Package ‘emIRT’

February 4, 2020

Type Package
Title EM Algorithms for Estimating Item Response Theory Models
Version 0.0.11
Date 2020-02-01
Author Kosuke Imai <kimai@princeton.edu>, James Lo <lojames@usc.edu>, Jonathan Olmsted <jpolmsted@gmail.com>
Maintainer James Lo <lojames@usc.edu>
Description Various Expectation-Maximization (EM) algorithms are implemented for item response theory (IRT) models. The current implementation includes IRT models for binary and ordinal responses, along with dynamic and hierarchical IRT models with binary responses. The latter two models are derived and implemented using variational EM. Subsequent edits also include variational network and text scaling models.
License GPL (>= 3)
Depends R (>= 2.10), pscl (>= 1.0.0), Rcpp (>= 0.10.6)
LinkingTo Rcpp, RcppArmadillo
NeedsCompilation yes
Repository CRAN
Date/Publication 2020-02-04 06:20:02 UTC

R topics documented:

AsahiTodai .................................................. 2
binIRT .................................................. 3
boot_emIRT .................................................. 7
correct .................................................. 8
dwnom .................................................. 10
dynIRT .................................................. 11
getStarts .................................................. 15
hierIRT .................................................. 16
makePriors .................................................. 20
manifesto .................................................. 21
Asahi-Todai Elite Survey

Description

The Asahi-Todai Elite survey was conducted by the University of Tokyo in collaboration with a major national newspaper, the Asahi Shimbun, covering all candidates (both incumbents and challengers) for the eight Japanese Upper and Lower House elections that occurred between 2003 and 2013. In six out of eight waves, the survey was also administered to a nationally representative sample of voters with the sample size ranging from approximately 1,100 to about 2,000. The novel feature of the data is that there are a set of common policy questions, which can be used to scale both politicians and voters over time on the same dimension.

All together, the data set contains a total of N = 19,443 respondents, including 7,734 politicians and 11,709 voters. There are J = 98 unique questions in the survey, most of which consisted of questions asking for responses on a 5-point Likert scale. However, these scales were collapsed into a 3-point Likert scale for estimation with ordIRT(). In the data set, we include estimates obtained via MCMC using both the 3 and 5-category data. See Hirano et al. 2011 for more details.

Usage

data(AsahiTodai)

Value

AsahiTodai list, containing the following elements:

- dat.all Survey data, formatted for input to ordIRT().
- start.values Start values, formatted for input to ordIRT().
- priors Priors, formatted for input to ordIRT().
- ideal3 Ideal point estimates with data via MCMC, using collapsed 3-category data.
- ideal5 Ideal point estimates with data via MCMC, using original 5-category data.
- obs.attri Attribute data of the respondents.
binIRT

Two-parameter Binary IRT estimation via EM

Description

binaryIRT estimates a binary IRT model with two response categories. Estimation is conducted using the EM algorithm described in the reference paper below. The algorithm will produce point estimates that are comparable to those of ideal, but will do so much more rapidly and also scale better with larger data sets.

Usage

```r
binIRT(.rc, .starts = NULL, .priors = NULL, .D = 1L, .control = NULL, 
   .anchor_subject = NULL, .anchor_outcomes = FALSE)
```

References


See Also

'ordIRT'.

Examples

```r
## Not run:
### Real data example: Asahi-Todai survey (not run)
## Collapses 5-category ordinal survey items into 3 categories for estimation
data(AsahiTodai)
out.varinf <- binIRT(.rc = AsahiTodai$dat.all, .starts = AsahiTodai$start.values, 
   .priors = AsahiTodai$priors, .D = 1, 
   .control = {list(verbose = TRUE, 
               thresh = 1e-6, maxit = 500)})

## Compare against MCMC estimates using 3 and 5 categories
cor(ideal3, out.varinf$means$x)
cor(ideal5, out.varinf$means$x)

## End(Not run)
```
Arguments

.rc a list object, in which .rc$votes is a matrix of numeric values containing the data to be scaled. Respondents are assumed to be on rows, and items assumed to be on columns, so the matrix is assumed to be of dimension (N x J). For each item, ‘1’, and ‘-1’ represent different responses (i.e. yes or no votes) with ‘0’ as a missing data record.

