Package 'LNIRT'

October 12, 2022

Type Package

Title LogNormal Response Time Item Response Theory Models

Version 0.5.1

Author Jean-Paul Fox, Konrad Klotzke, Rinke Klein Entink

Maintainer Konrad Klotzke <omd.bms.utwente.stats@gmail.com>

Imports MASS, methods, stats, utils

Depends R (>= 2.10)

Suggests coda, mcmcse

Description Allows the simultaneous analysis of responses and response times in an Item Response Theory (IRT) modelling framework. Supports variable person speed functions (intercept, trend, quadratic), and covariates for item and person (random) parameters. Data missing-by-design can be specified. Parameter estimation is done with a MCMC algorithm. LNIRT replaces the package CIRT, which was written by Rinke Klein Entink. For reference, see the paper by Fox, Klein Entink and Van der Linden (2007), ``Modeling of Responses and Response Times with the Package cirt", Journal of Statistical Software, <doi:10.18637/jss.v020.i07>.

License GPL-3

Encoding UTF-8

LazyData TRUE

RoxygenNote 7.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2021-01-19 18:50:09 UTC

R topics documented:

`.onAttach` *.onAttach start message*

Description

.onAttach start message

Usage

```
.onAttach(libname, pkgname)
```

Arguments

`libname`	defunct
`pkgname`	defunct

Value

invisible()

`AmsterdamChess` *Amsterdam Chess Test (ACT) data*

Description

Responses and response time data from the Amsterdam Chess Test (ACT).

Usage

```
data(AmsterdamChess)
```

Format

A dataframe with 259 rows and 81 variables.

Details

Variables:

- ELO: Standardized ELO rating (numeric)
- Y1-Y40: item correct score (1 or 0) for scored items 1 – 40 (numeric)
- RT1-RT40: response time (seconds) for scored items 1 – 40 (numeric)

Three components of chess expertise are measured:

- Tactical skill (20 items): item 1-20
- Positional skill (10 items): item 21-30
- End-game skill (10 items): item 31-40

References

van der Maas, H. L., & Wagenmakers, E. J. (2005). A psychometric analysis of chess expertise. The American journal of psychology, 118(1), 29-60. (PubMed)

