

Package: LKT (via r-universe)

October 9, 2026

Title Logistic Knowledge Tracing

Version 2.0.0

Description Computes Logistic Knowledge Tracing ('LKT') which is a general method for tracking human learning in an educational software system. Please see Pavlik, Eglington, and Harrell-Williams (2021) <[doi:10.1109/TLT.2021.3128569](https://doi.org/10.1109/TLT.2021.3128569)>. 'LKT' is a method to compute features of student data that are used as predictors of subsequent performance. 'LKT' allows great flexibility in the choice of predictive components and features computed for these predictive components. The system is built on top of 'LiblineaR', which enables extremely fast solutions compared to base glm() in R.

License GPL-3

Encoding UTF-8

LazyData true

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Collate 'LKT-package.R' 'data.R' 'lkt-feature-input.R'
'lkt-feature-computation.R' 'lkt-model-interface.R'
'lkt-model-liblinear.R' 'lkt-model-online-calibration.R'
'lkt-fast-online-simple-adaptive.R' 'lkt-latency.R'
'lkt-prediction.R' 'LKTfunctions.R' 'lkt-hdi.R' 'lkt-search.R'

Depends R (>= 3.5.0), SparseM (>= 1.83), methods, Matrix, data.table (>= 1.13.2), LiblineaR (>= 2.10-8)

Imports glmnet (>= 4.0-2), lme4 (>= 1.1-23), cluster (>= 2.1.3), pROC (>= 1.16.2), crayon, HDInterval (>= 0.2.2)

Suggests rmarkdown, knitr, utils, ggplot2

Config/pak/sysreqs cmake make

Repository <https://optimal-learning-lab.r-universe.dev>

Date/Publication 2026-07-11 01:51:36 UTC

RemoteUrl <https://github.com/optimal-learning-lab/lkt>

RemoteRef HEAD

RemoteSha e6810522c23f030845b0cd41a39b23c8fe4403d2

Contents

buildLKTModel	2
computefeatures	4
computeSpacingPredictors	5
countOutcomeold	6
largerawsample	7
LASSOLKTData	7
LASSOLKTModel	9
LibLinearModel	10
LKT	11
LKT_HDI	13
LKTCustomModel	14
LKTFeatureInput	15
LKTModelFit	16
LKTOnlineSimpleAdaptiveDecayEval	17
LKTOnlineSimpleAdaptiveEval	18
LKTOnlineSimpleAdaptiveGradient	18
LKTOnlineSimpleAdaptiveInput	19
LKTOptimizeOnlineSimpleAdaptive	20
LKTOptimizeOnlineSimpleAdaptiveAlpha	21
LKTOptimizeOnlineSimpleAdaptiveDecayAlpha	22
OnlineAdaptiveModel	23
OnlineCalibrationModel	25
predict_lkt	26
samplelkt	28
smallSet	28
ViewExcel	29
Index	30

buildLKTModel

Search for an LKT model specification

Description

Runs forward and/or backward stepwise search over candidate component-feature pairs, tracking nonlinear parameters and common fit metrics. This is a legacy public model-selection helper used by the vignette examples.

Usage

```

buildLKTModel(
  data,
  usefolds = NA,
  allcomponents,
  allfeatures,
  currentcomponents = c(),
  specialcomponents = c(),
  specialfeatures = c(),
  forv,
  bacv,
  preset = NA,
  presetint = T,
  currentfeatures = c(),
  verbose = FALSE,
  currentfixedpars = c(),
  maxitv = 10,
  interc = FALSE,
  forward = TRUE,
  backward = TRUE,
  metric = "BIC",
  removefeat = c(),
  removecomp = c()
)

```

Arguments

<code>data</code>	Data frame or data table containing <code>Anon.Student.Id</code> , <code>CF..ansbin.</code> , component columns, and optional fold.
<code>usefolds</code>	Numeric vector of folds passed to <code>LKT()</code> for fitting.
<code>allcomponents</code>	Character vector of candidate component columns.
<code>allfeatures</code>	Character vector of candidate feature names.
<code>currentcomponents</code>	Character vector of component columns in the starting model.
<code>specialcomponents</code>	Character vector of components paired one-to-one with <code>specialfeatures</code> and included outside the crossed search grid.
<code>specialfeatures</code>	Character vector of special feature names paired with <code>specialcomponents</code> .
<code>forv</code>	Minimum improvement required to add a term.
<code>bacv</code>	Maximum loss tolerated when removing a term.
<code>preset</code>	Optional preset model family: "static", "AFM", "PFA", "advanced", "AFMLLTM", "PFALLTM", or "advancedLLTM".
<code>presetint</code>	Logical; include preset intercept terms when TRUE.
<code>currentfeatures</code>	Character vector of feature names in the starting model.

verbose	Logical passed to LKT().
currentfixedpars	Numeric vector of fixed parameter values for the starting model.
maxitv	Maximum nonlinear optimizer iterations passed to LKT().
interc	Logical passed to LKT() to include a global intercept.
forward	Logical; allow forward additions.
backward	Logical; allow backward removals.
metric	Selection metric. One of "BIC", "AUC", "AIC", "RMSE", or "R2".
removefeat	Character vector of feature names to exclude from the crossed candidate list.
removecomp	Character vector of component names paired with removefeat to exclude from the crossed candidate list.

