [plot-lcModel-method](#), [plotClusterTrajectories\(\)](#), [postprob\(\)](#), [predict.lcModel\(\)](#), [predictAssignments\(\)](#), [predictForCluster\(\)](#), [predictPostprob\(\)](#), [qqPlot\(\)](#), [residuals.lcModel\(\)](#), [sigma.lcModel\(\)](#), [strip\(\)](#), [time.lcModel\(\)](#), [trajectoryAssignments\(\)](#)

Examples

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

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

plotMetric

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())
```

Arguments

models	A lcModels or list of lcModel objects to compute and plot the metrics of.
name	The name(s) of the metric(s) to compute. If no names are given, the names specified in the <code>latrend.metric</code> option (WRSS, APPA, AIC, BIC) are used.
by	The argument name along which methods are plotted.
subset	Logical expression based on the lcModel method arguments, indicating which lcModel objects to keep.
group	The argument names to use for determining groups of different models. By default, all arguments are included. Specifying <code>group = character()</code> disables grouping. Specifying a single argument for grouping uses that specific column as the grouping column. In all other cases, groupings are represented by a number.

Value

ggplot2 object.

Functionality

- [Print](#) an argument summary for each of the models.
- [Convert](#) to a `data.frame` of method arguments.
- [Subset](#) the list.
- Compute an [internal metric](#) or [external metric](#).
- Obtain the best model according to [minimizing](#) or [maximizing](#) a [metric](#).
- Obtain the summed [estimation time](#).
- [Plot a metric](#) across a variable.
- [Plot the cluster trajectories](#).

See Also

Other lcModels functions: [as.lcModels\(\)](#), [lcModels](#), [lcModels-class](#), [max.lcModels\(\)](#), [min.lcModels\(\)](#), [print.lcModels\(\)](#), [subset.lcModels\(\)](#)

Examples

```

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

if (require("ggplot2")) {
  plotMetric(models, "WMAE")
}

if (require("ggplot2") && require("clusterCrit")) {
  plotMetric(models, c("WMAE", "Dunn"))
}

```

plotTrajectories	<i>Plot the data trajectories</i>
------------------	-----------------------------------

Description

Plots the output of [trajectories](#) for the given object.

Usage

```

plotTrajectories(object, ...)

## S4 method for signature 'data.frame'
plotTrajectories(
  object,
  response,
  cluster,
  time = getOption("latrend.time"),
  id = getOption("latrend.id"),
  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.
...	Additional arguments passed to trajectories .
response	Response variable character name or a call.

<code>cluster</code>	Whether to plot trajectories grouped by cluster (determined by the "Cluster" column). Alternatively, the name of the cluster column indicating trajectory cluster membership. If unspecified, trajectories are grouped if the object contains a "Cluster" column.
<code>time</code>	The time variable name, see timeVariable .
<code>id</code>	The identifier variable name, see idVariable .
<code>facet</code>	Whether to facet by cluster.

See Also[trajectories](#)[trajectories](#) [plotFittedTrajectories](#) [plotClusterTrajectories](#)[trajectories](#)**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"
  )
}
data(latrendData)
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time")
model <- latrend(method, latrendData, nClusters = 3)

if (require("ggplot2")) {
  plotTrajectories(model)
}

```

postFit	lcMethod <i>estimation step: logic for post-processing the fitted lcModel</i>
---------	---

Description

Note: this function should not be called directly, as it is part of the lcMethod [estimation procedure](#). For fitting an lcMethod object to a dataset, use the [latrend\(\)](#) function or [one of the other standard estimation functions](#).

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

```
postFit(method, data, model, envir, verbose, ...)
```

```
## 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.
...	Not used.

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":

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

Estimation procedure

The steps for estimating a `lcMethod` object are defined and executed as follows:

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.

postprob

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

```
postprob(object, ...)
```

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

Arguments

<code>object</code>	The model.
<code>...</code>	Not used.

Details

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

Value

An I-by-K numeric matrix with $I = nIds(object)$ and $K = nClusters(object)$.

Implementation

Classes extending `lcModel` should override this method.

```
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 `lcModel` functions: [clusterNames\(\)](#), [clusterProportions\(\)](#), [clusterSizes\(\)](#), [clusterTrajectories\(\)](#), [coef.lcModel\(\)](#), [converged\(\)](#), [deviance.lcModel\(\)](#), [df.residual.lcModel\(\)](#), [estimationTime\(\)](#), [externalMetric\(\)](#), [fitted.lcModel\(\)](#), [fittedTrajectories\(\)](#), [getCall.lcModel\(\)](#), [getLcMethod\(\)](#), [ids\(\)](#), [lcModel-class](#), [metric\(\)](#), [model.frame.lcModel\(\)](#), [nClusters\(\)](#), [nIds\(\)](#), [nobs.lcModel\(\)](#), [plot-lcModel-method](#), [plotClusterTrajectories\(\)](#), [plotFittedTrajectories\(\)](#), [predict.lcModel\(\)](#), [predictAssignments\(\)](#), [predictForCluster\(\)](#), [predictPostprob\(\)](#), [qqPlot\(\)](#), [residuals.lcModel\(\)](#), [sigma.lcModel\(\)](#), [strip\(\)](#), [time.lcModel\(\)](#), [trajectoryAssignments\(\)](#)

Examples

```
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",
    idiag = TRUE,
    nClusters = 2
  )
}
```

```

)
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

```
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

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

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](#)

Other `lcModel` functions: [clusterNames\(\)](#), [clusterProportions\(\)](#), [clusterSizes\(\)](#), [clusterTrajectories\(\)](#), [coef.lcModel\(\)](#), [converged\(\)](#), [deviance.lcModel\(\)](#), [df.residual.lcModel\(\)](#), [estimationTime\(\)](#), [externalMetric\(\)](#), [fitted.lcModel\(\)](#), [fittedTrajectories\(\)](#), [getCall.lcModel\(\)](#), [getLcMethod\(\)](#), [ids\(\)](#), [lcModel-class](#), [metric\(\)](#), [model.frame.lcModel\(\)](#), [nClusters\(\)](#), [nIds\(\)](#), [nobs.lcModel\(\)](#), [plot-lcModel-method](#), [plotClusterTrajectories\(\)](#), [plotFittedTrajectories\(\)](#), [postprob\(\)](#), [predictAssignments\(\)](#), [predictForCluster\(\)](#), [predictPostprob\(\)](#), [qqPlot\(\)](#), [residuals.lcModel\(\)](#), [sigma.lcModel\(\)](#), [strip\(\)](#), [time.lcModel\(\)](#), [trajectoryAssignments\(\)](#)

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

Predict the most likely cluster membership for each trajectory in the given data.

Usage

```
predictAssignments(object, newdata = NULL, ...)

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

Arguments

object	The model.
newdata	A data.frame of trajectory data for which to compute trajectory assignments.
...	Not used.
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.

Details

The default implementation uses [predictPostprob](#) to determine the cluster membership.

Value

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

See Also

[predictPostprob](#) [predict.lcModel](#)

Other `lcModel` functions: [clusterNames\(\)](#), [clusterProportions\(\)](#), [clusterSizes\(\)](#), [clusterTrajectories\(\)](#), [coef.lcModel\(\)](#), [converged\(\)](#), [deviance.lcModel\(\)](#), [df.residual.lcModel\(\)](#), [estimationTime\(\)](#), [externalMetric\(\)](#), [fitted.lcModel\(\)](#), [fittedTrajectories\(\)](#), [getCall.lcModel\(\)](#), [getLcMethod\(\)](#), [ids\(\)](#), [lcModel-class](#), [metric\(\)](#), [model.frame.lcModel\(\)](#), [nClusters\(\)](#), [nIds\(\)](#), [nobs.lcModel\(\)](#), [plot-lcModel-method](#), [plotClusterTrajectories\(\)](#), [plotFittedTrajectories\(\)](#), [postprob\(\)](#), [predict.lcModel\(\)](#), [predictForCluster\(\)](#), [predictPostprob\(\)](#), [qqPlot\(\)](#), [residuals.lcModel\(\)](#), [sigma.lcModel\(\)](#), [strip\(\)](#), [time.lcModel\(\)](#), [trajectoryAssignments\(\)](#)

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` *Predict trajectories conditional on cluster membership*

Description

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

For `lcModel` objects, 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

```
predictForCluster(object, newdata = NULL, cluster, ...)

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

Arguments

<code>object</code>	The model.
<code>newdata</code>	A <code>data.frame</code> of trajectory data for which to compute trajectory assignments.
<code>cluster</code>	The cluster name (as character) to predict for.
<code>...</code>	Arguments passed on to predict.lcModel

	<code>useCluster</code>	Whether to use the "Cluster" column in the <code>newdata</code> argument for computing predictions conditional on the respective cluster. For <code>useCluster = NA</code> (the default), the feature is enabled if <code>newdata</code> contains the "Cluster" column.
<code>what</code>		The distributional parameter to predict. By default, the mean response 'mu' is predicted. The cluster membership predictions can be obtained by specifying <code>what = 'mb'</code> .

Details

The default `predictForCluster(lcModel)` 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.