Examples

```
## Not run:

  ###
  ### EXAMPLE APPLICATION AMSTERDAM CHESS DATA van der Maas and Wagenmakers (2005).
  ###

  library(LNIRT)
  data(AmsterdamChess)
  head(AmsterdamChess)

  N <- nrow(AmsterdamChess)
  Y <-as.matrix(AmsterdamChess[c(which(colnames(AmsterdamChess)=="Y1")
                                 :which(colnames(AmsterdamChess)=="Y40"))])
  K <- ncol(Y)
  ## replace missing 9 with NA
  Y[Y==9] <- NA
  ## Test takers with NAs
  ## Y[147,]
  ## Y[201,]
  ## Y[209,]

  RT <- as.matrix(AmsterdamChess[c(which(colnames(AmsterdamChess)=="RT1")
                                   :which(colnames(AmsterdamChess)=="RT40"))])
  ## replace missing 10000.000 with NA
  RT[RT==10000.000] <- NA
  RT<-log(RT) #logarithm of RTs

  # Define Time Scale
  X <- 1:K
  X <- (X - 1)/K
```

```
set.seed(12345) ## used to obtain the results reported in the paper ##
outchess <- LNIRTQ(Y=Y,RT=RT,X=X,XG=10000)
summary(outchess)

## Check MCMC convergence
##
## check several MCMC chains
##

library(mcmcse)
ess(outchess1$MAB[1001:10000,1,1]) ## effective sample size
mcse(outchess1$MAB[1001:10000,1,1], size = 100
     , g = NULL,method = "bm", warn = FALSE) #standard error

ess(outchess$MAB[1001:10000,1,2]) ## effective sample size
mcse(outchess$MAB[1001:10000,1,2], size = 100
     , g = NULL,method = "bm", warn = FALSE) #standard error

ess(outchess$MAB[1001:10000,1,3]) ## effective sample size
mcse(outchess$MAB[1001:10000,1,3], size = 100
     , g = NULL,method = "bm", warn = FALSE) #standard error

ess(outchess$MAB[1001:10000,1,4]) ## effective sample size
mcse(outchess$MAB[1001:10000,1,4], size = 100
     , g = NULL,method = "bm", warn = FALSE) #standard error

ess(outchess$MSP[1001:10000,1,4]) ## effective sample size
mcse(outchess$MSP[1001:10000,1,4], size = 100
     , g = NULL,method = "bm", warn = FALSE) #standard error

ess(outchess$MSI[1001:10000,1,4]) ## effective sample size
mcse(outchess$MSI[1001:10000,1,4], size = 100
     , g = NULL,method = "bm", warn = FALSE) #standard error

## Convergence Checks
library(coda)
summary(as.mcmc(outchess$MAB[1001:10000,1,1]))
summary(as.mcmc(outchess$MAB[1001:10000,1,4]))
summary(as.mcmc(outchess$MSI[1001:10000,1,1]))
summary(as.mcmc(outchess$MSI[1001:10000,1,4]))
summary(as.mcmc(outchess$MSP[1001:10000,1,4]))
summary(as.mcmc(outchess$MSP[1001:10000,1,1]))

## check some chains on convergence
geweke.diag(as.mcmc(outchess$MAB[1001:10000,1,1]), frac1=0.1, frac2=0.5)
geweke.plot(as.mcmc(outchess$MAB[1001:10000,1,1]), frac1=0.1, frac2=0.5)
heidel.diag(as.mcmc(outchess$MAB[1001:10000,1,1]), eps=0.1, pvalue=0.05)

geweke.diag(as.mcmc(outchess$MSI[1001:10000,1,1]), frac1=0.1, frac2=0.5)
geweke.plot(as.mcmc(outchess$MSI[1001:10000,1,1]), frac1=0.1, frac2=0.5)
heidel.diag(as.mcmc(outchess$MSI[1001:10000,1,1]), eps=0.1, pvalue=0.05)

geweke.diag(as.mcmc(outchess$MSP[1001:10000,3,3]), frac1=0.1, frac2=0.5)
```

```
geweke.plot(as.mcmc(outchess$MSP[1001:10000,3,3]), frac1=0.1, frac2=0.5)
heidel.diag(as.mcmc(outchess$MSP[1001:10000,3,3]), eps=0.1, pvalue=0.05)

## complete missing data
outchess$Mtheta[147,]

#########################################################################
### THIS PART IS NOT DISCUSSED IN THE PAPER                          ###
#########################################################################

# PLOT PERSON PARAMETERS
# ABILITY VS SPEED

par(mar=c(5, 5, 2,4), xpd=F)
layout(matrix(c(1,2,3,4), 2, 2, byrow = TRUE), widths=c(2,2), heights=c(2,2))

plot(outchess$Mtheta[,1],outchess$Mtheta[,2]
     ,xlab=expression(paste("Ability"~~(theta)))
     ,ylab=expression(paste("Intercept"~~(zeta[0]))),
     xlim=c(-2,2),ylim=c(-1,1),bty="l",cex.lab=1.5,cex.axis=1.25)
abline(lm(outchess$Mtheta[,2]~outchess$Mtheta[,1]))
abline(h = 0,lty = 2)

plot(outchess$Mtheta[,1],outchess$Mtheta[,3]
     ,xlab=expression(paste("Ability"~~(theta)))
     ,ylab=expression(paste("Linear slope"~~(zeta[1]))),
     xlim=c(-2,2),ylim=c(-1,1),bty="l",cex.lab=1.5,cex.axis=1.25)
abline(lm(outchess$Mtheta[,3]~outchess$Mtheta[,1]))
abline(h = 0,lty = 2)

plot(outchess$Mtheta[,1],outchess$Mtheta[,4]
     ,xlab=expression(paste("Ability"~~(theta)))
     ,ylab=expression(paste("Quadratic slope"~~(zeta[2]))),
     xlim=c(-2,2),ylim=c(-1,1),bty="l",cex.lab=1.5,cex.axis=1.25)
abline(lm(outchess$Mtheta[,4]~outchess$Mtheta[,1]))
abline(h = 0,lty = 2)

plot(outchess$Mtheta[,3],outchess$Mtheta[,4]
     ,xlab=expression(paste("Linear slope"~~(zeta[1])))
     ,ylab=expression("Quadratic slope"~~paste((zeta[2]))),
     xlim=c(-0.5,1),ylim=c(-0.5,0.5),bty="l",cex.lab=1.5,cex.axis=1.25)
abline(lm(outchess$Mtheta[,4]~outchess$Mtheta[,3]))
abline(h = 0,lty = 2)

### include residual analysis ###

set.seed(12345)
outchessr <- LNIRTQ(Y=Y,RT=RT,X=X,XG=10000,burnin=10,XGresid=1000,resid=TRUE)
summary(outchessr)

## plot of person-fit scores for RT patterns
plot(outchessr$lZPT,outchessr$lZP)
```

```
## End(Not run)
```