.starts a list containing several matrices of starting values for the parameters. The list should contain the following matrices:

• alpha A (J x 1) matrix of starting values for the item difficulty parameter \( \alpha \).
• beta A (J x D) matrix of starting values for the item discrimination parameter \( \beta \).
• x An (N x D) matrix of starting values for the respondent ideal points \( x_i \).

.priors list, containing several matrices of starting values for the parameters. The list should contain the following matrices:

• x$mu A (D x D) prior means matrix for respondent ideal points \( x_i \).
• x$sigma A (D x D) prior covariance matrix for respondent ideal points \( x_i \).
• beta$mu A (D+1 x 1) prior means matrix for \( \alpha_j \) and \( \beta_j \).
• beta$sigma A (D+1 x D+1) prior covariance matrix for \( \alpha_j \) and \( \beta_j \).

.D integer, indicates number of dimensions to estimate. Only a 1 dimension is currently supported. If a higher dimensional model is requested, binIRT exits with an error.

.control list, specifying some control functions for estimation. Options include the following:

• threads integer, indicating number of cores to use. Default is to use a single core, but more can be supported if more speed is desired.
• verbose boolean, indicating whether output during estimation should be verbose or not. Set FALSE by default.
• thresh numeric. Algorithm will run until all parameters have a correlation greater than \( (1 - \text{threshold}) \) across consecutive iterations. Set at 1e-6 by default.
• maxit integer. Sets the maximum number of iterations the algorithm can run. Set at 500 by default.
• checkfreq integer. Sets frequency of verbose output by number of iterations. Set at 50 by default.

.anchor_subject integer, the index of the subject to be used in anchoring the orientation/polarity of the underlying latent dimensions. Defaults to NULL and no anchoring is done.

.anchor_outcomes logical, should an outcomes-based metric be used to anchor the orientation of the underlying space. The outcomes-based anchoring uses a model-free/non-parametric approximation to quantify each item’s difficulty and each subject’s ability. The post-processing then rotates the model-dependent results to match the model-free polarity. Defaults to FALSE and no anchoring is done.
Value

An object of class `binIRT`.

**means**  
list, containing several matrices of point estimates for the parameters corresponding to the inputs for the priors. The list should contain the following matrices:

- `x` A (N x 1) matrix of point estimates for the respondent ideal points $x_i$.
- `beta` A (J x D+1) matrix of point estimates for the item parameters $\alpha$ and $\beta$.

**vars**  
list, containing several matrices of variance estimates for parameters corresponding to the inputs for the priors. Note that these variances are those recovered via variational approximation, and in most cases they are known to be far too small and generally unusable. Better estimates of variances can be obtained manually via the parametric bootstrap. The list should contain the following matrices:

- `x` A (N x 1) matrix of variances for the respondent ideal points $x_i$.
- `beta` A (J x D+1) matrix of variances for the item parameters $\alpha$ and $\beta$.

**runtime**  
A list of fit results, with elements listed as follows:

- `iters` integer, number of iterations run.
- `conv` integer, convergence flag. Will return 1 if threshold reached, and 0 if maximum number of iterations reached.
- `threads` integer, number of threads used to estimate the model.
- `tolerance` numeric, tolerance threshold for convergence. Identical to thresh argument in input to .control list.

**n**  
Number of respondents in estimation, should correspond to number of rows in roll call matrix.

**j**  
Number of items in estimation, should correspond to number of columns in roll call matrix.

**d**  
Number of dimensions in estimation.

**call**  
Function call used to generate output.

**Author(s)**

Kosuke Imai <kimai@princeton.edu>
James Lo <lojames@usc.edu>
Jonathan Olmsted <jpolmsted@gmail.com>

**References**


**See Also**

`convertRC`, `makePriors`, `getStarts`.
Examples

## Data from 109th US Senate
data(s109)

## Convert data and make starts/priors for estimation
rc <- convertRC(s109)
p <- makePriors(rc$n, rc$m, 1)
s <- getStarts(rc$n, rc$m, 1)