```
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 `lcModel` functions: [clusterNames\(\)](#), [clusterProportions\(\)](#), [clusterSizes\(\)](#), [clusterTrajectories\(\)](#), [coef.lcModel\(\)](#), [converged\(\)](#), [deviance.lcModel\(\)](#), [df.residual.lcModel\(\)](#), [estimationTime\(\)](#), [externalMetric\(\)](#), [fitted.lcModel\(\)](#), [fittedTrajectories\(\)](#), [getCall.lcModel\(\)](#), [getLcMethod\(\)](#), [ids\(\)](#), [lcModel-class](#), [metric\(\)](#), [model.frame.lcModel\(\)](#), [nClusters\(\)](#), [nIds\(\)](#), [nobs.lcModel\(\)](#), [plot-lcModel-method](#), [plotClusterTrajectories\(\)](#), [plotFittedTrajectories\(\)](#), [postprob\(\)](#), [predict.lcModel\(\)](#), [predictAssignments\(\)](#), [predictPostprob\(\)](#), [qqPlot\(\)](#), [residuals.lcModel\(\)](#), [sigma.lcModel\(\)](#), [strip\(\)](#), [time.lcModel\(\)](#), [trajectoryAssignments\(\)](#)

Examples

```
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	<i>Posterior probability for new data</i>
-----------------	---

Description

Returns the observation-specific posterior probabilities for the given data.

For `lcModel`: The default implementation returns a uniform probability matrix.

Usage

```

predictPostprob(object, newdata = NULL, ...)

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

```

Arguments

<code>object</code>	The model.
<code>newdata</code>	Optional <code>data.frame</code> for which to compute the posterior probability. If omitted, the model training data is used.
<code>...</code>	Additional arguments passed to postprob .

Value

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

Implementation

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

```

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

```

See Also[postprob](#)

Other lcModel functions: [clusterNames\(\)](#), [clusterProportions\(\)](#), [clusterSizes\(\)](#), [clusterTrajectories\(\)](#), [coef.lcModel\(\)](#), [converged\(\)](#), [deviance.lcModel\(\)](#), [df.residual.lcModel\(\)](#), [estimationTime\(\)](#), [externalMetric\(\)](#), [fitted.lcModel\(\)](#), [fittedTrajectories\(\)](#), [getCall.lcModel\(\)](#), [getLcMethod\(\)](#), [ids\(\)](#), [lcModel-class](#), [metric\(\)](#), [model.frame.lcModel\(\)](#), [nClusters\(\)](#), [nIds\(\)](#), [nobs.lcModel\(\)](#), [plot-lcModel-method](#), [plotClusterTrajectories\(\)](#), [plotFittedTrajectories\(\)](#), [postprob\(\)](#), [predict.lcModel\(\)](#), [predictAssignments\(\)](#), [predictForCluster\(\)](#), [qqPlot\(\)](#), [residuals.lcModel\(\)](#), [sigma.lcModel\(\)](#), [strip\(\)](#), [time.lcModel\(\)](#), [trajectoryAssignments\(\)](#)

`preFit``lcMethod` *estimation step: method preparation logic*

Description

Note: this function should not be called directly, as it is part of the `lcMethod` [estimation procedure](#). For fitting an `lcMethod` object to a dataset, use the [latrend\(\)](#) function or [one of the other standard estimation functions](#).

The `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 [fit\(\)](#) function to make use of the prepared work.

Usage

```
preFit(method, data, envir, verbose, ...)
```

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

Arguments

<code>method</code>	An object inheriting from <code>lcMethod</code> with all its arguments having been evaluated and finalized.
<code>data</code>	A <code>data.frame</code> representing the transformed training data.
<code>envir</code>	The environment containing additional data variables returned by prepareData() .
<code>verbose</code>	A R.utils::Verbose object indicating the level of verbosity.
<code>...</code>	Not used.

Value

The updated environment that will be passed to [fit\(\)](#).

Implementation

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

Estimation procedure

The steps for estimating a `lcMethod` object are defined and executed as follows:

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.

```
prepareData
```

```
lcMethod estimation step: logic for preparing the training data
```

Description

Note: this function should not be called directly, as it is part of the `lcMethod` [estimation procedure](#). For fitting an `lcMethod` object to a dataset, use the `latrend()` function or [one of the other standard estimation functions](#).

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

```
prepareData(method, data, verbose, ...)
```

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

Arguments

method	An object inheriting from <code>lcMethod</code> with all its arguments having been evaluated and finalized.
data	A <code>data.frame</code> representing the transformed training data.
verbose	A <code>R.utils::Verbose</code> object indicating the level of verbosity.
...	Not used.

Value

An environment.

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.

```
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)
})
```

Estimation procedure

The steps for estimating a `lcMethod` object are defined and executed as follows:

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

```
## 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

```
## 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.

Functionality

- [Print](#) an argument summary for each of the models.
- [Convert](#) to a `data.frame` of method arguments.
- [Subset](#) the list.
- Compute an [internal metric](#) or [external metric](#).
- Obtain the best model according to [minimizing](#) or [maximizing](#) a [metric](#).
- Obtain the summed [estimation time](#).
- [Plot a metric](#) across a variable.
- [Plot the cluster trajectories](#).

See Also

Other `lcModels` functions: [as.lcModels\(\)](#), [lcModels](#), [lcModels-class](#), [max.lcModels\(\)](#), [min.lcModels\(\)](#), [plotMetric\(\)](#), [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

```
qqPlot(model, byCluster = FALSE, ...)
```

Arguments

<code>model</code>	<code>lcModel</code>
<code>byCluster</code>	Whether to plot the Q-Q line per cluster
<code>...</code>	Additional arguments passed to residuals.lcModel , qqplotr::geom_qq_band() , qqplotr::stat_qq_line() , and qqplotr::stat_qq_point() .

Value

A `ggplot` object.

See Also

[residuals.lcModel](#) [metric](#) [plotClusterTrajectories](#)

Other lcModel functions: [clusterNames\(\)](#), [clusterProportions\(\)](#), [clusterSizes\(\)](#), [clusterTrajectories\(\)](#), [coef.lcModel\(\)](#), [converged\(\)](#), [deviance.lcModel\(\)](#), [df.residual.lcModel\(\)](#), [estimationTime\(\)](#), [externalMetric\(\)](#), [fitted.lcModel\(\)](#), [fittedTrajectories\(\)](#), [getCall.lcModel\(\)](#), [getLcMethod\(\)](#), [ids\(\)](#), [lcModel-class](#), [metric\(\)](#), [model.frame.lcModel\(\)](#), [nClusters\(\)](#), [nIds\(\)](#), [nobs.lcModel\(\)](#), [plot-lcModel-method](#), [plotClusterTrajectories\(\)](#), [plotFittedTrajectories\(\)](#), [postprob\(\)](#), [predict.lcModel\(\)](#), [predictAssignments\(\)](#), [predictForCluster\(\)](#), [predictPostprob\(\)](#), [residuals.lcModel\(\)](#), [sigma.lcModel\(\)](#), [strip\(\)](#), [time.lcModel\(\)](#), [trajectoryAssignments\(\)](#)

Examples

```
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

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

Arguments

<code>object</code>	The lcModel object.
<code>...</code>	Additional arguments.
<code>clusters</code>	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 lcModel functions: [clusterNames\(\)](#), [clusterProportions\(\)](#), [clusterSizes\(\)](#), [clusterTrajectories\(\)](#), [coef.lcModel\(\)](#), [converged\(\)](#), [deviance.lcModel\(\)](#), [df.residual.lcModel\(\)](#), [estimationTime\(\)](#), [externalMetric\(\)](#), [fitted.lcModel\(\)](#), [fittedTrajectories\(\)](#), [getCall.lcModel\(\)](#), [getLcMethod\(\)](#), [ids\(\)](#), [lcModel-class](#), [metric\(\)](#), [model.frame.lcModel\(\)](#), [nClusters\(\)](#), [nIds\(\)](#), [nobs.lcModel\(\)](#), [plot-lcModel-method](#), [plotClusterTrajectories\(\)](#), [plotFittedTrajectories\(\)](#), [postprob\(\)](#), [predict.lcModel\(\)](#), [predictAssignments\(\)](#), [predictForCluster\(\)](#), [predictPostprob\(\)](#), [qqPlot\(\)](#), [sigma.lcModel\(\)](#), [strip\(\)](#), [time.lcModel\(\)](#), [trajectoryAssignments\(\)](#)

responseVariable	<i>Extract response variable</i>
------------------	----------------------------------

Description

Extracts the response variable from the given object.