`CredentialForm1` *Credential Form data*

Description

Responses and response time data from the credential data set of Cizek and Wollack (2016).

Usage

```
data(CredentialForm1)
```

Format

A dataframe with 1636 rows and 610 variables.

Details

Variables:

- EID: Examinee ID (character)
- FormID: Test form name (character)
- Flagged: 1/0 variable to indicate whether the test vendor suspects the examinee may have engaged in inappropriate behavior (numeric)
- Pretest: Pretest item set assigned to candidate (numeric)
- Attempt: Count of the attempt number for the candidate. A score of 1 indicates that candidate is a new, first-time examinee. Any examinee sitting for the exam for the fourth time or more is marked as 4+ (character)
- Country: Country where candidate was educated (character)
- StateCode: 2-digit code corresponding to the state in which the Candidate applied for licensure (numeric)
- School_ID: 4-digit code corresponding to the particular institution in which the Candidate received his/her educational training (numeric)
- Cent_id: 4-digit code corresponding to the particular testing center in which the Candidate sat for the exam (numeric)
- Tot_time: The number of seconds testing (numeric)
- iresp.1-170: item responses (1 to 4 or NA) for scored items 1 – 170 (numeric)
- iresp.171-180: item responses (1 to 4 or NA) for 10 pilot items for pilot set 6 or 9 (numeric)
- iresp.181-190: item responses (1 to 4 or NA) for 10 pilot items for pilot set 7 or 10 (numeric)
- iresp.191-200: item responses (1 to 4 or NA) for 10 pilot items for pilot set 8 or 11 (numeric)
- iraw.1-170: item correct score (1 or 0) for scored items 1 – 170 (numeric)

- iraw.171-180: item correct score (1 or 0) for 10 pilot items for pilot set 6 or 9 (numeric)
- iraw.181-190: item correct score (1 or 0) for 10 pilot items for pilot set 7 or 10 (numeric)
- iraw.191-200: item correct score (1 or 0) for 10 pilot items for pilot set 8 or 11 (numeric)
- idur.1-170: response time (in seconds) for scored items 1 – 170 (numeric)
- idur.171-180: response time (in seconds) for 10 pilot items for pilot set 6 or 9 (numeric)
- idur.181-190: response time (in seconds) for 10 pilot items for pilot set 7 or 10 (numeric)
- idur.191-200: response time (in seconds) for 10 pilot items for pilot set 8 or 11 (numeric)

References

Cizek GJ, Wollack JA (eds.) (2016). Handbook of Quantitative Methods for Detecting Cheating on Tests. Routledge. (Taylor&Francis)

