# Package 'LKT'

November 29, 2021

**Title** Logistic Knowledge Tracing

**Version** 1.0.1

**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 Harrel-Williams (2021) <https://ieeexplore.ieee.org/document/9616435>. '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

**RoxygenNote** 7.1.2

**Depends** R (>= 3.5.0), SparseM (>= 1.78), Matrix, methods, data.table (>= 1.13.2), LiblineaR (>= 2.10-8)

**Imports** glmnet (>= 4.0-2), glmnetUtils (>= 1.1.8), lme4 (>= 1.1-23)

**Suggests** rmarkdown, pROC (>= 1.16.2), knitr, caret, utils

**NeedsCompilation** no

**Author** Philip I. Pavlik Jr. [aut, ctb, cre] (<https://orcid.org/0000-0001-6467-9452>), Luke G. Eglington [aut, ctb] (<https://orcid.org/0000-0002-8432-9203>)

**Maintainer** Philip I. Pavlik Jr. <imrryr@gmail.com>

**Repository** CRAN

**Date/Publication** 2021-11-29 07:40:02 UTC

## R topics documented:

---

`computefeatures` *computefeatures*

---

## Description

Compute feature describing prior practice effect.

## Usage

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

## Arguments

| | |
|---|---|
| `data` | copy of main data frame. |
| `feat` | is the feature to be computed. |
| `par1` | nonlinear parameters used for nonlinear features. |
| `par2` | nonlinear parameters used for nonlinear features. |
| `index` | a student by component levels index |
| `index2` | a component levels index |
| `par3` | nonlinear parameters used for nonlinear features. |
| `par4` | nonlinear parameters used for nonlinear features. |
| `par5` | nonlinear parameters used for nonlinear features. |
| `fcomp` | the component name. |

## Value

a vector suitable for regression input.

computeSpacingPredictors *computeSpacingPredictors*

## Description

Compute repetition spacing time based features from input data CF..Time. and/or CF..reltime. which will be automatically computed from Duration..sec. if not present themselves.

## Usage

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

## Arguments

| | |
|---|---|
| data | is a dataset with Anon.Student.Id and CF..ansbin. |
| KCs | are the components for which spaced features will be specified in LKT |

## Value

data which is the same frame with the added spacing relevant columns.

countOutcome *countOutcome*

## Description

Compute the prior sum of the response appearing in the outcome column for the index

## Usage

```
countOutcome(data, index, response)
```

## Arguments

| | |
|---|---|
| data | the dataset to compute an outcome vector for |
| index | the subsets to count over |
| response | the actually response value being counted |

## Value

the vector of the lagged cumulative sum.

LKT *LKT*

## Description

Compute a logistic regression model of learning for input data.

## Usage

```
LKT(
  data,
  components,
  features,
  fixedpars = NA,
  seedpars = NA,
  covariates = NA,
  dualfit = FALSE,
  interc = FALSE,
  cv = FALSE,
  elastic = FALSE,
  verbose = TRUE,
  epsilon = 1e-04,
  cost = 512,
  type = 0,
  maketimes = FALSE,
  bias = 0
)
```

## Arguments

| | |
|---|---|
| data | A dataset with Anon.Student.Id and CF..ansbin. |
| components | A vector of factors that can be used to compute each features for each subject. |
| features | a vector methods to use to compute a feature for the component. |
| fixedpars | a vector of parameters for all features+components. |
| seedpars | a vector of parameters for all features+components to seed non-linear parameter search. |
| covariates | A list of components that interacts with component by feature in the main specification. |
| dualfit | TRUE or FALSE, fit a simple latency using logit. Requires Duration..sec. column in data. |
| interc | TRUE or FALSE, include a global intercept. |
| cv | TRUE or FALSE, if TRUE runs N-fold cv. Requires premade column named 'fold' with integers denoting the N folds |
| elastic | glmnet, cv.glmnet, cva.glmnet or FALSE. |

| | |
|---|---|
| verbose | provides more output in some cases. |
| epsilon | passed to LiblineaR |
| cost | passsed to LiblineaR |
| type | passsed to LiblineaR |
| maketimes | Boolean indicating whether to create time based features (or may be precomputed) |
| bias | passsed to LiblineaR |