Get the response variable, i.e., the dependent variable.

Usage

```
responseVariable(object, ...)

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

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

Arguments

object	The object.
...	Not used.

Details

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

Value

A nonempty string, as character.

See Also

Other variables: [idVariable\(\)](#), [timeVariable\(\)](#)

Examples

```

method <- lcMethodLMKM(Y ~ Time)
responseVariable(method) # "Y"
data(latrendData)
method <- lcMethodRandom("Y", 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

[coef.lcModel](#) [metric](#)

Other lcModel functions: [clusterNames\(\)](#), [clusterProportions\(\)](#), [clusterSizes\(\)](#), [clusterTrajectories\(\)](#), [coef.lcModel\(\)](#), [converged\(\)](#), [deviance.lcModel\(\)](#), [df.residual.lcModel\(\)](#), [estimationTime\(\)](#), [externalMetric\(\)](#), [fitted.lcModel\(\)](#), [fittedTrajectories\(\)](#), [getCall.lcModel\(\)](#), [getLcMethod\(\)](#), [ids\(\)](#), [lcModel-class](#), [metric\(\)](#), [model.frame.lcModel\(\)](#), [nClusters\(\)](#), [nIds\(\)](#), [nobs.lcModel\(\)](#), [plot-lcModel-method](#), [plotClusterTrajectories\(\)](#), [plotFittedTrajectories\(\)](#), [postprob\(\)](#), [predict.lcModel\(\)](#), [predictAssignments\(\)](#), [predictForCluster\(\)](#), [predictPostprob\(\)](#), [qqPlot\(\)](#), [residuals.lcModel\(\)](#), [strip\(\)](#), [time.lcModel\(\)](#), [trajectoryAssignments\(\)](#)

strip

*Reduce the memory footprint of an object for serialization***Description**

Reduce the (serialized) memory footprint of an object.

Usage

```
strip(object, ...)

## 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 model.
...	Not used.
classes	The object classes for which to remove their assigned environment. By default, only environments from formula are removed.

Details

Serializing references to environments results in the serialization of the object together with any associated environments and references. This method removes those environments and references, greatly reducing the serialized object size.

Value

The stripped (i.e., updated) object.

Implementation

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

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

See Also

Other lcModel functions: `clusterNames()`, `clusterProportions()`, `clusterSizes()`, `clusterTrajectories()`, `coef.lcModel()`, `converged()`, `deviance.lcModel()`, `df.residual.lcModel()`, `estimationTime()`, `externalMetric()`, `fitted.lcModel()`, `fittedTrajectories()`, `getCall.lcModel()`, `getLcMethod()`, `ids()`, `lcModel-class`, `metric()`, `model.frame.lcModel()`, `nClusters()`, `nIds()`, `nobs.lcModel()`, `plot-lcModel-method`, `plotClusterTrajectories()`, `plotFittedTrajectories()`, `postprob()`, `predict.lcModel()`, `predictAssignments()`, `predictForCluster()`, `predictPostprob()`, `qqPlot()`, `residuals.lcModel()`, `sigma.lcModel()`, `time.lcModel()`, `trajectoryAssignments()`

Examples

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

subset.lcModels	<i>Subsetting a lcModels list based on method arguments</i>
-----------------	---

Description

Subsetting a lcModels list based on method arguments

Usage

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

Arguments

x	The lcModels or list of lcModel to be subsetted.
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 lcModels list with the subset of lcModel objects.

Functionality

- [Print](#) an argument summary for each of the models.
- [Convert](#) to a data.frame of method arguments.
- [Subset](#) the list.
- Compute an [internal metric](#) or [external metric](#).

- Obtain the best model according to [minimizing](#) or [maximizing](#) a [metric](#).
- Obtain the summed [estimation time](#).
- [Plot a metric](#) across a variable.
- [Plot the cluster trajectories](#).

See Also

Other lcModels functions: [as.lcModels\(\)](#), [lcModels](#), [lcModels-class](#), [max.lcModels\(\)](#), [min.lcModels\(\)](#), [plotMetric\(\)](#), [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')
```

summary.lcModel

Summarize a lcModel

Description

Extracts all relevant information from the underlying model into a list

Usage

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

Arguments

`object` The lcModel object.
`...` Additional arguments.

test.latrend	<i>Test the implementation of an lcMethod and associated lcModel subclasses</i>
--------------	---

Description

Test a lcMethod subclass implementation and its resulting lcModel implementation.

Usage

```
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.

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", tests = c("method", "basic"), clusterRecovery = "skip")
```

time.lcModel	<i>Sampling times of a lcModel</i>
--------------	------------------------------------

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 lcModel functions: [clusterNames\(\)](#), [clusterProportions\(\)](#), [clusterSizes\(\)](#), [clusterTrajectories\(\)](#), [coef.lcModel\(\)](#), [converged\(\)](#), [deviance.lcModel\(\)](#), [df.residual.lcModel\(\)](#), [estimationTime\(\)](#), [externalMetric\(\)](#), [fitted.lcModel\(\)](#), [fittedTrajectories\(\)](#), [getCall.lcModel\(\)](#), [getLcMethod\(\)](#), [ids\(\)](#), [lcModel-class](#), [metric\(\)](#), [model.frame.lcModel\(\)](#), [nClusters\(\)](#), [nIds\(\)](#), [nobs.lcModel\(\)](#), [plot-lcModel-method](#), [plotClusterTrajectories\(\)](#), [plotFittedTrajectories\(\)](#), [postprob\(\)](#), [predict.lcModel\(\)](#), [predictAssignments\(\)](#), [predictForCluster\(\)](#), [predictPostprob\(\)](#), [qqPlot\(\)](#), [residuals.lcModel\(\)](#), [sigma.lcModel\(\)](#), [strip\(\)](#), [trajectoryAssignments\(\)](#)

timeVariable	<i>Extract the time variable</i>
--------------	----------------------------------

Description

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

Usage

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

Arguments

object	The object.
...	Not used.

Value

The time variable name, as character.

See Also

Other variables: [idVariable\(\)](#), [responseVariable\(\)](#)

Examples

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

trajectories

Get the trajectories

Description

Transform or extract the trajectories from the given object to a standardized format.

Trajectories are ordered by Id and observation time.

For estimated models; get the trajectories used for estimation, along with the cluster membership. This data can be used for plotting or post-hoc analysis.

Usage

```
trajectories(
  object,
  id = idVariable(object),
  time = timeVariable(object),
  response = responseVariable(object),
  cluster = "Cluster",
  ...
)

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

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

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

## S4 method for signature 'lcModel'
trajectories(
```



```

    object,
    id = idVariable(object),
    time = timeVariable(object),
    response = responseVariable(object),
    cluster = "Cluster",
    ...
  )

```

Arguments

object	The data or model or extract the trajectories from.
id	The identifier variable name, see idVariable .
time	The time variable name, see timeVariable .
response	The response variable name, see responseVariable .
cluster	Experimental feature for data.frame input: a vector of cluster membership per id
...	Arguments passed to trajectoryAssignments for generating the Cluster column.
envir	The environment used to evaluate the data object in (e.g., in case object is of type call).

Details

The standardized data format is for method estimation by [latrend](#), and for plotting functions.

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, response and cluster name arguments.

See Also

[plotTrajectories](#) [latrend](#)

Examples

```

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

```

trajectoryAssignments *Get the cluster membership of each trajectory*

Description

Get the cluster membership of each trajectory associated with the given model.

For `lcModel`: 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.