Examples

```
## Not run:

  ###
  ### EXAMPLE APPLICATION CREDENTIAL FORM1 DATA CIZEK and WOLLACK (2016).
  ###

  library(LNIRT)
  data(CredentialForm1)

  ### DATA OBJECTS FOR LNIRT
  ### RA Data
  Y <- as.matrix(CredentialForm1[c(which(colnames(CredentialForm1)=="iraw.1")
                                   :which(colnames(CredentialForm1)=="iraw.170"))])
  N <- nrow(Y)

  ### RT Data
  RT<-as.matrix(CredentialForm1[c(which(colnames(CredentialForm1)=="idur.1")
                                   :which(colnames(CredentialForm1)=="idur.170"))])
  RT[RT==0]<-NA ## zero RTs are coded as missing values
  RT<-log(RT) ## logarithmic transformation of RT

  ## RUN LNIRT MODEL 0
  set.seed(12345) ## used to obtain the results reported in the paper ##
  out0 <- LNIRT(RT=RT,Y=Y,XG=5000,burnin=10,ident=2)
  summary(out0)

  ## Check MCMC convergence

  library(mcmcse)
  ##
  ## check several MCMC chains
  ##
  ## effective sample size and effective sample size
  ess(out0$MCMC.Samples$Cov.Person.Ability.Speed[1001:5000]) ## effective sample size
  mcse(out0$MCMC.Samples$Cov.Person.Ability.Speed[1001:5000]
```

```
     , size = 100, g = NULL,method = "bm", warn = FALSE) #standard error

ess(out0$MCMC.Samples$Var.Person.Ability[1001:5000]) ## effective sample size
mcse(out0$MCMC.Samples$Var.Person.Ability[1001:5000]
     , size = 100, g = NULL,method = "bm", warn = FALSE) #standard error

ess(out0$MCMC.Samples$Var.Person.Speed[1001:5000]) ## effective sample size
mcse(out0$MCMC.Samples$Var.Person.Speed[1001:5000]
     , size = 100, g = NULL,method = "bm", warn = FALSE) #standard error

ess(out0$MCMC.Samples$Item.Discrimination[1001:5000,155]) ## effective sample size
mcse(out0$MCMC.Samples$Item.Discrimination[1001:5000,155]
     , size = 100, g = NULL,method = "bm", warn = FALSE) #standard error

ess(out0$MCMC.Samples$Time.Discrimination[1001:5000,1]) ## effective sample size
mcse(out0$MCMC.Samples$Time.Discrimination[1001:5000,1]
     , size = 100, g = NULL,method = "bm", warn = FALSE) #standard error

ess(out0$MCMC.Samples$Person.Ability[1001:5000,1]) ## effective sample size
mcse(out0$MCMC.Samples$Person.Ability[1001:5000,1]
     , size = 100, g = NULL,method = "bm", warn = FALSE) #standard error

ess(out0$MCMC.Samples$Person.Speed[1001:5000,1]) ## effective sample size
mcse(out0$MCMC.Samples$Person.Speed[1001:5000,1]
     , size = 100, g = NULL,method = "bm", warn = FALSE) #standard error

## Convergence Checks
library(coda)
summary(as.mcmc(out0$MCMC.Samples$Cov.Person.Ability.Speed[1001:5000]))
summary(as.mcmc(out0$MCMC.Samples$Var.Person.Ability[1001:5000]))
summary(as.mcmc(out0$MCMC.Samples$Var.Person.Speed[1001:5000]))
summary(as.mcmc(out0$MCMC.Samples$Item.Discrimination[1001:5000,155]))
summary(as.mcmc(out0$MCMC.Samples$Time.Discrimination[1001:5000,1]))
summary(as.mcmc(out0$MCMC.Samples$Person.Ability[1001:5000,1]))
summary(as.mcmc(out0$MCMC.Samples$Person.Speed[1001:5000,1]))

## check some chains on convergence
geweke.diag(as.mcmc(out0$MCMC.Samples$Cov.Person.Ability.Speed[500:5000]), frac1=0.1, frac2=0.5)
geweke.plot(as.mcmc(out0$MCMC.Samples$Cov.Person.Ability.Speed[500:5000]), frac1=0.1, frac2=0.5)