## Conduct estimates
lout <- binIRT(.rc = rc,
               .starts = s,
               .priors = p,
               .control = {
                  list(threads = 1,
                       verbose = FALSE,
                       thresh = 1e-6
                  )
               })

## Look at first 10 ideal point estimates
lout$means$x[1:10]

lout2 <- binIRT(.rc = rc,
               .starts = s,
               .priors = p,
               .control = {
                  list(threads = 1,
                       verbose = FALSE,
                       thresh = 1e-6
                  ),
                  .anchor_subject = 2
               })

# Rotates so that Sen. Sessions (R AL)
# has more of the estimated trait

lout3 <- binIRT(.rc = rc,
               .starts = s,
               .priors = p,
               .control = {
                  list(threads = 1,
                       verbose = FALSE,
                       thresh = 1e-6
                  ),
                  .anchor_subject = 10
               })

# Rotates so that Sen. Boxer (D CA)
# has more of the estimated trait

\[
\text{cor(lout2$means$x[, 1], }
\text{lout3$means$x[, 1]})
\]

# = -1 --> same numbers, flipped
# orientation

---

**boot_emIRT**

*Parametric bootstrap of EM Standard Errors*

**Description**

`boot_emIRT` take an emIRT() object (from binary, ordinal, dynamic, or hierarchical models) and implements a parametric bootstrap of the standard errors for the ideal points. It assumes you have already run the model successfully on one of those functions, and takes the output from that estimate, along with the original arguments, as the arguments for the bootstrap function.

**Usage**

```r
boot_emIRT(emIRT.out, .data, .starts, .priors, .control, Ntrials=50, verbose=10)
```

**Arguments**

- **emIRT.out**: an emIRT() object, which is output from a call to binIRT(), dynIRT(), ordIRT(), or hierIRT()
- **.data**: the data used to produce the emIRT object.
- **.starts**: the starts used to produce the emIRT object.
- **.priors**: the priors used to produce the emIRT object.
- **.control**: the control arguments used to produce the emIRT object.
- **Ntrials**: Number of bootstrap trials to run.
- **verbose**: Number of trials before progress notification triggers.

**Value**

An object of class emIRT. The output takes the original emIRT.out object and appends the following:

- **bse**: list, containing only the matrix:
  - x A (N x 1) matrix of bootstrapped standard errors

**Author(s)**

Kosuke Imai <kimai@princeton.edu>
James Lo <lojames@usc.edu>
Jonathan Olmsted <jpolmsted@gmail.com>
References


See Also

`'binIRT'`, `'ordIRT'`, `'hierIRT'`, `'dynIRT'`.

Examples

```r
## Not run:

### Binary IRT example
example(binIRT)
boot.bin <- boot_emIRT(lout, .data = rc, .starts = s, .priors = p,
   .control = list(threads = 1, verbose = FALSE, thresh = 1e-06), Ntrials=10, verbose=2)
boot.bin$bse$x

### Dynamic IRT example
example(dynIRT)
boot.dyn <- boot_emIRT(lout, .data = mq_data$data.mq, .starts = mq_data$cur.mq,
   .priors = mq_data$priors.mq, .control = list(threads = 1, verbose = FALSE,
   thresh = 1e-06), Ntrials=10, verbose=2)
boot.dyn$bse$x

### Ordinal IRT example
example(ordIRT)
boot.ord <- boot_emIRT(lout, .data=newrc, .starts=cur, .priors=priors,
   .control = list(threads = 1, verbose = TRUE, thresh = 1e-6, maxit=300,
   checkfreq=50), Ntrials=5, verbose=1)
boot.ord$bse$x

### Hierarchical IRT example
example(hierIRT, run.dontrun=TRUE)
boot.hier <- boot_emIRT(lout, .data=dwnom$data.in, .starts=dwnom$cur, .priors=dwnom$priors,
   .control=list(threads = 8, verbose = TRUE, thresh = 1e-4, maxit=200, checkfreq=1),
   Ntrials=5, verbose=1)
boot.hier$bse$x_implied

## End(Not run)
```

---

**convertRC**  
*Convert Roll Call Matrix Format*

**Description**

convertRC takes a `rollcall` object and converts it into a format suitable for estimation with `'binIRT'`. 
Usage

convertRC(.rc, type = "binIRT")

Arguments

.rc

a rollcall object containing votes to be scaled using 'binIRT'.

type

string, only "binIRT" is supported for now, and argument is ignored.