Usage

```
trajectoryAssignments(object, ...)

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

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

Arguments

<code>object</code>	The model.
<code>...</code>	Any additional arguments passed to the strategy function.
<code>strategy</code>	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.
<code>clusterNames</code>	Optional character vector with the cluster names. If <code>clusterNames = NULL</code> , <code>make.clusterNames()</code> is used.

Details

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

Value

A factor vector indicating the cluster membership for each trajectory.

See Also

[postprob](#) [clusterSizes](#) [predictAssignments](#)

Other `lcModel` functions: [clusterNames\(\)](#), [clusterProportions\(\)](#), [clusterSizes\(\)](#), [clusterTrajectories\(\)](#), [coef.lcModel\(\)](#), [converged\(\)](#), [deviance.lcModel\(\)](#), [df.residual.lcModel\(\)](#), [estimationTime\(\)](#), [externalMetric\(\)](#), [fitted.lcModel\(\)](#), [fittedTrajectories\(\)](#), [getCall.lcModel\(\)](#), [getLcMethod\(\)](#), [ids\(\)](#), [lcModel-class](#), [metric\(\)](#), [model.frame.lcModel\(\)](#), [nClusters\(\)](#), [nIds\(\)](#), [nobs.lcModel\(\)](#), [plot-lcModel-method](#), [plotClusterTrajectories\(\)](#), [plotFittedTrajectories\(\)](#), [postprob\(\)](#), [predict.lcModel\(\)](#), [predictAssignments\(\)](#), [predictForCluster\(\)](#), [predictPostprob\(\)](#), [qqPlot\(\)](#), [residuals.lcModel\(\)](#), [sigma.lcModel\(\)](#), [strip\(\)](#), [time.lcModel\(\)](#)

Examples

```
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)
```

transformFitted	<i>Helper function for custom lcModel classes implementing fitted.lcModel()</i>
-----------------	---

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 `nobs(object) * nClusters(object)` rows.

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

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

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

Usage

```

transformFitted(pred, model, clusters)

## S4 method for signature 'NULL,lcModel'
transformFitted(pred, model, clusters = NULL)

## S4 method for signature 'matrix,lcModel'
transformFitted(pred, model, clusters = NULL)

## S4 method for signature 'list,lcModel'
transformFitted(pred, model, clusters = NULL)

## S4 method for signature 'data.frame,lcModel'
transformFitted(pred, model, clusters = NULL)

```

Arguments

<code>pred</code>	The cluster-specific predictions for each observation
<code>model</code>	The <code>lcModel</code> by which the prediction was made.
<code>clusters</code>	The trajectory cluster assignment per observation. Optional.

Value

If the `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 `fitted.lcModel()` for your own `lcModel` class would have the following format:

```

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

```

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

transformPredict	<i>Helper function for custom lcModel classes implementing predict.lcModel()</i>
------------------	--

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, $N = \text{nrow}(\text{newdata})$ and $K = \text{nClusters}(\text{object})$.

`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

```
transformPredict(pred, model, newdata)

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

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

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

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

Arguments

<code>pred</code>	The (per-cluster) predictions for <code>newdata</code> .
<code>model</code>	The <code>lcModel</code> for which the prediction was made.
<code>newdata</code>	A <code>data.frame</code> 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.

```
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

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

<code>data</code>	The matrix containing a trajectory on each row.
<code>response</code>	The response column name.
<code>id</code>	The id column name.
<code>time</code>	The time column name.
<code>ids</code>	A vector specifying the id names. Should match the number of rows of data.
<code>times</code>	A numeric vector specifying the times of the measurements. Should match the number of columns of data.
<code>as.data.table</code>	Whether to return the result as a <code>data.table</code> , or a <code>data.frame</code> 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

```
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

<code>data</code>	The matrix containing a trajectory on each row.
<code>response</code>	The response column name.
<code>id</code>	The id column name.
<code>time</code>	The time column name.
<code>fill</code>	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.

See Also

[tsframe](#)

update.lcMethod	<i>Update a method specification</i>
-----------------	--------------------------------------

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.lcMethod\(\),as.data.frame.lcMethods\(\),as.lcMethods\(\),as.list.lcMethod\(\),evaluate.lcMethod\(\),formula.lcMethod\(\),lcMethod-class,names,lcMethod-method](#)

Examples

```
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

```
## 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 the lcMethod subclass to instantiate.
data	The data of the trajectories to which to estimate the method for. Any inputs supported by trajectories() can be used, including <code>data.frame</code> and <code>matrix</code> .
envir	The environment in which to evaluate the method arguments via compose() . If the data argument is of type <code>call</code> then this environment is also used to evaluate the data argument.
verbose	The level of verbosity. Either an object of class <code>Verbose</code> (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 <code>c('info', 'fine', 'finest')</code> .

Value

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

See Also

[latrend](#) [getCall](#)

Examples

```
data(latrendData)
method <- lcMethodLMKM(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	<i>lcMethod estimation step: method argument validation logic</i>
----------	---

Description

Note: this function should not be called directly, as it is part of the `lcMethod` [estimation procedure](#). For fitting an `lcMethod` object to a dataset, use the `latrend()` function or [one of the other standard estimation functions](#).

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

```
validate(method, data, envir, ...)

## S4 method for signature 'lcMethod'
validate(method, data, envir = NULL, ...)
```

Arguments

<code>method</code>	An object inheriting from <code>lcMethod</code> with all its arguments having been evaluated and finalized.
<code>data</code>	A <code>data.frame</code> representing the transformed training data.
<code>envir</code>	The environment in which the <code>lcMethod</code> should be evaluated
<code>...</code>	Not used.

Value

Either `TRUE` if all validation checks passed, or a scalar character containing a description of the failed validation checks.

Implementation

An example implementation checking for the existence of specific arguments and type:

```
library(assertthat)
setMethod("validate", "lcMethodExample", function(method, data, envir = NULL, ...) {
  validate_that(
```

```

    hasName(method, "myArgument"),
    hasName(method, "anotherArgument"),
    is.numeric(method$myArgument)
  )
})

```

Estimation procedure

The steps for estimating a `lcMethod` object are defined and executed as follows:

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.lcMethod\(\)](#), [as.data.frame.lcMethods\(\)](#), [as.lcMethods\(\)](#), [as.list.lcMethod\(\)](#), [evaluate.lcMethod\(\)](#), [formula.lcMethod\(\)](#), [lcMethod-class](#), [names.lcMethod-method](#), [update.lcMethod\(\)](#)

Examples

```
method <- lcMethodLMKM(Y ~ Time, id = "Id", time = "Time", nClusters = 3)
method$nClusters # 3
m = lcMethodLMKM(Y ~ Time, id = "Id", time = "Time", nClusters = 5)
m[["nClusters"]] # 5