heidel.diag(as.mcmc(out0$MCMC.Samples$Cov.Person.Ability.Speed[500:5000], eps=0.1, pvalue=0.05))

geweke.diag(as.mcmc(out0$MCMC.Samples$Item.Discrimination[500:5000,155]), frac1=0.1, frac2=0.5)
geweke.plot(as.mcmc(out0$MCMC.Samples$Item.Discrimination[500:5000,155]), frac1=0.1, frac2=0.5)
heidel.diag(as.mcmc(out0$MCMC.Samples$Item.Discrimination[500:5000,155]), eps=0.1, pvalue=0.05)

geweke.diag(as.mcmc(out0$MCMC.Samples$Person.Ability[500:5000,1]), frac1=0.1, frac2=0.5)
geweke.plot(as.mcmc(out0$MCMC.Samples$Person.Ability[500:5000,1]), frac1=0.1, frac2=0.5)
heidel.diag(as.mcmc(out0$MCMC.Samples$Person.Ability[500:5000,1]), eps=0.1, pvalue=0.05)

## Item parameter estimates

min(apply(out0$MAB[500:5000,,1],2,mean))
max(apply(out0$MAB[500:5000,,1],2,mean))
```

```
min(apply(out0$MAB[500:5000,,2],2,mean))
max(apply(out0$MAB[500:5000,,2],2,mean))

min(apply(out0$MAB[500:5000,,3],2,mean))
max(apply(out0$MAB[500:5000,,3],2,mean))

min(apply(out0$MAB[500:5000,,4],2,mean))
max(apply(out0$MAB[500:5000,,4],2,mean))

plot(apply(out0$MAB[500:5000,,4],2,mean),(apply(RT,2,mean,na.rm=TRUE)))

### Explanatory Variables Test-takers
XFT <- data.frame(CredentialForm1[1:10],stringsAsFactors=TRUE) #Background Variables
XFT$Tot_time <- (XFT$Tot_time-mean(XFT$Tot_time))/sqrt(var(XFT$Tot_time))

## DUMMY CODING FOR CATEGORICAL PREDICTORS
## Pretest Groups
XFT$Pgroup <- matrix(0,ncol=2,nrow=N)
XFT$Pgroup[XFT$Pretest==6,1] <- -1
XFT$Pgroup[XFT$Pretest==6,2] <- -1
XFT$Pgroup[XFT$Pretest==7,1] <- 1
XFT$Pgroup[XFT$Pretest==8,2] <- 1

## Countries
XFT$Cgroup <- matrix(0,ncol=3,nrow=N)
XFT$Cgroup[XFT$Country=="USA",1] <- 1
XFT$Cgroup[XFT$Country=="Philippines",2] <- 1
XFT$Cgroup[XFT$Country=="India",3] <- 1
XFT$Cgroup[c(XFT$Country!="USA" & XFT$Country!="India" & XFT$Country!="Philippines"),1:3] <- -1

XA <- matrix(unlist(XFT[,c("Pgroup","Tot_time")]),ncol=3,nrow=N)
XT <- matrix(unlist(XFT[,c("Pgroup")]),ncol=2,nrow=N)

## RUN LNIRT MODEL 1 (Pretest and total test time)
## Include residual analysis
set.seed(12345) ## used to obtain the results reported in the paper ##
out1 <- LNIRT(RT=RT,Y=Y,XG=5000,XPA=XA,XPT=XT,residual=TRUE)
summary(out1)


#########################################################################
### THIS PART IS NOT DISCUSSED IN THE PAPER                          ###
#########################################################################

## RUN LNIRT MODEL 2 (Pretest and Country)
XA <- matrix(unlist(XFT[,c("Pgroup","Cgroup")]),ncol=5,nrow=N)
XT <- matrix(unlist(XFT[,c("Pgroup","Cgroup")]),ncol=5,nrow=N)

set.seed(12345) ##
out2 <- LNIRT(RT=RT,Y=Y,XG=5000,XPA=XA,XPT=XT)
summary(out2)
```

```
XA <- matrix(unlist(XFT[,c("Pgroup","Cgroup","Tot_time")]),ncol=6,nrow=N)
XT <- matrix(unlist(XFT[,c("Pgroup","Cgroup")]),ncol=5,nrow=N)

## RUN LNIRT MODEL 3
set.seed(12345) ##
out3 <- LNIRT(RT=RT,Y=Y,XG=5000,XPA=XA,XPT=XT)
summary(out3)

#############################################################################
#############################################################################
#############################################################################