Value

An object of class rollcall, with votes recoded such that yea=1, nay=-1, missing data = 0.

Author(s)

Kosuke Imai <kimai@princeton.edu>
James Lo <lojames@usc.edu>
Jonathan Olmsted <jpolmsted@gmail.com>

References


See Also

'binIRT', 'makePriors', 'getStarts'.

Examples

## Data from 109th US Senate
data(s109)

## Convert data and make starts/priors for estimation
rc <- convertRC(s109)
p <- makePriors(rc$n, rc$m, 1)
s <- getStarts(rc$n, rc$m, 1)

## Conduct estimates
lout <- binIRT(.rc = rc, .starts = s, .priors = p, .control = {
  list(threads = 1, verbose = FALSE, thresh = 1e-6
  )
})

}
## Look at first 10 ideal point estimates
lout$means$x[1:10]

---

**dwnom**

*Poole-Rosenthal DW-NOMINATE data and scores, 80-110 U.S. Senate*

### Description

This data set contains materials related to the Poole-Rosenthal DW-NOMINATE measure of senator ideology. The software (and other materials) is available at Voteview, which includes a simpler example of an application to the 80-110 U.S. Senate. The data set here is derived from the data and estimates from that example, but are formatted to be run in `hierIRT()`. In particular, start values for estimation are identical to those provided by the example.

### Usage

```r
data(dwnom)
```

### Value

`dwnom` list, containing the following elements:

- `data.in` Legislator voting data, formatted for input to `hierIRT()`.
- `cur` Start values, formatted for input to `hierIRT()`.
- `priors` Priors, formatted for input to `hierIRT()`.
- `legis` data frame, containing contextual information about the legislators estimated.
- `nomres` data frame, containing estimates from DW-NOMINATE on the same data. These are read from the file SL80110C21.DAT.

### References


### See Also

`'hierIRT'`.
### Real data example of US Senate 80-110 (not run)
### Based on voteview.com example of DW-NOMINATE
### We estimate a hierarchical model without noise and a linear time covariate
### This model corresponds very closely to the DW-NOMINATE model

```
## Not run:
data(dwnom)

## This takes about 10 minutes to run on 8 threads
## You may need to reduce threads depending on what your machine can support
lout <- hierIRT(.data = dwnom$data.in,
    .starts = dwnom$cur,
    .priors = dwnom$priors,
    .control = {list(
        threads = 8,
        verbose = TRUE,
        thresh = 1e-4,
        maxit=200,
        checkfreq=1
    )})

## Bind ideal point estimates back to legislator data
final <- cbind(dwnom$legis, idealpt.hier=lout$means$x_implied)

## These are estimates from DW-NOMINATE as given on the Voteview example
## From file "SL80110C21.DAT"
nomres <- dwnom$nomres

## Merge the DW-NOMINATE estimates to model results by legislator ID
## Check correlation between hierIRT() and DW-NOMINATE scores
res <- merge(final, nomres, by=c("senate","id"),all.x=TRUE,all.y=FALSE)
cor(res$idealpt.hier, res$dwnom1d)

## End(Not run)
```

---

dynIRT

`dynIRT` is an R package for dynamic IRT estimation via Variational Inference.

**Description**

`ordIRT` estimates an dynamic IRT model with two response categories per item, over several sessions. Ideal points over time follow a random walk prior, and the model originates from the work of Martin and Quinn (2002). Estimation is conducted using the variational EM algorithm described in the reference paper below. The algorithm will produce point estimates that are comparable to those of `MCMCdynamicIRT1d`, but will do so much more rapidly and also scale better with larger data sets.
Usage

dynIRT(.data, .starts = NULL, .priors = NULL, .control = NULL)

Arguments

.data a list with the following items.