Value

A list containing tractable search history and the final currentfit LKT model.

Examples

```
data(samplelkt)
samplelkt$CF..ansbin. <- as.integer(samplelkt$Outcome == "CORRECT")
searched <- buildLKTModel(
  data = samplelkt,
  allcomponents = "KC..Default.",
  allfeatures = "lineafm",
  forv = 0,
  bacv = 0,
  interc = TRUE,
  verbose = FALSE
)
names(searched)
```

computefeatures	<i>Compute an LKT feature vector</i>
-----------------	--------------------------------------

Description

Computes one feature vector for a prepared LKT component context. This is a legacy public helper used by examples and advanced workflows; most users should call LKT() instead.

Usage

```
computefeatures(data, feat, par1, par2, index, index2, par3, par4, par5, fcomp)
```

Arguments

data	Data frame or data table after LKT component preparation.
feat	Feature name to compute.
par1	First nonlinear feature parameter.
par2	Second nonlinear feature parameter.
index	Student-by-component index vector.
index2	Component-level index vector.
par3	Third nonlinear feature parameter.
par4	Fourth nonlinear feature parameter.
par5	Fifth nonlinear feature parameter.
fcomp	Component column name used by features that read component-level spacing or numeric data.

Value

A vector suitable for use as one model-matrix input column.

Examples

```
data(samplelkt)
data.table::setDT(samplelkt)
samplelkt$CF..ansbin. <- as.integer(samplelkt$Outcome == "CORRECT")
prepared <- computeSpacingPredictors(samplelkt, "KC..Default.")
prepared$history_index <- paste(prepared$KC..Default., prepared$Anon.Student.Id)
prepared$index <- prepared$history_index
prepared$indexcomp <- prepared$KC..Default.
prepared$cor <- countOutcomeold(prepared, "history_index", "CORRECT")
prepared$icor <- countOutcomeold(prepared, "history_index", "INCORRECT")
head(computeFeatures(
  prepared, "lineafm", 0, 0,
  prepared$index, prepared$indexcomp, 0, 0, 0, "KC..Default."
))
```

computeSpacingPredictors

Compute spacing predictors for LKT features

Description

Adds repetition-spacing history columns for one or more component columns. These columns are required by recency, spacing, and forgetting features in LKT(). If CF..reltime. or CF..Time. is missing, the function derives it from Duration..sec. when possible.

Usage

```
computeSpacingPredictors(data, KCs)
```

Arguments

data	Data frame or data table containing Anon.Student.Id, CF..ansbin., and each component named in KCs. Requires either CF..reltime. or Duration..sec..
KCs	Character vector of component columns for which spacing predictors should be computed.

Value

The input data with added spacing-history columns for each component: spacing, relspacing, prev, meanspacing, relmeanspacing, and spacinglagged suffixes.

Examples

```
data(samplelkt)
data.table::setDT(samplelkt)
samplelkt$CF..ansbin. <- as.integer(samplelkt$Outcome == "CORRECT")
spacing_data <- computeSpacingPredictors(samplelkt, "KC..Default.")
head(spacing_data$KC..Default.spacing)
```

countOutcomeold	<i>Count prior outcomes within an index</i>
-----------------	---

Description

Legacy exported helper that counts previous rows whose Outcome value equals response within each index group. It is retained for compatibility with older examples; new package code uses internal helpers.

Usage

```
countOutcomeold(data, index, response)
```

Arguments

data	Data frame or data table containing an Outcome column.
index	Grouping column name or data.table by expression that defines the history stream for each row.
response	Outcome value to count, such as "CORRECT" or "INCORRECT".

Value

Numeric vector of lagged cumulative counts, one value per row.

Examples

```
data(samplelkt)
data.table::setDT(samplelkt)
samplelkt$CF..ansbin. <- as.integer(samplelkt$Outcome == "CORRECT")
samplelkt$history_index <- paste(samplelkt$KC.Default., samplelkt$Anon.Student.Id)
head(countOutcomeold(samplelkt, "history_index", "CORRECT"))
```

largerawsample	<i>Raw Memphis DataShop sample</i>
----------------	------------------------------------

Description

A raw trial-level sample from the Memphis DataShop export used in examples.

Usage

```
largerawsample
```

Format

A data frame with the original exported DataShop columns.

Source

<https://pslcdatashop.web.cmu.edu/Export?datasetId=5513>

LASSOLKTData	<i>Build LASSO input data for LKT feature search</i>
--------------	--

Description

Constructs the sparse design matrices used by `LASSOLKTModel()` by expanding nonlinear features across a grid of fixed parameter values. This legacy public helper is vignette-facing and returns the `nosolve = TRUE` form of `LKT()`.