#####################################################################
### THIS PART IS DISCUSSED IN THE PAPER                           ###
#####################################################################

## Subsection "Planned Missing By Design"
## Include pretest item data
MBDM<-matrix(rep(0,1636*200),nrow=1636,ncol=200)
MBDM[XFT$Pretest==6,171:180]<-1
MBDM[XFT$Pretest==7,181:190]<-1
MBDM[XFT$Pretest==8,191:200]<-1
MBDM[,1:170]<-1

Yt <- CredentialForm1[c(which(colnames(CredentialForm1)=="iraw.1")
                        :which(colnames(CredentialForm1)=="iraw.200"))]
## transform pretest data to numeric
Yt[,171:200] <- unlist(lapply(Yt[,171:200]
                         ,function(x) as.numeric(x))) #warnings about NA can be ignored
Yt <- as.matrix(Yt,ncol=200,nrow=1636)

RTt <- (CredentialForm1[as.numeric(c(which(colnames(CredentialForm1)=="idur.1")
                                     :which(colnames(CredentialForm1)=="idur.200")))])
RTt[,171:200] <- unlist(lapply(RTt[,171:200]
           , function(x) as.numeric(as.character(x)))) #warnings about NA can be ignored
RTt[RTt==0] <- NA ## zero RTs are coded as missing values
RTt <- log(RTt) ## logarithmic transformation of RT
RTt <- as.matrix(RTt,ncol=200,nrow=1636)

# To fit the model, item discrimination parameters are restricted to one.
alpha1<-rep(1,200) ### Pre-defined item discrimination parameters
## RUN LNIRT MODEL 4
set.seed(12345) ## used to obtain the results reported in the paper ##
out4 <- LNIRT(RT=RTt,Y=Yt,XG=5000,alpha=alpha1,MBDY=MBDM,MBDT=MBDM)
summary(out4)

### Subsection "Model-Fit Analysis"
### Return to output of out1

#report fit results
summary(out1)
```

```
## estimated average residual variance
mean(out1$Msigma2[500:5000,])

#recoding of number of zero attempts
XFT$Attempt[XFT$Attempt==0] <- 1

## explain heterogeneity in person-fit statistics RA and RT
summary(lm(out1$PFl ~ as.factor(XFT$Attempt)+(XFT$Cgroup)+(XFT$Pgroup)))
summary(lm(out1$lZPT ~ as.factor(XFT$Attempt)+(XFT$Cgroup)+(XFT$Pgroup)))

### overview plot of person fit RA versus person-fit RT per country
dev.new()
plot(out1$PFl,out1$lZPT,xlab="Person-fit Statistic RA",ylab="Person-fit Statistic RT",
     col="black",cex=.5,bty="l",xlim=c(-3,3)
     , ylim=c(0,500),cex.main=.8,cex.axis=.7,cex.lab=.8,pch=15)
## US
set1 <- which(XFT$Country=="USA")
points(out1$PFl[set1],out1$lZPT[set1],col="blue",pch=10,cex=.5)
## India
set2 <- which(XFT$Country=="India")
points(out1$PFl[set2],out1$lZPT[set2],col="red",pch=13,cex=.5)
## Philippines
set3 <- which(XFT$Country=="Philippines")
points(out1$PFl[set3],out1$lZPT[set3],col="green",pch=16,cex=.5)
abline(h = qchisq(.95, df= 170),lty = 2,col="red")
abline(v = qnorm(.95),lty = 2,col="red")
legend(-3,500,c("India","US","Philippines","Other"),
       col=c("red","blue","green","black"),pch = c(13,10,16,15), bg = "gray95",cex=.7)

##############################################################################
##############################################################################
##############################################################################


## End(Not run)
```

