Package ‘tmt’
May 3, 2024

Type Package

Title Estimation of the Rasch Model for Multistage Tests

Version 0.3.4-0

Date 2024-05-02

Author Jan Steinfeld [cre, aut] (<https://orcid.org/0000-0001-9853-8260>), Alexander Robitzsch [aut] (<https://orcid.org/0000-0002-8226-3132>)

Maintainer Jan Steinfeld <jan.d.steinfeld@gmail.com>


BugReports https://github.com/jansteinfeld/tmt/issues


License GPL-3

LazyLoad yes

VignetteBuilder knitr

Depends R (>= 3.0)

Encoding UTF-8

NeedsCompilation yes

Suggests roxygen2, eRm, knitr, prettydoc, psychotools, testthat, markdown, dexterMST

Imports parallel, ggplot2, Rcpp (>= 0.12.0), stats, rlang

LinkingTo Rcpp

RoxygenNote 7.3.1
**Description**


**Details**

In multistage tests different groups of items (modules) are presented to persons depending on their response behavior to previous item groups. Multistage testing is thus a simple form of adaptive testing. If data is collected on the basis of such a multistage design and the items are estimated using the Conditional Maximum Likelihood (CML) method, Glas (1989) <doi:10.3102/10769986013001045> has shown, that the item parameters are biased. Zwitser and Maris (2015) <doi:10.1007/s11336-013-9369-6> showed in their work, that taking the applied multistage design in consideration and including it in the estimation of the item parameters, the estimation of item parameters is not biased using the CML method. Their proposed solution is implemented in our package. MST designs with a probabilistic instead of a deterministic routing rule (see, e.g. Chen, Yamamoto, & von Davier, 2014 <doi:10.1201/b16858>) are not estimated with this method, therefore the proposed solouting is again modified by Steinfeld and Robitzsch (2021) <doi:10.31234/osf.io/ew27f> which is also integrated into this package.

An application example can be found in the vignette by using the following command in the R console:`vignette("introduction_to_tmt")`
Author(s)

Maintainer: Jan Steinfeld <jan.d.steinfeld@gmail.com> (ORCID)

Authors:

- Alexander Robitzsch <robitzsch@leibniz-ipn.de> (ORCID)

References

Description

This function performs a so-called graphical model check on the basis of the previously performed Likelihood Ratio Test [tmt::tmt_lrttest()]. The estimated item parameters of the two groups are plotted against each other. There is the possibility in this function to highlight items, to be excluded items from the plot, and to produce confidence-ellipses if desired.

Usage

tmt_gmc(
  object,
  title = "graphical model check",
  xaxis = NULL,
  yaxis = NULL,
  lim = NULL,
  ellipse = FALSE,
  drop = NULL,
  alpha = 0.05,
  legendtitle = "split criteria",
  info = NULL
)

See Also

Useful links:
- https://jansteinfeld.github.io/tmt/
- https://github.com/jansteinfeld/tmt
- Report bugs at https://github.com/jansteinfeld/tmt/issues
tmt_gmc

Arguments

object object of the function [tmt::tmt_lrttest()]
title of the plot
xaxis description of the x-axis
yaxis description of the y-axis
lim of the plot
ellipse should confidence-ellipse be plotted
drop which items should be excluded from the plot
alpha which alpha should be used for the ellipse
legendtitle Title of the Legend
info vector with further information for the Plot with names of submitted items

Author(s)

Jan Steinfeld

Examples

#########################################################################
# Example of Graphical Model Check
#########################################################################
items <- seq(-3,3,length.out = 16)
names(items) <- paste0("i",1:16)
persons = 500
dat <- tmt:::sim.rm(theta = persons, b = items, seed = 1234)
dat.rm <- tmt_rm(dat)
dat.lrt <- tmt_lrttest(dat.rm, split = "median")

info <- rep("group_a","group_b"), each = 8)
names(info) <- paste0("i",1:16)

drop <- c("i1","i18")

#library(ggplot2)
plot <- tmt_gmc(object = dat.lrt, ellipse = TRUE, info = info, drop = drop, title = "graphical model check", alpha = 0.05, legendtitle = "split criteria")
tmt_lrtest  Computation of Andersen’s Likelihood-Ratio Test

Description
This function applies the Likelihood Ratio Test of Andersen. Note that all persons with raw score equal to "median" are assigned to the lower group in cases of a median split. Is is also allowed to split after "mean" or submit any dichotomous vector as split criteria.