Usage

```
LASSOLKTData(
  data,
  gridpars,
  allcomponents,
  allfeatures,
  preset = NA,
  presetint = T,
  specialcomponents = c(),
  specialfeatures = c(),
```

```

    specialpars = c(),
    removefeat = c(),
    removecomp = c()
  )

```

Arguments

<code>data</code>	Data frame or data table containing <code>Anon.Student.Id</code> , <code>CF..ansbin.</code> , component columns, and optional fold.
<code>gridpars</code>	Numeric vector of parameter values used to materialize nonlinear feature candidates.
<code>allcomponents</code>	Character vector of candidate component columns.
<code>allfeatures</code>	Character vector of candidate feature names.
<code>preset</code>	Optional preset model family: "static", "AFM", "PFA", "advanced", "AFMLLTM", "PFALLTM", or "advancedLLTM".
<code>presetint</code>	Logical; include preset intercept terms when TRUE.
<code>specialcomponents</code>	Character vector of components paired one-to-one with <code>specialfeatures</code> .
<code>specialfeatures</code>	Character vector of special feature names paired with <code>specialcomponents</code> .
<code>specialpars</code>	Numeric vector of parameter values for <code>specialfeatures</code> .
<code>removefeat</code>	Character vector of feature names to exclude from the crossed candidate list.
<code>removecomp</code>	Character vector of component names paired with <code>removefeat</code> to exclude from the crossed candidate list.

Value

A list from `LKT(nosolve = TRUE)` containing LASSO feature names and sparse design data.

Examples

```

data(samplelkt)
samplelkt$CF..ansbin. <- as.integer(samplelkt$Outcome == "CORRECT")
lasso_data <- LASSOLKTData(
  data = samplelkt,
  gridpars = c(0.25, 0.5),
  allcomponents = "KC..Default.",
  allfeatures = "lineafm"
)
names(lasso_data)

```

LASSOLKTModel

*Fit a LASSO-selected LKT model***Description**

Builds LKT feature matrices with LASSOLKTData() and fits a binomial glmnet path. This is a legacy public model-selection helper retained for vignette compatibility.

Usage

```
LASSOLKTModel(
  data,
  gridpars,
  allcomponents,
  preset = NA,
  presetint = T,
  allfeatures,
  specialcomponents = c(),
  specialfeatures = c(),
  specialpars = c(),
  target_n,
  removefeat = c(),
  removecomp = c(),
  test_fold = 1
)
```

Arguments

data	Data frame or data table containing Anon.Student.Id, CF..ansbin., component columns, and a fold column.
gridpars	Numeric vector of parameter values used to materialize nonlinear feature candidates.
allcomponents	Character vector of candidate component columns.
preset	Optional preset model family: "static", "AFM", "PFA", "advanced", "AFMLLTM", "PFALLTM", or "advancedLLTM".
presetint	Logical; include preset intercept terms when TRUE.
allfeatures	Character vector of candidate feature names.
specialcomponents	Character vector of components paired one-to-one with specialfeatures.
specialfeatures	Character vector of special feature names paired with specialcomponents.
specialpars	Numeric vector of parameter values for specialfeatures.
target_n	Desired number of nonzero model features used to choose a lambda from the fitted path.

removefeat	Character vector of feature names to exclude from the crossed candidate list.
removecomp	Character vector of component names paired with removefeat to exclude from the crossed candidate list.
test_fold	Fold value used as the held-out test fold.

Value

A list containing training and test matrices, response vectors, the fitted glmnet object, AUC/RMSE/BIC by lambda, selected lambda index, predictions, selected coefficients, and selected feature summary.

Examples

```
## Not run:
data(samplelkt)
samplelkt$CF..ansbin. <- as.integer(samplelkt$Outcome == "CORRECT")
samplelkt$fold <- rep(1:2, length.out = nrow(samplelkt))
lasso_model <- LASSOLKTModel(
  data = samplelkt,
  gridpars = c(0.25, 0.5),
  allcomponents = "KC..Default.",
  allfeatures = "lineafm",
  target_n = 2,
  test_fold = 1
)
names(lasso_model)

## End(Not run)
```

LibLinearModel

Define the default LibLinear LKT model

Description

Creates the default LKT model adapter backed by LiblinearR. This model uses the shared LKT feature input and returns the common LKTModelFit() shape.

Usage

```
LibLinearModel()
```

Value

A model adapter suitable for LKT(model = ...).

Examples

```
adapter <- LibLinearModel()
adapter$name
```

LKT

*Fit a logistic knowledge tracing model***Description**

Computes LKT features from trial-level learning data and fits a logistic model that predicts CF..ansbin.. The default model adapter uses LibLinear; custom adapters can be supplied with LKTCustomModel().

