Package: spfa (via r-universe)

October 4, 2026

Type Package

Title Semi-Parametric Factor Analysis

Version 1.0

Date 2023-04-25

Maintainer Yang Liu <yliu87@umd.edu>

Description Estimation, scoring, and plotting functions for the semi-parametric factor model proposed by Liu & Wang (2022) <doi:10.1007/s11336-021-09832-8> and Liu & Wang (2023) <doi:10.48550/arXiv.2303.10079>. Both the conditional densities of observed responses given the latent factors and the joint density of latent factors are estimated non-parametrically. Functional parameters are approximated by smoothing splines, whose coefficients are estimated by penalized maximum likelihood using an expectation-maximization (EM) algorithm. E- and M-steps can be parallelized on multi-thread computing platforms that support 'OpenMP'. Both continuous and unordered categorical response variables are supported.

License MIT + file LICENSE

Imports graphics, Rcpp

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.2.3

LazyData true

Encoding UTF-8

Depends R (>= 2.10)

NeedsCompilation yes

Author Yang Liu [cre, aut], Weimeng Wang [aut, ctb]

Repository https://cran.r-universe.dev

Date/Publication 2023-05-26 07:20:02 UTC

RemoteUrl https://github.com/cran/spfa

RemoteRef HEAD

RemoteSha 0f18b7a1db8cb601ad3146cae34e87358311c667

Contents

spfa-package 2
fscores 2
plotitem.cont 3
simdata 5
spfa 5
spfa_example 7

Index **8**

`spfa-package` *Semiparametric Factor Analysis*

Description

The package provides estimation, scoring, and plotting functions for the semiparametric parametric factor model proposed by Liu & Wang (2022; 2023). Both the conditional densities of observed responses given the latent factors and the joint density of latent factors are estimated nonparametrically. Functional parameters are approximated by smoothing splines, whose coefficients are estimated by penalized maximum likelihood using an expectation-maximization (EM) algorithm. E- and M-steps can be parallelized on multi-thread computing platforms that support OpenMP. Both continuous and unordered categorical response variables are supported.

`fscores` *Computing EAP scores*

Description

Computing EAP scores

Usage

```
fscores(
  data,
  fit,
  dimension = rep(0, ncol(data)),
  discrete = rep(FALSE, ncol(data)),
  normal = TRUE,
  control = list()
)
```

Arguments

data	data to be scored
fit	the function return from fitting a spfa model.
dimension	a vector of integers containing indicators of the latent factor. The default is rep(0, ncol(data)) indicating all item loads on the same latent factor.
discrete	a vector of TRUE or FALSE indicating whether the item is discrete with TRUE indicating the item is a discrete variable. The length of the vector should be the same as the number of columns of the input data.
normal	a logical value TRUE or FALSE indicating which density is used to rescale y.
control	a list of technical control variables. See spfa.

Value

EAP scores for the fitted spfa model and reliability

Examples

```
RT <- spfa::simdata[,1:8]
myeaps <- fscores(data = RT, fit = spfa::spfa_example,
dimension = rep( 0, ncol(RT)), discrete = rep(FALSE, ncol(RT) ))
```

---

plotitem.cont *Item level perspective plots or contour plots for spfa models*

Description

For continuous response data use plotitem.cont whereas discrete response data use plotitem.disc. For joint continuous and discrete data, use plotgroup.

Usage

```
plotitem.cont(
  param,
  nquad = 21,
  npoints = 101,
  xlim = c(-2.5, 2.5),
  ylim = c(0, 1),
  normal = TRUE,
  FUN = NULL,
  plot = TRUE,
  type = "contour",
  ...
)

plotitem.disc(
```

```
  param,
  ncat,
  npoints = 101,
  xlim = c(-2.5, 2.5),
  normal = TRUE,
  FUN = NULL,
  plot = TRUE,
  col = 1:ncat,
  lty = rep(1, ncat),
  ...
)

plotgroup(
  param,
  nquad = 21,
  npoints = 101,
  lim = c(-2.5, 2.5),
  normal = TRUE,
  plot = TRUE,
  type = "contour",
  ...
)
```