Usage
```
tmt_lrtest(object, split = "median", cores = NULL, se = TRUE, ...)```

Arguments
- **object**: it is necessary to submit an object of the function mst or nmst
- **split**: default is the split criteria "median" of the raw score, optional are "mean" or any dichotomous vector
- **cores**: submit integer of cores you would like to apply
- **se**: logical: if true, the standard error is estimated
- **...**: further arguments for the tmt_rm function

Value
List with following entries
- **data_orig**: Submitted data frame with item responses
- **betapars_subgroup**: List of item parameters (difficulty) for each subgroup
- **se.beta_subgroup**: List of standard errors of the estimated item parameters
- **model**: Used model ((mst) for Rasch model with multistage design)
- **LRvalue**: LR-value
- **df**: Degrees of freedoms for the test statistic
- **pvalue**: P-value of the likelihood ratio test
- **loglik_subgroup**: Log-likelihoods for the subgroups
- **split_subgroup**: List of split vector for each subgroup
- **call**: Submitted arguments for the function (matched call)
- **fitobj**: List of objects from subgroup estimation

Author(s)
Jan Steinfeld
References


See Also
tmt_rm

Examples

```r
# example for tmt_lrtest
#############################################################################
# Example Rasch model and Likelihood Ratio Test
#############################################################################
dat <- tmt:::sim.rm(theta = 100, b = 10, seed = 1111)
dat.rm <- tmt_rm(dat = dat)
dat.lrt <- tmt_lrtest(dat.rm)
summary(dat.lrt)
```

Function to Translate the mstdesign Syntax

Description

This function translates the specified multistage design for different purposes and functions used in this package. It is possible to apply this function on deterministic as well as probabilistic multistage designs with either sequential or cumulative routing. A detailed instruction of the application can be found in the package vignette.

Usage

```r
tmt_mstdesign(
  mstdesign,
  options = c("design", "simulation", "modules", "items")
)
```

Arguments

- `mstdesign`: definition of desired multistage design
- `options`: vector of required output. 'modules' = Matrix with the classification of modules and items. 'simulation' = list of all stages. 'design' = matrix of all branches. 'items' vector of all Items.
Value

List with following entries

- **modules**: Matrix which contains each module with its corresponding items
- **simulation**: List of the multistage design. Each element within the list contains a matrix for each stage
- **design**: Matrix of all possible branches
- **items**: Vector of item names

Author(s)

Jan Steinfeld

Examples

```r
# example for tmt_mstdesign
## Not run:
###################################
# Example-1
###################################
mstdesign <- 
B1 =~ c(i1, i2, i3, i4, i5)  
B2 =~ c(i6, i7, i8, i9, i10)  
B3 =~ c(i11, i12, i13, i14, i15)  
B4 =~ c(i16, i17, i18, i19, i20)  
B5 =~ c(i21, i22, i23, i24, i25)  
B6 =~ c(i26, i27, i28, i29, i30)  

# define branches
b1 := B4(0,2) + B2(0,2) + B1(0,5)  
b2 := B4(0,2) + B2(3,5) + B3(0,5)  
b3 := B4(3,5) + B5(0,2) + B3(0,5)  
b4 := B4(3,5) + B5(3,5) + B6(0,5)  

# for simulation purposes
tmt_mstdesign(mstdesign, options = "simulation")$simulation

# summary of the submitted design
tmt_mstdesign(mstdesign, options = "design")$design

# matrix of all modules with the containing items
tmt_mstdesign(mstdesign, options = "modules")$modules

# vector of all items
tmt_mstdesign(mstdesign, options = "items")$items
```

# list of all four elements
```r
tmt_mstdesign(mstdesign, options = c("design", "simulation", "modules", "items"))
```

## End(Not run)

### Example-2
```r
mstdesign <- "
  B1 =~ paste0('i',1:5)
  B2 =~ paste0('i',6:10)
  B3 =~ paste0('i',11:15)
  B4 =~ paste0('i',16:20)
  B5 =~ paste0('i',21:25)
  B6 =~ paste0('i',26:30)
  # define branches
  b1 := B4(0,2) + B2(0,2) + B1
  b2 := B4(0,2) + B2(3,5) + B3
  b3 := B4(3,5) + B5(0,2) + B3
  b4 := B4(3,5) + B5(3,5) + B6
"
```

designelements <- tmt_mstdesign(mstdesign, 
  options = c("design", "simulation", "modules", "items"))

---

### tmt_msttemplate

**Function to create a template for the multistage design used in tmt**

#### Description

This function creates a template for the definition of multistage designs as required by the estimation function (in multistage design cases). The defines multistage design is then handed over to the function `tmt_mstdesign`. Essentially, these are the modules, rules and path sections. In the formula-based notation, it is also possible to state additional conditions (constraints) that can be found in the data and are reflected in the multistage design.

#### Usage