Usage

```
LKT(
  data,
  usefolds = NA,
  components,
  features,
  fixedpars = NA,
  seedpars = NA,
  interacts = NA,
  curvefeats = NA,
  dualfit = FALSE,
  interc = FALSE,
  verbose = TRUE,
  epsilon = 1e-04,
  cost = 512,
  lowb = 1e-05,
  highb = 0.99999,
  type = 0,
  maketimes = FALSE,
  bias = 0,
  maxitv = 100,
  factrv = 1e+12,
  nosolve = FALSE,
  autoKC = rep(0, length(components)),
  autoKCcont = rep("NA", length(components)),
  connectors = rep("+", max(1, length(components) - 1)),
  distribution = "exp",
  weibull_mse_weight = 0,
  weibull_t0_fixed = NULL,
  model = LibLinearModel(),
  model_options = list()
)
```

Arguments

data	Data frame or data table containing one row per trial. Must include Anon.Student.Id, CF..ansbin., and every component column named in components. Some features also require timing columns such as CF..Time., CF..reltime., or Duration..sec..
------	---

usefolds	Numeric vector of fold values to use for fitting. Features are still computed over all rows so held-out rows can be evaluated later. NA fits on all rows.
components	Character vector naming component columns or component specifications used to build feature histories.
features	Character vector of LKT feature names, parallel to components.
fixedpars	Numeric vector of nonlinear feature parameters. Use NA entries to optimize corresponding parameters.
seedpars	Numeric vector of starting values for nonlinear parameter optimization. NA uses default starts.
interacts	Optional vector of interaction terms added to component feature terms. Use NA for no interaction.
curvefeats	Optional vector of columns used by legacy difficulty curve features.
dualfit	Logical; if TRUE, also fit a response-time model. Requires Duration..sec..
interc	Logical; if TRUE, include a global intercept.
verbose	Logical; if TRUE, print progress and fit diagnostics.
epsilon	Numeric stopping tolerance passed to LiblineaR.
cost	Numeric regularization cost passed to LiblineaR.
lowb	Lower bound for nonlinear parameter optimization.
highb	Upper bound for nonlinear parameter optimization.
type	Integer LiblineaR solver type.
maketimes	Logical; if TRUE, create missing time columns from Duration..sec. where possible.
bias	Numeric bias term passed to LiblineaR.
maxitv	Maximum iteration count for nonlinear optimization.
factrv	Factr control value passed to optim() for nonlinear optimization.
nosolve	Logical; if TRUE, return the constructed sparse design data instead of fitting a model. This is a legacy public interface used by LASSO search helpers.
autoKC	Integer vector controlling automatic KC clustering. 0 disables clustering for the corresponding component; positive values are cluster counts.
autoKCcont	Character vector controlling automatic KC randomization. Use "rand" to randomize cluster assignment for comparison runs.
connectors	Character vector of formula operators ("+", ":", or "*") between model terms. Defaults to additive terms.
distribution	Latency model distribution. Supported values are "exp" and "Weibull" when dualfit = TRUE.
weibull_mse_weight	Numeric weight in [0, 1] for the point-prediction component of Weibull latency fitting.
weibull_t0_fixed	Optional fixed minimum response time, in seconds, for Weibull latency fitting.
model	LKT model adapter. Defaults to LibLinearModel().
model_options	Optional list of adapter-specific model options.

Value

A list containing the fitted model, coefficients, predictions, fit statistics, processed data, feature metadata, and optional latency-model results. With `nosolve = TRUE`, returns the constructed sparse matrices and feature names for downstream legacy model-selection helpers.

Examples

```
data(samplelkt)
fit <- LKT(
  data = samplelkt,
  components = "KC..Default.",
  features = "intercept",
  interc = TRUE,
  verbose = FALSE
)
head(fit$prediction)
```

LKT_HDI

Estimate coefficient HDIs by bootstrap resampling

Description

Fits a full LKT model and repeated student-level bootstrap samples, then summarizes coefficient intervals with `HDInterval`. This helper is computation-heavy and is retained as a legacy public interpretation aid.

Usage

```
LKT_HDI(
  dat,
  n_boot,
  n_students,
  comps,
  feats,
  conns = rep("+", max(1, length(comps) - 1)),
  ints = NA,
  fixeds,
  get_hdi = TRUE,
  cred_mass = 0.95
)
```

Arguments

<code>dat</code>	Data frame or data table containing trial data for <code>LKT()</code> .
<code>n_boot</code>	Number of bootstrap subsamples to fit.
<code>n_students</code>	Number of students per bootstrap subsample.

comps	Character vector of components passed to LKT().
feats	Character vector of features passed to LKT().
conns	Character vector of formula connectors passed to LKT().
ints	Optional interactions passed to LKT().
fixeds	Fixed nonlinear parameters passed to LKT().
get_hdi	Logical compatibility argument. The current implementation always returns coef_hdi.
cred_mass	Credibility mass used for HDI width.

Value

A list containing bootstrap coefficient replicates (par_reps), the full-data model (mod_full), and coefficient intervals (coef_hdi).

Examples

```
data(samplelkt)
hdi_result <- LKT_HDI(
  samplelkt,
  n_boot = 2,
  n_students = 2,
  comps = "KC.Default.",
  feats = "intercept",
  fixeds = numeric(0)
)
names(hdi_result)
```

LKTCustomModel	<i>Define a custom LKT model</i>
----------------	----------------------------------

Description

Creates a model adapter that can be passed to LKT(model = ...). Model fit functions receive one argument named request and may also accept verbose. The request contains the shared feature input (formula, data, response, feature_spec, and sparse design matrices) plus fit controls such as usefolds and model options.