Arguments

param	parameter vector estimated from spfa model
nquad	an integer value of number of quadrature points. Default is 21
npoints	an integer value of number of x and y levels in the plot
xlim	the x limits of the plot. Two numerical values indicating the lower and upper limits
ylim	the y limits of the plot. Two numerical values indicating the lower and upper limits of the density. Note y is rescaled to a uniform [0,1] distribution.
normal	a logical value TRUE or FALSE indicating which density is used to rescale y.
FUN	a user supplied function to rescale.
plot	a logical value TRUE or FALSE indicating whether the plot is visualized.
type	the type of plot to be visualized. The default is the contour plot contour. It can also be changed to "persp" indicating perspective plots.
...	additional arguments passed to contour and persp
ncat	an integer value indicating the number of categories for the discrete item.
col	color of the line.
lty	line type
lim	limit

Value

plots. Item level perspective and contour plot

See Also

contour and persp

Examples

```
# Contour plot of the first item

plotitem.cont(spfa::spfa_example$par[[1]])
```

---

simdata *simdata*

Description

There are 16 columns with 1000 rows. The first 8 columns contain continuous item response time data and the last 8 columns with discrete item responses. Among the discrete items, there are 4 dichotomous and 4 four-category ones.

---

spfa *Fitting Semi-parametric Factor Analysis Model*

Description

spfa fits a unidimensional or two-dimension factor analysis spfa model using penalized maximum likelihood estimation. A unidimensional spfa model can handle discrete response data (i.e., item responses including binary responses and polytomous responses) or continuous response data (e.g., response time). A two-dimensional spfa model can only handle simple structure model with two latent factors load to continuous and discrete response data, respectively.

Usage

```
spfa(
  data,
  dimension = rep(0, ncol(data)),
  discrete = rep(FALSE, ncol(data)),
  control = list()
)
```

Arguments

data	a matrix that consists of item responses with missing data coded as NA.
dimension	a vector of integers containing indicators of the latent factor. The default is rep(0, ncol(data)) indicating all item load on the same latent factor.

`discrete` a vector of `TRUE` or `FALSE` indicating whether the item is discrete. `TRUE`: discrete variable. The length of the vector should be the same as the number of columns of the input data.

`control` a list containing technical parameters for estimation. May be:

- `shortpar` a list containing the starting values of spfa model parameters for each item.
- `pos` a list containing positivity constraints
- `lmbd` a vector of the penalty parameter (lambda). Default value is a vector of 1s.
- `n_basis` number of basis. Default is 11.
- `n_quad` number of quadrature points. Default is 21
- `maxit_em` the maximum number of iterations for the EM cycles. Default is 500.
- `maxit_mstep` the maximum number of iterations for the mstep optimizer.
- `maxit_start`
- `tol_em` tolerance for the EM convergence. Default is 1e-4.
- `tol_mstep` tolerance for the m-step optimizer. Default is 1e-6.
- `n_thrd` number of cores used for the penalized EM algorithm to run. Default is 1.

Value

a list including spfa model parameter estimates and marginal log-likelihood.

References

Liu, Y., & Wang, W. (2022). Semiparametric Factor Analysis for Item-Level Response Time Data. *Psychometrika, 87*(2), 666–692. doi:10.1007/s11336021098328

Liu, Y., & Wang, W. (2023). What Can We Learn from a Semiparametric Factor Analysis of Item Responses and Response Time? An Illustration with the PISA 2015 Data. Retrieved from https://arxiv.org/abs/2303.10079

Examples

```
# load item response time data
RT <- spfa::simdata[,1:8]

# Fit a unidimensional spfa model with continuous data (Response time)



spfa_example <- spfa(data = RT,
       dimension = rep(0, ncol(RT)),
       discrete = rep(FALSE, ncol(RT)))


# In the spfa pacakge, the output of spfa_example can be directly extracted.
# See example code below:
```

```
spfa::spfa_example$shortpar

# Visualize the result for item 1 as an example

plotitem.cont(spfa::spfa_example$par[[1]])
```

---

spfa_example *spfa_example*

Description

An R object containing an example output from fitting an spfa model using the following code: # RT <- spfa::simdata[,1:8] # spfa_example <- spfa(data = RT, dimension = rep(0, ncol(RT)), discrete = rep(FALSE, ncol(RT)))

Index

∗ **data**
simdata, 5
spfa_example, 7
∗ **package**
spfa-package, 2

contour, *4, 5*

fscores, 2

persp, *4, 5*
plotgroup *(*plotitem.cont*)*, 3
plotitem.cont, 3
plotitem.disc *(*plotitem.cont*)*, 3

simdata, 5
spfa, *3, 4*, 5
spfa-package, 2
spfa_example, 7