```r
tmt_msttemplate(formula = NULL, full = TRUE, eval = TRUE)
```

#### Arguments

- **formula**
  - formula for the desired template of a multistage design. If formula is left empty, a matrix as MST design template is generated.
- **full**
  - logical if the modules and rules sections should also be created
- **eval**
  - logical should the text input be evaluated (e.g. 3:6 = c(3, 4, 5, 6))

#### Author(s)

Jan Steinfeld
tmt_rm

Estimation (CML) of the Rasch model with or without multistage designs.

Description

The tmt_rm function estimates the Rasch model. If the data are collected based on a multistage design (see Zwitser and Maris, 2015) the specific multistage design mstdesign has to be submitted.

Usage

```r
tmt_rm(
  dat,
  mstdesign = NULL,
  weights = NULL,
  start = NULL,
  sum0 = TRUE,
  se = TRUE,
  optimization = "nlminb",
  ...
)
```

Arguments

dat a matrix of dichotomous (0/1) data or a list of the function tmt_designsim
tmt_rm

mstdesign  Model for the multistage design, if CML estimation without multistage designs

weights  is optional for the weights of cases

start  Vector of start values. If no vector is provided, the start values will be automatic

sum0  logical: If the item parameters should be normed to 'sum = 0' as recommended

se  logical: should the standard error should be estimated?

optimization  character: Per default 'nlminb' is used but 'optim' is also supported.

...  optional further arguments for optim and nlminb use control = list() with argu-

Details


is biased if the data is collected in multistage designs and this design is not considered. Zwitser

and Maris (2015) <doi:10.1007/s11336-013-9369-6> propose to use an additional design matrix to

fragment the elementary symmetric function. Their approach is implemented in this package. MST

designs with a probabilistic instead of a deterministic routing rule (see, e.g. Chen, Yamamoto, & von

Davier, 2014 <doi:10.1201/b16858>) are not estimated with this method, therefore the proposed

solouting is again modified by Steinfeld and Robitzsch (2021) <doi:10.31234/osf.io/ew27f> which

is also integrated into this package.

Value

List with following entries

betapar  Estimated item difficulty parameters (if sum0=FALSE, than the first item is set
to 0)

se.beta  Standard errors of the estimated item parameters

loglik  Conditional log-likelihood of the model

df  Number of estimated parameters

N  Number of Persons

I  Number of items

data_orig  Submitted data frame with item responses

data  Used data frame with item responses

desmat  Design matrix

convergence  Convergence criterion

iterations  Number of iterations

hessian  Hessian-Matrix

model  Used model ((mst) for Rasch model with multistage design)

call  Submitted arguments for the function (matched call)

designelements  If the multistage version is requested, the preprocessed design is returned, oth-

otherwise NULL

mstdesign  If the multistage version is requested, the submitted design is returned, otherwise

NULL
Author(s)

Jan Steinfeld

References


See Also

tmt_lrtest
Examples

# example for tmt_rm
#############################################################################
# Example-1 simple Rasch model
#############################################################################
dat <- tmt:::sim.rm(theta = 100, b = 10, seed = 1111)
dat.rm <- tmt_rm(dat = dat)
summary(dat.rm)

#############################################################################
# Example-1 for multistage-design
#############################################################################
mstdesign <- "
  M1 =~ c(i1, i2, i3, i4, i5)
  M2 =~ c(i6, i7, i8, i9, i10)
  M3 =~ c(i11, i12, i13, i14, i15)
"

items <- seq(-1,1,length.out = 15)
names(items) <- paste0("i",1:15)
persons = 1000

dat <- tmt_sim(mstdesign = mstdesign,
  items = items, persons = persons)
dat.rm <- tmt_rm(dat = dat, mstdesign = mstdesign)
summary(dat.rm)

## Not run:
############################################################################

# Example-2 simple Rasch model
#############################################################################
dat <- tmt:::sim.rm(theta = 100, b = 10, seed = 1111)
dat.rm <- tmt_rm(dat = dat)
summary(dat.rm)

#############################################################################
# Example-2 for multistage-design
#############################################################################
# also using 'paste' is possible
mstdesign <- "
  M1 =- paste0("i",1:5)
  M2 =- paste0("i",6:10)
  M3 =- paste0("i",11:15)
  M4 =- paste0("i",16:20)
  M5 =- paste0("i",21:25)
  M6 =- paste0("i",26:30)
"

# define path
```
p1 := M4(0,2) + M2(0,2) + M1
p2 := M4(0,2) + M2(3,5) + M3
p3 := M4(3,5) + M5(0,2) + M3
p4 := M4(3,5) + M5(3,5) + M6
```