k = 2
m = lcMethodLMKM(Y ~ Time, id = "Id", time = "Time", nClusters = k)
m[["nClusters", eval=FALSE]] # k
```

Index

* datasets

- latrendData, 65
- PAP.adh, 118
- PAP.adh1y, 119

* lcMethod functions

- [[,lcMethod-method, 165
- as.data.frame.lcMethod, 8
- as.data.frame.lcMethods, 9
- as.lcMethods, 11
- as.list.lcMethod, 12
- evaluate.lcMethod, 29
- formula.lcMethod, 37
- lcMethod-class, 69
- names,lcMethod-method, 114
- update.lcMethod, 161

* lcMethod implementations

- getArgumentDefaults, 40
- getArgumentExclusions, 41
- lcMethod-class, 69
- lcMethodAkmedoids, 72
- lcMethodCrimCV, 73
- lcMethodDtwclust, 74
- lcMethodFeature, 75
- lcMethodFunction, 79
- lcMethodFunFEM, 80
- lcMethodGCKM, 81
- lcMethodKML, 83
- lcMethodLcmmGBTM, 84
- lcMethodLcmmGMM, 85
- lcMethodLMKM, 87
- lcMethodMclustLLPA, 88
- lcMethodMixAK_GLMM, 89
- lcMethodMixtoolsGMM, 91
- lcMethodMixtoolsNPRM, 92
- lcMethodRandom, 94
- lcMethodStratify, 97

* lcMethod package interfaces

- lcMethodFlexmix, 77
- lcMethodFlexmixGBTM, 78

* lcModel functions

- clusterNames, 13
- clusterProportions, 15
- clusterSizes, 16
- clusterTrajectories, 17
- coef.lcModel, 18
- converged, 21
- deviance.lcModel, 26
- df.residual.lcModel, 27
- estimationTime, 28
- externalMetric, 30
- fitted.lcModel, 35
- fittedTrajectories, 36
- getLcMethod, 46
- ids, 48
- lcModel-class, 100
- metric, 108
- model.frame.lcModel, 113
- nClusters, 115
- nIds, 116
- nobs.lcModel, 117
- plot-lcModel-method, 121
- plotClusterTrajectories, 122
- plotFittedTrajectories, 125
- postprob, 130
- predict.lcModel, 132
- predictAssignments, 134
- predictForCluster, 135
- predictPostprob, 137
- qqPlot, 142
- residuals.lcModel, 143
- sigma.lcModel, 145
- strip, 146
- time.lcModel, 150
- trajectoryAssignments, 154

* lcModels functions

- as.lcModels, 11
- lcModels, 103
- lcModels-class, 104

- max.lcModels, 106
- min.lcModels, 111
- plotMetric, 126
- print.lcModels, 141
- subset.lcModels, 147
- * **longitudinal cluster fit functions**
 - latrend, 52
 - latrendBatch, 61
 - latrendBoot, 62
 - latrendCV, 63
 - latrendRep, 66
- * **metric functions**
 - defineExternalMetric, 25
 - defineInternalMetric, 26
 - externalMetric, 30
 - getExternalMetricDefinition, 43
 - getExternalMetricNames, 44
 - getInternalMetricDefinition, 44
 - getInternalMetricNames, 45
 - metric, 108
- * **model-specific methods**
 - logLik.lcModel, 105
- * **validation methods**
 - createTestDataFold, 22
 - createTestDataFolds, 23
 - createTrainDataFolds, 24
 - latrendBoot, 62
 - latrendCV, 63
- * **variables**
 - idVariable, 49
 - responseVariable, 144
 - timeVariable, 151
- [[,lcMethod-method, 165
- \$,lcMethod-method ([[,lcMethod-method), 165

- A general overview of the lcModels class can be found here, 103
- An overview of the latrend package and its capabilities can be found here, 52
- APPA, 8, 21, 118
- APPA(), 58, 109
- approx, 68
- as.data.frame.lcMethod, 8, 10, 11, 13, 30, 38, 70, 114, 161, 165
- as.data.frame.lcMethods, 9, 9, 11, 13, 30, 38, 70, 114, 161, 165
- as.data.frame.lcModels, 10
- as.lcMethods, 9, 10, 11, 13, 30, 38, 70, 114, 161, 165
- as.lcModels, 11, 104, 107, 112, 126, 142, 148
- as.list.lcMethod, 9–11, 12, 30, 38, 70, 114, 161, 165
- assertthat::validate_that, 164
- atomic, 8

- base::difftime, 28
- bootstrapping, 6

- cluster metrics, 6
- cluster trajectories, 6
- clusterCrit::extCriteria(), 31, 32, 59
- clusterNames, 13, 15–17, 19, 22, 27–29, 33, 36, 37, 46, 49, 101, 111, 113, 115–117, 121, 124, 125, 131, 133, 135, 136, 138, 143–145, 147, 150, 155
- clusterNames(), 99
- clusterNames<-, 14
- clusterProportions, 14, 15, 16, 17, 19, 21, 22, 27–29, 33, 36, 37, 46, 49, 101, 111, 113, 115–117, 121, 124, 125, 131, 133, 135, 136, 138, 143–145, 147, 150, 155
- clusterProportions(), 99
- clusterProportions,lcModel-method (clusterProportions), 15
- clusterSizes, 14, 15, 16, 17, 19, 22, 27–29, 33, 36, 37, 46, 49, 101, 111, 113, 115–117, 121, 124, 125, 131, 133, 135, 136, 138, 143–145, 147, 150, 155
- clusterSizes(), 99
- clusterTrajectories, 7, 14–16, 17, 19, 22, 27–29, 33, 36, 37, 46, 49, 101, 111, 113, 115–117, 121, 123–125, 131, 133, 135, 136, 138, 143–145, 147, 150, 155
- clusterTrajectories(), 98, 99
- clusterTrajectories,lcModel-method (clusterTrajectories), 17
- coef.lcModel, 14–17, 18, 22, 27–29, 33, 36, 37, 46, 49, 101, 111, 113, 115–117, 121, 124, 125, 131, 133, 135, 136, 138, 143–145, 147, 150, 155
- compose, 19, 30

- compose(), *20, 34, 52, 63, 64, 66, 71, 130, 139, 140, 162, 164*
 compose, lcMetaMethod-method
 (interface-metaMethods), *50*
 compose, lcMethod-method (compose), *19*
 confusionMatrix, *8, 20, 118*
 converged, *14–17, 19, 21, 27–29, 33, 36, 37, 46, 49, 101, 111, 113, 115–117, 121, 124, 125, 131, 133, 135, 136, 138, 143–145, 147, 150, 155*
 converged(), *58, 99, 109*
 converged, lcModel-method (converged), *21*
 Convert, *10, 12, 103, 104, 107, 112, 126, 142, 147*
 createTestDataFold, *22, 23, 24, 63, 64*
 createTestDataFolds, *23, 23, 24, 63, 64*
 createTrainDataFolds, *23, 24, 63, 64*
 crimCV:::crimCV, *73*

 dcastRepeatedMeasures (tsmatrix), *159*
 defineExternalMetric, *25, 26, 33, 43–45, 111*
 defineExternalMetric(), *32, 58*
 defineInternalMetric, *25, 26, 33, 43–45, 111*
 defineInternalMetric(), *58, 109*
 deviance.lcModel, *14–17, 19, 22, 26, 28, 29, 33, 36, 37, 46, 49, 101, 111, 113, 115–117, 121, 124, 125, 131, 133, 135, 136, 138, 143–145, 147, 150, 155*
 df.residual.lcModel, *14–17, 19, 22, 27, 27, 29, 33, 36, 37, 46, 49, 101, 111, 113, 115–117, 121, 124, 125, 131, 133, 135, 136, 138, 143–145, 147, 150, 155*
 dist, *31*
 dtwclust:::tsclust, *74*

 environment, *70*
 estimated models, *57*
 estimation procedure, *19, 33, 129, 138, 139, 163*
 estimation time, *10, 12, 103, 104, 107, 112, 126, 142, 148*
 estimationTime, *14–17, 19, 22, 27, 28, 28, 33, 36, 37, 46, 49, 101, 111, 113, 115–117, 121, 124, 125, 131, 133, 135, 136, 138, 143–145, 147, 150, 155*
 estimationTime(), *58, 99, 109*
 estimationTime, lcModel-method
 (estimationTime), *28*
 estimationTime, lcModels-method
 (estimationTime), *28*
 estimationTime, list-method
 (estimationTime), *28*
 evaluate and compare, *54, 56*
 evaluate.lcMethod, *9–11, 13, 20, 29, 38, 70, 114, 161, 165*
 external metric, *10, 12, 103, 104, 107, 112, 126, 142, 147*
 External metrics, *58*
 externalMetric, *14–17, 19, 22, 25–29, 30, 36, 37, 43–46, 49, 59, 101, 107, 111–113, 115–117, 121, 124, 125, 131, 133, 135, 136, 138, 143–145, 147, 150, 155*
 externalMetric(), *99*
 externalMetric, lcModel, lcModel-method
 (externalMetric), *30*
 externalMetric, lcModels, character-method
 (externalMetric), *30*
 externalMetric, lcModels, lcModel-method
 (externalMetric), *30*
 externalMetric, lcModels, lcModels-method