Usage

```
LKTCustomModel(name, fit)
```

Arguments

name	Single model name used in returned LKT objects and errors.
fit	Function with signature like function(request, verbose = FALSE).

Value

A model adapter suitable for LKT(model = ...).

Examples

```
adapter <- LKTCustomModel(
  "constant",
  function(request, verbose = FALSE) {
    LKTModelFit(
      model = list(name = "constant"),
      pred = rep(0.5, length(request$response))
    )
  }
)
inherits(adapter, "LKTModel")
```

LKTFeatureInput

Create model-agnostic LKT feature input

Description

Builds the shared input consumed by LKT models. The feature input contains the fitted formula, processed data, response vector, feature specification, sparse design matrices, and design-matrix column names.

Usage

```
LKTFeatureInput(formula, data, feature_spec, terms_cache = NULL)
```

Arguments

formula	Model formula after LKT feature construction.
data	Processed trial-level data used for fitting.
feature_spec	Per-feature metadata collected during feature construction.
terms_cache	Optional environment used to cache formula terms.

Value

An LKT feature input object consumed by model adapters.

Examples

```
data(samplelkt)
data.table::setDT(samplelkt)
samplelkt$CF..ansbin. <- as.integer(samplelkt$Outcome == "CORRECT")
input <- LKTFeatureInput(
  CF..ansbin. ~ KC..Default.,
  samplelkt,
```

```

    feature_spec = data.frame(feature = "intercept", component = "KC..Default.")
  )
  input$column_names

```

LKTModelFit

Create an LKT model fit result

Description

Standardizes the fit-result object returned by LKT model adapters. `model` and `pred` are required. `pred` must be one finite probability per row in the shared feature input, strictly between 0 and 1. If `modelvs`, `fitstat`, or predictor matrices are omitted, LKT fills common defaults where possible.

Usage

```

LKTModelFit(
  model,
  pred,
  modelvs = NULL,
  fitstat = NULL,
  predictset = NULL,
  predictset2 = NULL
)

```

Arguments

<code>model</code>	Fitted model object.
<code>pred</code>	Numeric probability vector, one value per input row.
<code>modelvs</code>	Optional coefficient matrix/data frame with a coefficient column.
<code>fitstat</code>	Optional log likelihood or comparable fit statistic.
<code>predictset</code>	Optional sparse model matrix.
<code>predictset2</code>	Optional CSR model matrix used by existing LKT outputs.

Value

A model fit-result list.

Examples

```

fit_result <- LKTModelFit(
  model = list(name = "constant"),
  pred = c(0.5, 0.5)
)
names(fit_result)

```

LKTOonlineSimpleAdaptiveDecayEval

Evaluate online simple-adaptive decay learning

Description

Replays the online simple-adaptive model while adapting the recency decay parameter within subject using $d \leftarrow d - \alpha_{\text{decay}} * \text{error}$.

Usage

```
LKTOonlineSimpleAdaptiveDecayEval(
  par,
  input,
  clip_epsilon = 1e-06,
  denom_epsilon = 1e-06,
  decay_lower = 1e-05,
  decay_upper = 0.99999,
  return_details = FALSE,
  require_native = FALSE
)
```

Arguments

par	Numeric vector: intercept, recency beta, logsuc beta, logitdec beta, recency alpha, logsuc alpha, and recency-decay alpha.
input	Output from LKTOonlineSimpleAdaptiveInput().
clip_epsilon	Probability clipping value.
denom_epsilon	Denominator stabilizer for beta updates.
decay_lower	Lower bound for the within-subject recency decay state.
decay_upper	Upper bound for the within-subject recency decay state.
return_details	Logical; return predictions and final subject states.
require_native	Logical; stop if the compiled native evaluator is not loaded.

Value

Negative log likelihood, or details when requested.

LKTONlineSimpleAdaptiveEval

Evaluate the fast online simple-adaptive objective

Description

Evaluate the fast online simple-adaptive objective

Usage

```
LKTONlineSimpleAdaptiveEval(
  par,
  input,
  clip_epsilon = 1e-06,
  denom_epsilon = 1e-06,
  return_details = FALSE,
  require_native = FALSE
)
```

Arguments

par	Numeric vector: intercept, recency beta, logsuc beta, logitdec beta, recency alpha, logsuc alpha.
input	Output from LKTONlineSimpleAdaptiveInput().
clip_epsilon	Probability clipping value.
denom_epsilon	Denominator stabilizer.
return_details	Logical; return predictions and final subject betas.
require_native	Logical; stop if the compiled native evaluator is not loaded.

Value

Negative log likelihood, or details when requested.

LKTONlineSimpleAdaptiveGradient

Gradient for the fast online simple-adaptive objective

Description

Gradient for the fast online simple-adaptive objective

Usage

```
LKTOonlineSimpleAdaptiveGradient(
  par,
  input,
  clip_epsilon = 1e-06,
  denom_epsilon = 1e-06,
  require_native = FALSE
)
```

Arguments

par	Numeric vector: intercept, recency beta, logsuc beta, logitdec beta, recency alpha, logsuc alpha.
input	Output from LKTOonlineSimpleAdaptiveInput().
clip_epsilon	Probability clipping value.
denom_epsilon	Denominator stabilizer.
require_native	Logical; stop if the compiled native evaluator is not loaded.