```
items <- seq(-1,1,length.out = 30)
names(items) <- paste0("i",1:30)
persons = 1000
dat <- tmt_sim(mstdesign = mstdesign, 
   items = items, persons = persons)
dat.rm <- tmt_rm(dat = dat, mstdesign = mstdesign)
summary(dat.rm)
```

```
# Example-3 for cumulative multistage-design
# also using "paste" is possible
mstdesign <- 
   "
   M1 =~ paste0("i",21:30)
   M2 =~ paste0("i",11:20)
   M3 =~ paste0(1:10)
   M4 =~ paste0(31:40)
   M5 =~ paste0(41:50)
   M6 =~ paste0(51:60)
"

# define path
p1 := M1(0, 5) += M2( 0,10) += M3
p2 := M1(0, 5) += M2(11,15) += M4
p3 := M1(6,10) += M5( 6,15) += M4
p4 := M1(6,10) += M5(16,20) += M6
```

```
items <- seq(-1,1,length.out = 60)
names(items) <- paste0("i",1:60)
persons = 1000
dat <- tmt_sim(mstdesign = mstdesign, 
   items = items, persons = persons)
dat.rm <- tmt_rm(dat = dat, mstdesign = mstdesign)
summary(dat.rm)
```

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

---

### tmt_sim

**Function for the Simulation of Multistage-Designs**

**Description**

This function simulates data according to the specified and submitted multistage design. The persons are drawn from a standard normal distribution if the amount of persons are specified. As an additional argument, a seed can also be set. If requested, it is also possible to submit a vector or list of person parameters to specify different person distributions.
Usage

tmt_sim(
  mstdesign = NULL,
  items = NULL,
  persons = NULL,
  preconditions = NULL,
  ...
)

Arguments

mstdesign definition of desired multistage design
items vector of difficulty parameters for each items
persons amount of persons per starting module
preconditions definition of preconditions can optionally be specified. In the case of probabilistic routing preconditions such as a pre-test, which are taken into account in the MST design. For the specification the correlation with the true person parameter have to be specified. The submitted correlation is adjusted in the function according to Demirtas and Yavuz (2015; <doi:10.1080/10543406.2014.920868>) It is also possible to submit your own vector with integers for the preconditions.
... further optional arguments like setting a seed

Value

List with following entries

data Matrix with item responses
data_mst Data frame with item responses and additional a vector of used modules per person
persons Generated and used person parameters
mstdesign Submitted multistage design

Author(s)

Jan Steinfeld

References


Examples

```
#########################################################################
# translate multistage model 1
#########################################################################
#########################################################################
mstdesign <- "
```
M1 =~ c(i1, i2, i3, i4, i5)
M2 =~ c(i6, i7, i8, i9, i10)
M3 =~ c(i11, i12, i13, i14, i15)

# define branches
p1 := M2(0,2) + M1
p2 := M2(3,5) + M3

items <- seq(-3,3,length.out = 15)
names(items) <- paste0("i", seq(items))
data_1 <- tmt_sim(mstdesign = mstdesign,
                items = items,
                persons = 500,
                seed = 1111)

# translate multistage model 2
mstdesign <- "
M1 =~ c(i1, i2, i3, i4, i5)
M2 =~ c(i6, i7, i8, i9, i10)
M3 =~ c(i11, i12, i13, i14, i15)
M4 =~ c(i16, i17, i18, i19, i20)
M5 =~ c(i21, i22, i23, i24, i25)
M6 =~ c(i26, i27, i28, i29, i30)

# define branches
p1 := M4(0,2) + M2(0,2) + M1
p2 := M4(0,2) + M2(3,5) + M3
p3 := M4(3,5) + M5(0,2) + M3
p4 := M4(3,5) + M5(3,5) + M6

items <- seq(-3,3,length.out = 30)
names(items) <- paste0("i", seq(items))
data_2 <- tmt_sim(mstdesign = mstdesign,
                items = items,
                persons = 500,
                seed = 1111)
Index

tmt (tmt-package), 2
 tmt-package, 2
 tmt_gmc, 4
 tmt_lrtest, 6, 12
 tmt_mstdesign, 7
 tmt_msttemplate, 9
 tmt_rm, 7, 10
 tmt_sim, 14