 (externalMetric), *30*
 externalMetric, lcModels, missing-method
 (externalMetric), *30*
 externalMetric, list, lcModel-method
 (externalMetric), *30*

 fit, *33*
 fit(), *20, 28, 34, 52, 70, 71, 129, 130, 138–140, 164*
 fit, lcFitConverged-method
 (interface-metaMethods), *50*
 fit, lcFitRep-method
 (interface-metaMethods), *50*
 fit, lcMetaMethod-method
 (interface-metaMethods), *50*
 fit, lcMethod-method (fit), *33*
 fitted.lcApproxModel
 (lcApproxModel-class), *67*
 fitted.lcModel, *7, 14–17, 19, 22, 27–29, 33, 35, 36, 37, 46, 49, 101, 111, 113, 115–117, 121, 124, 125, 131, 133,*

- [135, 136, 138, 143–145, 147, 150, 155](#)
- `fitted.lcModel()`, [155, 156](#)
- `fittedTrajectories`, [7, 14–17, 19, 22, 27–29, 33, 36, 36, 46, 49, 101, 111, 113, 115–117, 121, 124, 125, 131, 133, 135, 136, 138, 143–145, 147, 150, 155](#)
- `fittedTrajectories()`, [99, 125](#)
- `fittedTrajectories.lcModel`-method (`fittedTrajectories`), [36](#)
- fitting procedure, [70](#)
- fitting procedure logic, [70](#)
- fitting until convergence, [56](#)
- `flexmix::flexmix`, [77, 78](#)
- `flexmix::FLXMRglm`, [78](#)
- `foreach`, [60](#)
- formals, [41](#)
- `formula.lcMethod`, [9–11, 13, 30, 37, 70, 114, 161, 165](#)
- `formula.lcModel`, [38](#)
- `funFEM::funFEM`, [81](#)
- `generateLongData`, [39, 65](#)
- `getArgumentDefaults`, [40, 42, 70, 72, 73, 75, 77, 80–83, 85, 87–91, 93, 95, 97](#)
- `getArgumentDefaults()`, [70](#)
- `getArgumentDefaults.lcMethod`-method (`getArgumentDefaults`), [40](#)
- `getArgumentExclusions`, [41, 41, 42, 70, 72, 73, 75, 77, 80–83, 85, 87–91, 93, 95, 97](#)
- `getArgumentExclusions.lcMethod`-method (`getArgumentExclusions`), [41](#)
- `getCall`, [162](#)
- `getCall.lcModel`, [14–17, 19, 22, 27–29, 33, 36, 37, 46, 49, 101, 111, 113, 115–117, 121, 124, 125, 131, 133, 135, 136, 138, 143–145, 147, 150, 155](#)
- `getCitation`, [42](#)
- `getCitation.lcMethod`-method (`getCitation`), [42](#)
- `getCitation.lcModel`-method (`getCitation`), [42](#)
- `getExternalMetricDefinition`, [25, 26, 33, 43, 44, 45, 111](#)
- `getExternalMetricNames`, [25, 26, 33, 43, 44, 44, 45, 111](#)
- `getExternalMetricNames()`, [30](#)
- `getInternalMetricDefinition`, [25, 26, 33, 43, 44, 44, 45, 111](#)
- `getInternalMetricNames`, [25, 26, 33, 43, 44, 45, 111](#)
- `getInternalMetricNames()`, [30, 108](#)
- `getLabel`, [45, 48](#)
- `getLabel.lcMethod`-method (`getLabel`), [45](#)
- `getLabel.lcModel`-method (`getLabel`), [45](#)
- `getLcMethod`, [14–17, 19, 22, 27–29, 33, 36, 37, 46, 49, 101, 111, 113, 115–117, 121, 124, 125, 131, 133, 135, 136, 138, 143–145, 147, 150, 155](#)
- `getLcMethod()`, [100](#)
- `getLcMethod.lcMetaMethod`-method (`interface-metaMethods`), [50](#)
- `getLcMethod.lcModel`-method (`getLcMethod`), [46](#)
- `getName`, [46, 47](#)
- `getName()`, [70](#)
- `getName.lcMetaMethod`-method (`interface-metaMethods`), [50](#)
- `getName.lcMethod`-method (`getName`), [47](#)
- `getName.lcModel`-method (`getName`), [47](#)
- `getName.NULL`-method (`getName`), [47](#)
- `getShortName`, [46, 48](#)
- `getShortName(getName)`, [47](#)
- `getShortName()`, [70](#)
- `getShortName.lcMetaMethod`-method (`interface-metaMethods`), [50](#)
- `getShortName.lcMethod`-method (`getName`), [47](#)
- `getShortName.lcModel`-method (`getName`), [47](#)
- `getShortName.NULL`-method (`getName`), [47](#)
- `ggplot2::ggplot`, [121](#)
- `gridExtra::arrangeGrob`, [122](#)
- here, [6](#)
- ids, [14–17, 19, 22, 27–29, 33, 36, 37, 46, 48, 101, 111, 113, 115–117, 121, 124, 125, 131, 133, 135, 136, 138, 143–145, 147, 150, 155](#)
- ids(), [99](#)
- idVariable, [49, 128, 144, 151, 153](#)
- idVariable, ANY-method (`idVariable`), [49](#)
- idVariable, lcMetaMethod-method (`interface-metaMethods`), [50](#)

- idVariable,lcMethod-method
(idVariable), 49
- idVariable,lcModel-method (idVariable),
49
- igraph::compare(), 32, 59
- igraph::split_join_distance(), 32, 59
- initialize,lcMethod-method, 50
- interface-metaMethods, 50
- internal metric, 10, 12, 103, 104, 107, 112,
126, 142, 147
- Internal metrics, 57
- internalMetric (metric), 108

- k-fold cross-validation, 6
- kml::kml, 83
- kml::parALGO, 83

- latrend, 7, 52, 62–64, 67, 71, 153, 162
- latrend estimation functions, 54, 98
- latrend(), 6, 19, 33, 45, 55, 71, 129, 138,
139, 163
- latrend-approaches, 7, 53, 57
- latrend-data, 7, 40, 54, 65, 119, 120
- latrend-estimation, 54, 55, 57
- latrend-generics, 56
- latrend-methods, 7, 54, 56
- latrend-metrics, 54, 57, 57
- latrend-package, 5, 55
- latrend-parallel, 7, 59, 61, 63, 64, 66
- latrend-procedure
(lcMethod-estimation), 71
- latrendBatch, 7, 53, 60, 61, 63, 64, 67
- latrendBatch(), 6, 55, 104
- latrendBoot, 7, 23, 24, 53, 60, 62, 62, 64, 67
- latrendBoot(), 55, 71
- latrendCV, 7, 23, 24, 53, 60, 62, 63, 63, 67
- latrendCV(), 55, 71
- latrendData, 6, 7, 55, 65
- latrendRep, 7, 53, 60, 62–64, 66
- latrendRep(), 55, 71, 104
- lcApproxModel (lcApproxModel-class), 67
- lcApproxModel-class, 67
- lcFitConverged (lcFitMethods), 68
- lcFitConverged-class (lcFitMethods), 68
- lcFitMethods, 68
- lcFitRep (lcFitMethods), 68
- lcFitRep-class (lcFitMethods), 68
- lcFitRepMax (lcFitMethods), 68
- lcFitRepMin (lcFitMethods), 68

- lcMetaMethod-class
(interface-metaMethods), 50
- lcMetaMethods (lcFitMethods), 68
- lcMethod, 7, 41, 42, 52, 63, 64, 66, 69, 71, 162
- lcMethod (lcMethod-class), 69
- lcMethod class, 71
- lcMethod-class, 69, 101
- lcMethod-estimation, 55, 71
- lcMethod-steps (lcMethod-estimation), 71
- lcMethodAkmedoids, 41, 42, 54, 56, 70, 72,
73, 75, 77, 80–83, 85, 87–91, 93, 95,
97
- lcMethodCrimCV, 41, 42, 54, 56, 70, 72, 73,
75, 77, 80–83, 85, 87–91, 93, 95, 97
- lcMethodDtwclust, 41, 42, 54, 56, 70, 72, 73,
74, 77, 80–83, 85, 87–91, 93, 95, 97
- lcMethodDtwclust(), 69
- lcMethodFeature, 41, 42, 54, 56, 70, 72, 73,
75, 75, 80–83, 85, 87–91, 93, 95, 97
- lcMethodFlexmix, 54, 56, 77, 79
- lcMethodFlexmixGBTM, 54, 56, 78, 78
- lcMethodFunction, 41, 42, 70, 72, 73, 75, 77,
79, 81–83, 85, 87–91, 93, 95, 97
- lcMethodFunFEM, 41, 42, 54, 56, 70, 72, 73,
75, 77, 80, 80, 82, 83, 85, 87–91, 93,
95, 97
- lcMethodGCKM, 41, 42, 54, 56, 70, 72, 73, 75,
77, 80, 81, 81, 83, 85, 87–91, 93, 95,
97
- lcMethodKML, 41, 42, 54, 56, 70, 72, 73, 75,
77, 80–82, 83, 85, 87–91, 93, 95, 97
- lcMethodKML(), 69
- lcMethodLcmmGBTM, 41, 42, 54, 56, 70, 72, 73,
75, 77, 80–83, 84, 87–91, 93, 95, 97
- lcMethodLcmmGBTM(), 69
- lcMethodLcmmGMM, 41, 42, 54, 56, 70, 72, 73,
75, 77, 80–83, 85, 85, 88–91, 93, 95,
97
- lcMethodLMKM, 41, 42, 54, 56, 70, 72, 73, 75,
77, 80–83, 85, 87, 87, 89–91, 93, 95,
97
- lcMethodLMKM(), 76
- lcMethodMclustLLPA, 41, 42, 54, 56, 70, 72,
73, 75, 77, 80–83, 85, 87, 88, 88, 90,