LNIRT *Log-normal response time IRT modelling*

Description

Log-normal response time IRT modelling

Usage

```
LNIRT(
  RT,
  Y,
  data,
  XG = 1000,
```

```
  burnin = 10,
  XGresid = 1000,
  guess = FALSE,
  par1 = FALSE,
  residual = FALSE,
  td = TRUE,
  WL = FALSE,
  ident = 2,
  alpha,
  beta,
  phi,
  lambda,
  XPA = NULL,
  XPT = NULL,
  XIA = NULL,
  XIT = NULL,
  MBDY = NULL,
  MBDT = NULL
)
```

Arguments

RT	a Person-x-Item matrix of log-response times (time spent on solving an item).
Y	a Person-x-Item matrix of responses.
data	either a list or a simLNIRT object containing the response time and response matrices and optionally the predictors for the item and person parameters. If a simLNIRT object is provided, in the summary the simulated item and time parameters are shown alongside of the estimates. If the required variables cannot be found in the list, or if no data object is given, then the variables are taken from the environment from which LNIRT is called.
XG	the number of MCMC iterations to perform (default: 1000).
burnin	the percentage of MCMC iterations to discard as burn-in period (default: 10).
XGresid	the number of MCMC iterations to perform before residuals are computed (default: 1000).
guess	include guessing parameters in the IRT model (default: false).
par1	use alternative parameterization (default: false).
residual	compute residuals, >1000 iterations are recommended (default: false).
td	estimate the time-discrimination parameter(default: true).
WL	define the time-discrimination parameter as measurement error variance parameter (default: false).
ident	set identification rule (default: 2).
alpha	an optional vector of pre-defined item-discrimination parameters.
beta	an optional vector of pre-defined item-difficulty parameters.
phi	an optional vector of predefined time discrimination parameters.

lambda	an optional vector of predefined time intensity parameters.
XPA	an optional matrix of predictors for the person ability parameters.
XPT	an optional matrix of predictors for the person speed parameters.
XIA	an optional matrix of predictors for the item-difficulty parameters.
XIT	an optional matrix of predictors for the item-intensity parameters.
MBDY	an optional indicator matrix for response missings due to the test design (0: missing by design, 1: not missing by design).
MBDT	an optional indicator matrix for response time missings due to the test design (0: missing by design, 1: not missing by design).

Value

an object of class LNIRT.

Examples

```
## Not run:
# Log-normal response time IRT modelling
data <- simLNIRT(N = 500, K = 20, rho = 0.8, WL = FALSE)
out <- LNIRT(RT = RT, Y = Y, data = data, XG = 1500, residual = TRUE, WL = FALSE)
summary(out) # Print results
out$Post.Means$Item.Difficulty # Extract posterior mean estimates

library(coda)
mcmc.object <- as.mcmc(out$MCMC.Samples$Item.Difficulty) # Extract MCMC samples for coda
summary(mcmc.object)
plot(mcmc.object)

## End(Not run)
```

LNIRTQ *Log-normal response time IRT modelling with variable person speed (intercept, trend, quadratic)*

Description

Log-normal response time IRT modelling with variable person speed (intercept, trend, quadratic)

Usage

```
LNIRTQ(
  Y,
  RT,
  X,
  data,
  XG = 1000,
  burnin = 10,
```

```
  XGresid = 1000,
  residual = FALSE
)
```

Arguments

Y	a Person-x-Item matrix of responses.
RT	a Person-x-Item matrix of log-response times (time spent on solving an item).
X	explanatory (time) variables for random person speed (default: (1:N.items - 1)/N.items).
data	either a list or a simLNIRTQ object containing the response time and response matrices and optionally the predictors for the item and person parameters. If a simLNIRTQ object is provided, in the summary the simulated item and time parameters are shown alongside of the estimates. If the required variables cannot be found in the list, or if no data object is given, then the variables are taken from the environment from which LNIRTQ is called.
XG	the number of MCMC iterations to perform (default: 1000).
burnin	the percentage of MCMC iterations to discard as burn-in period (default: 10).
XGresid	the number of MCMC iterations to perform before residuals are computed (default: 1000).
residual	compute residuals, >1000 iterations are recommended (default: false).

Value

an object of class LNIRTQ.