## Value

list of values "model", "coefs", "r2", "prediction", "nullmodel", "latencymodel", "optimizedpars","subjectrmse", "newdata", and "loglike"

## Examples

```
temp <- samplelkt
temp$CF..ansbin.<-ifelse(temp$Outcome=="CORRECT",1,ifelse(temp$Outcome=="INCORRECT",0,-1))
temp <- data.table::setDT(temp)
temp <- computeSpacingPredictors(temp, "KC..Default.")
temp <- temp[temp$CF..ansbin.==0 | temp$CF..ansbin.==1,]
temp$KC..Default.<-substr(temp$KC..Default.,1,10)
modelob <- LKT(
  data = temp, interc=TRUE,
  components = c("Anon.Student.Id", "KC..Default.", "KC..Default."),
  features = c("logitdec", "logitdec", "lineafm"),
  fixedpars = c(.9, .85)
)
print(modelob$coefs)
print(modelob$loglik)

modelob <- LKT(
  data = temp, interc=TRUE,
  components = c("Anon.Student.Id", "KC..Default.", "KC..Default."),
  features = c("logitdec", "logitdec", "lineafm"),
  seedpars = c(.9, .85)
)
print(modelob$coefs)
print(modelob$loglik)

modelob <- LKT(
  data = temp, interc=TRUE,
  components = c("Anon.Student.Id", "KC..Default.", "KC..Default."),
  features = c("logitdec", "logitdec$", "lineafm$"),
  fixedpars = c(.9, .85)
)
print(modelob$coefs)
print(modelob$loglik)

# this example illustrates how mean fit is worse for CV
# compared to the first example above. In this case,
```

```
# this is mainly do to the small dataset allowing overgeneralization
# despite the model only having 4 coefficients
temp <- samplelkt
unq <- sample(unique(temp$Anon.Student.Id))
sfold <- rep(1:5,length.out=length(unq))
temp$fold <- rep(0,length(temp[,1]))
for(i in 1:5){temp$fold[which(temp$Anon.Student.Id %in% unq[which(sfold==i)])]=i}
modelob <- LKT(
     data = temp, interc=TRUE,
      components = c("Anon.Student.Id", "KC..Default.", "KC..Default."),
      features = c("logitdec", "logitdec", "lineafm"),
      fixedpars = c(.9, .85),cv=TRUE
  )
print(modelob$cv_res)
print(mean(modelob$cv_res$rmse))
print(mean(modelob$cv_res$mcfad))

# this example illustrates the limitation of CV when data does not contain
# sufficient examples of each predictor
#modelob <- LKT(
#  data = temp, interc=TRUE,
#  components = c("Anon.Student.Id", "KC..Default.", "KC..Default."),
#  features = c("logitdec", "logitdec$", "lineafm$"),
#  fixedpars = c(.9, .85),cv=TRUE
#)
#print(modelob$cv_res)
```

---

`samplelkt` *Trial sequences for practice participants.*

---

## Description

A dataset containing a small sample of participants in a memory experiment.

## Usage

```
samplelkt
```

## Format

A data frame with 2074 rows and many variables:

**Anon.Student.Id** unique identifier for each student

**Duration..sec.** unique identifier for each student

**KC..Default.** unique identifier for each student

**Outcome** unique identifier for each student ...

## Source

https://datashop.memphis.edu/index.jsp

---

`smallSet` *smallSet*

---

## Description

smallSet

## Usage

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

## Arguments

| | |
|---|---|
| `data` | Dataframe of student data |
| `nSub` | Number of students |

# Index