91, 93, 95, 97
- lcMethodMixAK_GLMM, 41, 42, 54, 56, 70, 72,
73, 75, 77, 80–83, 85, 87–89, 89, 91,
93, 95, 97

- lcMethodMixtoolsGMM, [41](#), [42](#), [54](#), [56](#), [70](#), [72](#), [73](#), [75](#), [77](#), [80–83](#), [85](#), [87–90](#), [91](#), [93](#), [95](#), [97](#)
- lcMethodMixtoolsNPRM, [41](#), [42](#), [54](#), [56](#), [70](#), [72](#), [73](#), [75](#), [77](#), [80–83](#), [85](#), [87–91](#), [92](#), [95](#), [97](#)
- lcMethodMixTVEM, [54](#), [56](#), [93](#)
- lcMethodRandom, [41](#), [42](#), [56](#), [70](#), [72](#), [73](#), [75](#), [77](#), [80–83](#), [85](#), [87–91](#), [93](#), [94](#), [97](#)
- lcMethods, [7](#), [96](#)
- lcMethods(), [6](#)
- lcMethodStratify, [41](#), [42](#), [56](#), [70](#), [72](#), [73](#), [75](#), [77](#), [80–83](#), [85](#), [87–91](#), [93](#), [95](#), [97](#)
- lcmm::gridsearch, [84](#), [86](#)
- lcmm::hlme, [84](#), [86](#)
- lcModel, [7](#), [20](#), [35](#), [52](#), [53](#), [71](#), [98](#), [100](#), [130](#), [139](#), [140](#), [164](#)
- lcModel-class, [100](#)
- lcModelPartition, [101](#)
- lcModels, [12](#), [103](#), [104](#), [107](#), [112](#), [126](#), [142](#), [148](#)
- lcModels(), [104](#)
- lcModels-class, [104](#)
- lcModelWeightedPartition, [105](#)
- length, lcMethod-method
(names, lcMethod-method), [114](#)
- list of models, [55](#)
- lme4::lmer, [82](#), [91](#)
- logLik, [58](#), [109](#)
- logLik.lcModel, [105](#)
- longitudinal cluster method, [98](#)
- longitudinal cluster methods, [55](#)
- longitudinal dataset, [55](#), [98](#)
- longitudinal datasets, [55](#)
- make.clusterNames(), [154](#)
- make.clusterPropLabels, [123](#)
- many different methods, [6](#)
- max.lcModels, [12](#), [104](#), [106](#), [111](#), [112](#), [126](#), [142](#), [148](#)
- maximizing, [10](#), [12](#), [103](#), [104](#), [107](#), [112](#), [126](#), [142](#), [148](#)
- mclust::Mclust, [89](#)
- mclustcomp::mclustcomp(), [30–32](#), [58](#), [59](#)
- meltRepeatedMeasures (tsframe), [158](#)
- meta methods, [56](#)
- method, [52](#), [55](#)
- method estimation procedure, [52](#)
- method specification, [100](#)
- methods, [55](#)
- methods::new(), [69](#)
- metric, [10](#), [12](#), [14–17](#), [19](#), [22](#), [25–29](#), [33](#), [36](#), [37](#), [43–46](#), [49](#), [59](#), [101](#), [103](#), [104](#), [106](#), [107](#), [108](#), [112](#), [113](#), [115–117](#), [121](#), [124–126](#), [131](#), [133](#), [135](#), [136](#), [138](#), [142–145](#), [147](#), [148](#), [150](#), [155](#)
- metric(), [99](#)
- metric, lcModel-method (metric), [108](#)
- metric, lcModels-method (metric), [108](#)
- metric, list-method (metric), [108](#)
- metrics, [6](#)
- min.lcModels, [12](#), [104](#), [107](#), [111](#), [111](#), [126](#), [142](#), [148](#)
- minimizing, [10](#), [12](#), [103](#), [104](#), [107](#), [112](#), [126](#), [142](#), [148](#)
- mixAK::GLMM_MCMC, [90](#)
- mixtools::npEM, [92](#)
- mixtools::regmixEM.mixed, [91](#)
- model, [55](#)
- model class, [6](#)
- model.data, [150](#)
- model.data.lcModel, [102](#), [112](#), [113](#)
- model.frame.lcModel, [14–17](#), [19](#), [22](#), [27–29](#), [33](#), [36](#), [37](#), [46](#), [49](#), [101](#), [111](#), [113](#), [113](#), [115–117](#), [121](#), [124](#), [125](#), [131](#), [133](#), [135](#), [136](#), [138](#), [143–145](#), [147](#), [150](#), [155](#)
- names, lcMethod-method, [114](#)
- nClusters, [14–17](#), [19](#), [22](#), [27–29](#), [33](#), [36](#), [37](#), [46](#), [49](#), [101](#), [111](#), [113](#), [115](#), [116](#), [117](#), [121](#), [124](#), [125](#), [131](#), [133](#), [135](#), [136](#), [138](#), [143–145](#), [147](#), [150](#), [155](#)
- nClusters(), [99](#)
- nClusters, lcModel-method (nClusters), [115](#)
- nIds, [14–17](#), [19](#), [22](#), [27–29](#), [33](#), [36](#), [37](#), [46](#), [49](#), [101](#), [111](#), [113](#), [115](#), [116](#), [117](#), [121](#), [124](#), [125](#), [131](#), [133](#), [135](#), [136](#), [138](#), [143–145](#), [147](#), [150](#), [155](#)
- nIds(), [99](#)
- nobs, [28](#), [115](#), [116](#)
- nobs(), [99](#)
- nobs.lcModel, [14–17](#), [19](#), [22](#), [27–29](#), [33](#), [36](#), [37](#), [46](#), [49](#), [101](#), [111](#), [113](#), [115](#), [116](#), [117](#), [121](#), [124](#), [125](#), [131](#), [133](#), [135](#), [136](#), [138](#), [143–145](#), [147](#), [150](#), [155](#)

- OCC, [8](#), [21](#), [117](#)
- one of the other standard estimation functions, [19](#), [33](#), [129](#), [138](#), [139](#), [163](#)
- others, [104](#)
- PAP.adh, [7](#), [55](#), [118](#)
- PAP.adh1y, [55](#), [119](#), [119](#)
- parallel computation, [84](#), [86](#)
- parallel-package, [60](#)
- plot, [124](#), [125](#)
- Plot a metric, [10](#), [12](#), [103](#), [104](#), [107](#), [112](#), [126](#), [142](#), [148](#)
- Plot the cluster trajectories, [10](#), [12](#), [103](#), [104](#), [107](#), [112](#), [126](#), [142](#), [148](#)
- plot, lcModel, ANY-method (plot-lcModel-method), [121](#)
- plot, lcModel-method (plot-lcModel-method), [121](#)
- plot, lcModels, ANY-method (plot-lcModels-method), [122](#)
- plot, lcModels-method (plot-lcModels-method), [122](#)
- plot-lcModel-method, [121](#)
- plot-lcModels-method, [122](#)
- plotClusterTrajectories, [7](#), [14–17](#), [19](#), [22](#), [27–29](#), [33](#), [36](#), [37](#), [46](#), [49](#), [101](#), [111](#), [113](#), [115–117](#), [121](#), [122](#), [125](#), [128](#), [131](#), [133](#), [135](#), [136](#), [138](#), [143–145](#), [147](#), [150](#), [155](#)
- plotClusterTrajectories(), [98](#), [99](#)
- plotClusterTrajectories, data.frame-method (plotClusterTrajectories), [122](#)
- plotClusterTrajectories, lcModel-method (plotClusterTrajectories), [122](#)
- plotFittedTrajectories, [14–17](#), [19](#), [22](#), [27–29](#), [33](#), [36](#), [37](#), [46](#), [49](#), [101](#), [111](#), [113](#), [115–117](#), [121](#), [124](#), [125](#), [128](#), [131](#), [133](#), [135](#), [136](#), [138](#), [143–145](#), [147](#), [150](#), [155](#)
- plotFittedTrajectories(), [99](#)
- plotFittedTrajectories, lcModel-method (plotFittedTrajectories), [125](#)
- plotMetric, [12](#), [104](#), [107](#), [112](#), [126](#), [142](#), [148](#)
- plotMetric(), [7](#)
- plotTrajectories, [121](#), [124](#), [125](#), [127](#), [153](#)
- plotTrajectories(), [6](#), [99](#)
- plotTrajectories, ANY-method (plotTrajectories), [127](#)
- plotTrajectories, data.frame-method (plotTrajectories), [127](#)
- plotTrajectories, lcModel-method (plotTrajectories), [127](#)
- postFit, [129](#)
- postFit(), [20](#), [35](#), [71](#), [130](#), [139](#), [140](#), [164](#)