LNRT *Log-normal response time modelling*

Description

Log-normal response time modelling

Usage

```
LNRT(
  RT,
  data,
  XG = 1000,
  burnin = 10,
  XGresid = 1000,
  residual = FALSE,
  td = TRUE,
  WL = FALSE,
  XPT = NULL,
  XIT = NULL
)
```

Arguments

RT	a Person-x-Item matrix of log-response times (time spent on solving an item).
data	either a list or a simLNIRT object containing the response time matrix. If a simLNIRT object is provided, in the summary the simulated time parameters are shown alongside of the estimates. If the RT variable cannot be found in the list, or if no data object is given, then the RT variable is taken from the environment from which LNRT is called.
XG	the number of MCMC iterations to perform (default: 1000).
burnin	the percentage of MCMC iterations to discard as burn-in period (default: 10).
XGresid	the number of MCMC iterations to perform before residuals are computed (default: 1000).
residual	compute residuals, >1000 iterations are recommended (default: false).
td	estimate the time-discrimination parameter (default: true).
WL	define the time-discrimination parameter as measurement error variance parameter (default: false).
XPT	an optional matrix of predictors for the person speed parameters.
XIT	an optional matrix of predictors for the item time intensity parameters.

Value

an object of class LNRT.

Examples

```
## Not run:
# Log-normal response time modelling
data <- simLNIRT(N = 500, K = 20, rho = 0.8, WL = FALSE)
out <- LNRT(RT = RT, data = data, XG = 1500, residual = TRUE, td = TRUE, WL = FALSE)
summary(out) # Print results
out$Post.Means$Time.Intensity # Extract posterior mean estimates

library(coda)
mcmc.object <- as.mcmc(out$MCMC.Samples$Time.Intensity) # Extract MCMC samples for coda
summary(mcmc.object)
plot(mcmc.object)

## End(Not run)
```

LNRTQ *Log-normal response time modelling with variable person speed (intercept, trend, quadratic)*

Description

Log-normal response time modelling with variable person speed (intercept, trend, quadratic)

Usage

```
LNRTQ(RT, X, data, XG = 1000, burnin = 10)
```

Arguments

RT	a Person-x-Item matrix of log-response times (time spent on solving an item).
X	explanatory (time) variables for random person speed (default: (1:N.items - 1)/N.items).
data	either a list or a simLNIRTQ object containing the response time matrix. If a simLNIRTQ object is provided, in the summary the simulated time parameters are shown alongside of the estimates. If the RT variable cannot be found in the list, or if no data object is given, then the RT variable is taken from the environment from which LNRTQ is called.
XG	the number of MCMC iterations to perform (default: 1000).
burnin	the percentage of MCMC iterations to discard as burn-in period (default: 10).

Value

an object of class LNRTQ.

---

simLNIRT *Simulate data for log-normal response time IRT modelling*

Description

Simulate data for log-normal response time IRT modelling

Usage

```
simLNIRT(N, K, rho, td = FALSE, WL = FALSE, kpa, kpt, kia, kit)
```

Arguments

N	the number of persons.
K	the number of items.
rho	the correlation between the person ability and person speed parameter.
td	set time-discrimination to one (default: false).
WL	define the time-discrimination parameter as measurement error variance parameter (default: false).
kpa	the number of predictors for the person ability parameters (optional).
kpt	the number of predictors for the person speed parameters (optional).
kia	the number of predictors for the item-difficulty parameters (optional).
kit	the number of predictors for the item time intensity parameters (optional).

Value

an object of class simLNIRT.

`simLNIRTQ` *Simulate data for log-normal response time IRT modelling with variable person speed (intercept, trend, quadratic)*

Description

Simulate data for log-normal response time IRT modelling with variable person speed (intercept, trend, quadratic)

Usage

```
simLNIRTQ(N, K, ...)
```

Arguments

`N`	the number of persons.
`K`	the number of items.
`...`	optional arguments.

Value

an object of class simLNIRTQ.

`summaryIRTQ` *Summary Function for LNIRTQ*

Description

Summary Function for LNIRTQ

Usage

```
summaryIRTQ(out, data)
```

Arguments

`out`	a LNIRTQ object (the fitted model)
`data`	a simLNIRTQ object (the simulated data, optional)

Index