Value

Gradient of the negative log likelihood.

LKTOonlineSimpleAdaptiveInput

Fast online simple-adaptive objective input

Description

Builds compact numeric arrays for the fast compiled objective used by the simple-adaptive online recency/logsuc model.

Usage

```
LKTOonlineSimpleAdaptiveInput(
  modelob,
  data,
  subject_col = "Anon.Student.Id",
  response_col = "CF..ansbin.",
  recency_coef = "recencyKC..Default.",
  logsuc_coef = "logsucKC..Default.",
  logitdec_coef = "logitdecAnon.Student.Id",
  spacing_col = "KC..Default.spacing",
  recency_decay_start = NULL
)
```

Arguments

modelob	Fitted simple-adaptive LKT model with predictors and coefs.
data	Data used for the fit.
subject_col	Subject identifier column.
response_col	Binary response column.
recency_coef	Coefficient name for the recency feature.
logsuc_coef	Coefficient name for the log-success feature.
logitdec_coef	Coefficient name for the student logitdec feature.
spacing_col	Raw spacing column used to recompute recency when the recency decay parameter is adapted online.
recency_decay_start	Optional starting recency decay value. If NULL, the value is read from modelob\$model_specification.

Value

A list of compact arrays and starting parameters.

LKTOptimizeOnlineSimpleAdaptive
Optimize the fast online simple-adaptive model

Description

Optimize the fast online simple-adaptive model

Usage

```
LKTOptimizeOnlineSimpleAdaptive(
  input,
  start = NULL,
  lower = c(-Inf, -Inf, -Inf, -Inf, 0, 0),
  upper = rep(Inf, 6),
  control = list(maxit = 100),
  require_native = FALSE,
  use_gradient = TRUE
)
```

Arguments

input	Output from LKTOnlineSimpleAdaptiveInput().
start	Optional six-parameter starting vector.
lower	Lower bounds for L-BFGS-B.
upper	Upper bounds for L-BFGS-B.
control	Optimizer control list.
require_native	Logical; stop if the compiled native evaluator is not loaded.
use_gradient	Logical; pass the propagated gradient to the optimizer.

Value

A list containing the optimizer result and fit details.

```
LKTOptimizeOnlineSimpleAdaptiveAlpha
```

Optimize only the online learning rates

Description

Optimizes the recency and logsuc online learning rates while holding the global simple-adaptive coefficients fixed at their LibLinear optimum.

Usage

```
LKTOptimizeOnlineSimpleAdaptiveAlpha(
  input,
  start = c(alpha_recency = 0.03, alpha_logsuc = 0.05),
  lower = c(0, 0),
  upper = rep(Inf, 2),
  control = list(maxit = 100),
  require_native = FALSE,
  use_gradient = TRUE
)
```

Arguments

input	Output from LKTOnlineSimpleAdaptiveInput().
start	Optional two-parameter starting vector: recency alpha and logsuc alpha.
lower	Lower bounds for L-BFGS-B.
upper	Upper bounds for L-BFGS-B.
control	Optimizer control list.
require_native	Logical; stop if the compiled native evaluator is not loaded.
use_gradient	Logical; pass the propagated alpha gradient to the optimizer.

Value

A list containing the optimizer result and fit details.

LKTOptimizeOnlineSimpleAdaptiveDecayAlpha

Optimize online beta and decay learning rates

Description

Optimizes online learning rates while holding global coefficients and the global starting recency decay fixed.

Usage

```
LKTOptimizeOnlineSimpleAdaptiveDecayAlpha(
  input,
  start = c(alpha_recency = 0.03, alpha_logsuc = 0.05, alpha_decay = 0.01),
  lower = c(0, 0, 0),
  upper = rep(Inf, 3),
  control = list(maxit = 100),
  require_native = FALSE,
  use_gradient = FALSE,
  decay_lower = 1e-05,
  decay_upper = 0.99999
)
```

Arguments

input	Output from LKTOonlineSimpleAdaptiveInput().
start	Starting vector for recency beta alpha, logsuc beta alpha, and recency-decay alpha.
lower	Lower bounds for L-BFGS-B.
upper	Upper bounds for L-BFGS-B.
control	Optimizer control list.
require_native	Logical; stop if the compiled native evaluator is not loaded.
use_gradient	Reserved for future analytic-gradient support.
decay_lower	Lower bound for the within-subject recency decay state.
decay_upper	Upper bound for the within-subject recency decay state.

Value

A list containing the optimizer result and fit details.

OnlineAdaptiveModel	<i>Define the online adaptive LKT model</i>
---------------------	---

Description

Creates an LKT model adapter that first fits the standard LibLinear model, then applies an online parameter-adaptation layer.