- postFit, lcMetaMethod-method (interface-metaMethods), [50](#)
- postFit, lcMethod-method (postFit), [129](#)
- postprob, [7](#), [14–17](#), [19](#), [21](#), [22](#), [27–29](#), [33](#), [36](#), [37](#), [46](#), [49](#), [101](#), [111](#), [113](#), [115–117](#), [121](#), [124](#), [125](#), [130](#), [133](#), [135–138](#), [143–145](#), [147](#), [150](#), [155](#)
- postprob(), [15](#), [48](#), [99](#), [154](#)
- postprob, lcModel-method (postprob), [130](#)
- postprobFromAssignments, [132](#)
- predict(), [35](#), [135](#)
- predict.lcModel, [7](#), [14–17](#), [19](#), [22](#), [27–29](#), [33](#), [36](#), [37](#), [46](#), [49](#), [101](#), [111](#), [113](#), [115–117](#), [121](#), [124](#), [125](#), [131](#), [132](#), [135](#), [136](#), [138](#), [143–145](#), [147](#), [150](#), [155](#), [157](#), [158](#)
- predict.lcModel(), [136](#), [157](#)
- predictAssignments, [7](#), [14–17](#), [19](#), [22](#), [27–29](#), [33](#), [36](#), [37](#), [46](#), [49](#), [101](#), [111](#), [113](#), [115–117](#), [121](#), [124](#), [125](#), [131](#), [133](#), [134](#), [136](#), [138](#), [143–145](#), [147](#), [150](#), [155](#)
- predictAssignments(), [100](#)
- predictAssignments, lcModel-method (predictAssignments), [134](#)
- predictForCluster, [7](#), [14–17](#), [19](#), [22](#), [27–29](#), [33](#), [36](#), [37](#), [46](#), [49](#), [101](#), [111](#), [113](#), [115–117](#), [121](#), [124](#), [125](#), [131](#), [133](#), [135](#), [135](#), [138](#), [143–145](#), [147](#), [150](#), [155](#), [157](#), [158](#)
- predictForCluster(), [35](#), [99](#), [133](#)
- predictForCluster, lcApproxModel-method (lcApproxModel-class), [67](#)
- predictForCluster, lcModel-method (predictForCluster), [135](#)
- predictPostprob, [7](#), [14–17](#), [19](#), [22](#), [27–29](#), [33](#), [36](#), [37](#), [46](#), [49](#), [101](#), [111](#), [113](#), [115–117](#), [121](#), [124](#), [125](#), [131](#), [133–136](#), [137](#), [143–145](#), [147](#), [150](#), [155](#)
- predictPostprob(), [99](#)
- predictPostprob, lcModel-method

- (predictPostprob), 137
- preFit, 53, 138
- preFit(), 20, 34, 71, 129, 130, 139, 140, 164
- preFit,lcMetaMethod-method
 - (interface-metaMethods), 50
- preFit,lcMethod-method (preFit), 138
- prepareData, 139
- prepareData(), 20, 34, 71, 129, 130, 138–140, 164
- prepareData,lcMetaMethod-method
 - (interface-metaMethods), 50
- prepareData,lcMethod-method
 - (prepareData), 139
- Print, 10, 12, 103, 104, 107, 112, 126, 142, 147
- print.lcMethod, 141
- print.lcModels, 12, 104, 107, 112, 126, 141, 148
- psych::cohen.kappa(), 31, 58
- qqPlot, 14–17, 19, 22, 27–29, 33, 36, 37, 46, 49, 101, 111, 113, 115–117, 121, 124, 125, 131, 133, 135, 136, 138, 142, 144, 145, 147, 150, 155
- qqPlot(), 99
- qqplotr::geom_qq_band(), 142
- qqplotr::stat_qq_line(), 142
- qqplotr::stat_qq_point(), 142
- R.utils::Verbose, 34, 52, 53, 61, 63, 64, 67, 129, 138, 140, 162
- repeated fitting, 56
- residuals, 28
- residuals.lcModel, 14–17, 19, 22, 27–29, 33, 36, 37, 46, 49, 101, 111, 113, 115–117, 121, 124, 125, 131, 133, 135, 136, 138, 142, 143, 143, 145, 147, 150, 155
- responseVariable, 50, 123, 144, 151, 153
- responseVariable,lcMetaMethod-method
 - (interface-metaMethods), 50
- responseVariable,lcMethod-method
 - (responseVariable), 144
- responseVariable,lcModel-method
 - (responseVariable), 144
- see here, 56
- See here., 55
- sigma(), 99, 145
- sigma.lcModel, 14–17, 19, 22, 27–29, 33, 36, 37, 46, 49, 101, 111, 113, 115–117, 121, 124, 125, 131, 133, 135, 136, 138, 143, 144, 145, 147, 150, 155
- stats::AIC(), 58, 109
- stats::BIC(), 58, 109
- stats::deviance, 27
- stats::deviance(), 58, 109
- stats::df.residual, 28
- stats::fitted, 36
- stats::formula, 39
- stats::lm, 88
- stats::logLik, 106
- stats::logLik(), 58, 109
- stats::model.frame, 113
- stats::model.frame(), 113
- stats::predict, 133
- stats::sigma(), 58, 109
- Steps performed during estimation, 7
- strip, 14–17, 19, 22, 27–29, 33, 36, 37, 46, 49, 101, 111, 113, 115–117, 121, 124, 125, 131, 133, 135, 136, 138, 143–145, 146, 150, 155
- strip(), 100
- strip,ANY-method (strip), 146
- strip,lcMethod-method (strip), 146
- strip,lcModel-method (strip), 146
- Subset, 10, 12, 103, 104, 107, 112, 126, 142, 147
- subset.lcModels, 12, 104, 107, 112, 126, 142, 147
- summary(), 99
- summary.lcModel, 148
- test.latrend, 149
- time(), 99
- time.lcModel, 14–17, 19, 22, 27–29, 33, 36, 37, 46, 49, 101, 111, 113, 115–117, 121, 124, 125, 131, 133, 135, 136, 138, 143–145, 147, 150, 155
- timevariable,ANY-method (timeVariable), 151
- timeVariable, 50, 123, 128, 144, 150, 151, 153
- timeVariable,ANY-method (timeVariable), 151
- timeVariable,lcMetaMethod-method
 - (interface-metaMethods), 50

- timeVariable,lcMethod-method
(timeVariable), 151
- timeVariable,lcModel-method
(timeVariable), 151
- trajectories, 6, 55, 127, 128, 133, 144, 152
- trajectories(), 52, 66, 99, 162
- trajectories,call-method
(trajectories), 152
- trajectories,data.frame-method
(trajectories), 152
- trajectories,lcModel-method
(trajectories), 152
- trajectories,matrix-method
(trajectories), 152
- trajectoryAssignments, 7, 14–17, 19, 21,
22, 27–29, 33, 36, 37, 46, 49, 101,
111, 113, 115–117, 121, 124, 125,
131, 133, 135, 136, 138, 143–145,
147, 150, 153, 154, 164
- trajectoryAssignments(), 16, 21, 48, 98,
99
- trajectoryAssignments,lcModel-method
(trajectoryAssignments), 154
- trajectoryAssignments,matrix-method
(trajectoryAssignments), 154
- transformFitted, 36, 155
- transformFitted(), 35
- transformFitted,data.frame,lcModel-method
(transformFitted), 155
- transformFitted,list,lcModel-method
(transformFitted), 155
- transformFitted,matrix,lcModel-method
(transformFitted), 155
- transformFitted,NULL,lcModel-method
(transformFitted), 155
- transformPredict, 157
- transformPredict(), 133
- transformPredict,data.frame,lcModel-method
(transformPredict), 157
- transformPredict,matrix,lcModel-method
(transformPredict), 157
- transformPredict,NULL,lcModel-method
(transformPredict), 157
- transformPredict,vector,lcModel-method
(transformPredict), 157
- tsframe, 158, 160
- tsmatrix, 159, 159
- update(), 69, 100
- update.lcMetaMethod
(interface-metaMethods), 50
- update.lcMethod, 9–11, 13, 30, 38, 70, 114,
161, 165
- update.lcModel, 162
- utils::citation, 43
- validate, 163
- validate(), 20, 34, 70, 71, 130, 139, 140, 164
- validate,lcFitConverged-method
(interface-metaMethods), 50
- validate,lcFitRep-method
(interface-metaMethods), 50
- validate,lcMetaMethod-method
(interface-metaMethods), 50
- validate,lcMethod-method (validate), 163
- Verbose, 53, 61, 63, 64, 67, 162
- which.is.max, 154
- which.max(), 154
- which.weight, 164
- which.weight(), 154