Usage

```
OnlineAdaptiveModel(  
  alpha_recency_start = 0.03,  
  alpha_logsuc_start = 0.05,  
  alpha_decay_start = 0.01,  
  online_mode = "alpha_only",  
  beta_alpha_terms = NULL,  
  nonlinear_alpha_terms = character(0),  
  alpha_start = NULL,  
  alpha_lower = -1,  
  alpha_upper = 1,  
  nonlinear_lower = 1e-05,  
  nonlinear_upper = 0.99999,  
  maxit = 100,  
  factr = 1e+07,  
  optim_trace = 0,  
  optim_report = 1,  
  require_native = FALSE,  
  use_gradient = TRUE  
)
```

```
FastOnlineSimpleAdaptiveModel(  
  alpha_recency_start = 0.03,  
  alpha_logsuc_start = 0.05,  
  alpha_decay_start = 0.01,  
  online_mode = "alpha_only",  
  beta_alpha_terms = NULL,  
  nonlinear_alpha_terms = character(0),  
  alpha_start = NULL,  
  alpha_lower = -1,  
  alpha_upper = 1,  
  nonlinear_lower = 1e-05,  
  nonlinear_upper = 0.99999,  
  maxit = 100,  
  factr = 1e+07,  
  optim_trace = 0,  
  optim_report = 1,  
  require_native = FALSE,
```

```

        use_gradient = TRUE
    )

```

Arguments

<code>alpha_recency_start</code>	Starting value for the recency learning rate.
<code>alpha_logsuc_start</code>	Starting value for the logsuc learning rate.
<code>alpha_decay_start</code>	Legacy starting value for the recency decay learning rate.
<code>online_mode</code>	Online optimizer mode. "alpha_only" uses the generalized adaptive alpha layer. "six_parameter" and "decay_alpha" retain the named historical model behaviors.
<code>beta_alpha_terms</code>	Coefficient names whose beta values are updated online.
<code>nonlinear_alpha_terms</code>	Nonlinear parameter keys formatted as feature component slot.
<code>alpha_start</code>	Optional numeric starting vector for selected beta and nonlinear learning rates.
<code>alpha_lower</code>	Scalar or vector lower bounds for learning-rate optimization.
<code>alpha_upper</code>	Scalar or vector upper bounds for learning-rate optimization.
<code>nonlinear_lower</code>	Lower clamp for online nonlinear parameter states.
<code>nonlinear_upper</code>	Upper clamp for online nonlinear parameter states.
<code>maxit</code>	Maximum optimizer iterations.
<code>factr</code>	Optimizer factr control value.
<code>optim_trace</code>	Optimizer trace level passed to <code>stats::optim()</code> .
<code>optim_report</code>	Optimizer reporting interval passed to <code>stats::optim()</code> .
<code>require_native</code>	Logical; stop if the compiled native evaluator is not loaded.
<code>use_gradient</code>	Logical; pass the propagated gradient to the optimizer.

Value

A model adapter suitable for LKT(`model = ...`).

OnlineCalibrationModel

Define the online calibration LKT model

Description

Creates an LKT model adapter that first solves global feature weights with the standard static LKT/LibLinear model, then uses those weights as the starting beta vector for every subject. Within each subject, the full beta vector is updated online after each observed response: $\text{beta}[j, t + 1] = \text{beta}[j, t] + \alpha * x[j, t] * (y[t] - p[t]) / (x[j, t]^2 + \text{denom_epsilon})$ for each online-updated coefficient j . With $\alpha = 0$, this model reproduces the static global-beta predictions.

Usage

```
OnlineCalibrationModel(
  alpha_start = 0,
  alpha_lower = 0,
  alpha_upper = 10,
  beta_start = "liblinear",
  beta_lower = -Inf,
  beta_upper = Inf,
  optimize_alpha = TRUE,
  optimize_beta = FALSE,
  clip_epsilon = 1e-06,
  denom_epsilon = clip_epsilon,
  subject_col = "Anon.Student.Id",
  fixed_online_coefficients = c("(Intercept)", "intercept"),
  update_strategy = c("all", "best_loss"),
  maxit = 100,
  factr = 1e+07
)
```

Arguments

<code>alpha_start</code>	Initial learning-rate value.
<code>alpha_lower</code>	Lower bound for the learning rate.
<code>alpha_upper</code>	Upper bound for the learning rate.
<code>beta_start</code>	Initial value for feature weights. Use "liblinear" to initialize from the standard static LKT fit. Numeric scalars are recycled.
<code>beta_lower</code>	Lower bound for feature weights. A scalar is recycled.
<code>beta_upper</code>	Upper bound for feature weights. A scalar is recycled.
<code>optimize_alpha</code>	Logical; optimize the learning rate when TRUE.
<code>optimize_beta</code>	Reserved for future joint optimization of the initial beta vector. The implemented model solves the initial beta vector globally before optimizing alpha.

clip_epsilon	Probability clipping value used before log likelihoods.
denom_epsilon	Small denominator stabilizer for online updates.
subject_col	Subject identifier column used for within-subject updates.
fixed_online_coefficients	Coefficients that remain fixed during within-subject online updates. By default, the global intercept is kept fixed when present.
update_strategy	Online update rule. "all" updates every eligible active coefficient on the trial. "best_loss" tries each eligible one-coefficient update and applies only the update that most reduces that trial's immediate logistic loss.
maxit	Maximum optimizer iterations.
factr	Optimizer factr control value.

Value

A model adapter suitable for LKT(model = ...).

Examples

```
adapter <- OnlineCalibrationModel(maxit = 5)
adapter$name
```

predict_lkt	<i>Predict with an LKT model object</i>
-------------	---

Description

Generates predicted probabilities from a fitted LKT() object and, optionally, computes common evaluation statistics for selected folds.

Usage

```
predict_lkt(
  modelob,
  data,
  fold = NULL,
  return_stats = FALSE,
  min_pred_limit = 1e-05,
  max_pred_limit = 0.99999
)
```

Arguments

<code>modelob</code>	Fitted LKT model object containing coefficients and predictors.
<code>data</code>	Data frame or data table containing <code>CF..ansbin.</code> and, when fold filtering is requested, a <code>fold</code> column.
<code>fold</code>	Optional vector of fold values to include in the returned predictions and statistics. <code>NULL</code> or an empty vector uses all rows.
<code>return_stats</code>	Logical; if <code>TRUE</code> , return predictions plus log-likelihood, AUC, RMSE, and McFadden-style R-squared. If <code>FALSE</code> , return only predictions.
<code>min_pred_limit</code>	Minimum probability used to clamp predictions.
<code>max_pred_limit</code>	Maximum probability used to clamp predictions.

Value

If `return_stats` is `FALSE`, returns a list containing:

- `predictions`: The predicted probabilities for each observation in the specified fold(s).

If `return_stats` is `TRUE`, returns a list containing:

- `predictions`: The predicted probabilities for each observation in the specified fold(s).
- `LL`: Log-Likelihood of the model given the actual outcomes.
- `AUC`: Area Under the ROC Curve.
- `RMSE`: Root Mean Squared Error.
- `R2`: R-squared value, indicating the proportion of variance explained by the model.

Examples

```
data(samplelkt)
samplelkt$CF..ansbin. <- as.integer(samplelkt$Outcome == "CORRECT")
model <- LKT(
  data = samplelkt,
  components = "KC..Default.",
  features = "intercept",
  inter = TRUE,
  verbose = FALSE
)
preds <- predict_lkt(model, samplelkt)
head(preds$predictions)
```

samplelkt	<i>Sample practice trial sequences</i>
-----------	--

Description

A small trial-level dataset from a practice memory experiment.

Usage

```
samplelkt
```

Format

A data frame with 2074 rows and many variables:

Anon.Student.Id unique identifier for each student

Duration..sec. response duration in seconds

KC..Default. default knowledge component label

Outcome trial outcome label

Source

<https://pslcdatashop.web.cmu.edu/DatasetInfo?datasetId=5508>

smallSet	<i>Sample a subset of students</i>
----------	------------------------------------

Description

Legacy exported helper that samples up to nSub students and returns all of their rows. It is retained because LKT_HDI() and older examples use it.

Usage

```
smallSet(data, nSub)
```

Arguments

data Data frame or data table containing Anon.Student.Id.

nSub Number of students to sample.

Value

A data frame containing all rows for the sampled students.

Examples

```
data(samplelkt)
subset_data <- smallSet(samplelkt, 2)
length(unique(subset_data$Anon.Student.Id))
```

ViewExcel*Open a data frame in the system spreadsheet viewer*

Description

Legacy exported convenience helper that writes a data frame to a temporary CSV file and opens it with the operating system's default application where supported. It is intended for interactive use, not package automation.

Usage

```
ViewExcel(df = .Last.value, file = tempfile(fileext = ".csv"))
```

Arguments

df	Data frame to write. Defaults to <code>.Last.value</code> .
file	File path for the temporary CSV output.

Value

The value returned by the operating-system file opener.

Examples

```
## Not run:
ViewExcel(head(samplelkt))

## End(Not run)
```

Index

* datasets

- largerawsample, [7](#)
- samplelkt, [28](#)

buildLKTModel, [2](#)

computefeatures, [4](#)

computeSpacingPredictors, [5](#)

countOutcomeold, [6](#)

FastOnlineSimpleAdaptiveModel
(OnlineAdaptiveModel), [23](#)

largerawsample, [7](#)

LASSOLKTData, [7](#)

LASSOLKTModel, [9](#)

LibLinearModel, [10](#)

LKT, [11](#)

LKT_HDI, [13](#)

LKTCustomModel, [14](#)

LKTFeatureInput, [15](#)

LKTModelFit, [16](#)

LKTOnlineSimpleAdaptiveDecayEval, [17](#)

LKTOnlineSimpleAdaptiveEval, [18](#)

LKTOnlineSimpleAdaptiveGradient, [18](#)

LKTOnlineSimpleAdaptiveInput, [19](#)

LKTOptimizeOnlineSimpleAdaptive, [20](#)

LKTOptimizeOnlineSimpleAdaptiveAlpha,
[21](#)

LKTOptimizeOnlineSimpleAdaptiveDecayAlpha,
[22](#)

OnlineAdaptiveModel, [23](#)

OnlineCalibrationModel, [25](#)

predict_lkt, [26](#)

samplelkt, [28](#)

smallSet, [28](#)

ViewExcel, [29](#)