• rc A (N x J) matrix of observed votes. ‘1’ and ‘-1’ are the yea and nay codes, while ‘0’ is a missing data code.

• startlegis An (N x 1) matrix indicating the first session that each legislator serves. Justices are assumed to serve in all terms between (and including) ‘startlegis’ to ‘endlegis’. Terms start at term 0, and end at term T - 1.

• endlegis An (N x 1) matrix indicating the last session that each legislator serves. Justices are assumed to serve in all terms between (and including) ‘startlegis’ to ‘endlegis’. Terms start at term 0, and end at term T - 1.

• bill/session A (J x 1) matrix of integers indicating the session each bill occurred in. Session count begins at 0, so the maximum value of bill/session is T - 1.

• T integer, indicating total number of consecutive terms in the data. Count starts from 1, so the maximum values of startlegis/endlegis/bill/session is T - 1, since they start from term 0.

.starts a list containing several matrices of starting values for the parameters. The list should contain the following matrices:

• alpha A (J x 1) matrix of starting values for the item difficulty parameter $\alpha_j$.

• beta A (J x 1) matrix of starting values for the item discrimination parameter $\beta_j$.

• x An (N x T) matrix of starting values for the respondent ideal points $x_{it}$, with rows indicating the legislator and columns indicating the session. Although not strictly necessary, it is generally good practice here to set the start values for legislators who are not serving in a particular session to 0, as that is what the point estimate for them will return.

.priors list, containing several matrices of starting values for the parameters. The list should contain the following matrices:

• x.mu0 A (N x 1) prior means matrix for respondent ideal points $c_{i0}$. These are generally set to be somewhat informative to resolve the standard rotational invariance problem in ideal point models.

• x.sigma0 A (N x 1) prior variance matrix for respondent ideal points $C_{i0}$.

• beta.mu A (2 x 1) prior means matrix for all bill parameters $\alpha_j$ and $\beta_j$.

• beta.sigma A (2 x 2) prior covariance matrix for all bill parameters $\alpha_j$ and $\beta_j$.

• omega2 A (N x 1) matrix with the evolutionary variance for each legislator $\omega_{ix}^2$.

.control list, specifying some control functions for estimation. Options include the following:
• threads integer, indicating number of cores to use. Default is to use a single core, but more can be supported if more speed is desired.
• verbose boolean, indicating whether output during estimation should be verbose or not. Set FALSE by default.
• thresh numeric. Algorithm will run until all parameters correlate at 1 - thresh across consecutive iterations. Set at 1e-6 by default.
• maxit integer. Sets the maximum number of iterations the algorithm can run. Set at 500 by default.
• checkfreq integer. Sets frequency of verbose output by number of iterations. Set at 50 by default.

Value

An object of class dynIRT.

means list, containing several matrices of point estimates for the parameters corresponding to the inputs for the priors. The list should contain the following matrices.

• x A (N x T) matrix of point estimates for the respondent ideal points \( x_{it} \), with rows indicating respondent and columns indicating session of the estimated ideal point. Estimates will equal exactly 0 for legislator/period combinations in which the legislator did not serve.
• alpha A (J x 1) matrix of point estimates for the item difficulty parameter \( \alpha_j \).
• beta A (J x 1) matrix of point estimates for the item discrimination parameter \( \beta_j \).

vars list, containing several matrices of variance estimates for parameters corresponding to the inputs for the priors. Note that these variances are those recovered via variational approximation, and in most cases they are known to be far too small and generally unusable. Better estimates of variances can be obtained manually via the parametric bootstrap. The list should contain the following matrices:

• x A (N x T) matrix of variance estimates for the respondent ideal points \( x_{it} \), with rows indicating respondent and columns indicating session of the estimated ideal point. Estimates will equal exactly 0 for legislator/period combinations in which the legislator did not serve.
• alpha A (J x 1) matrix of variance estimates for the item difficulty parameter \( \alpha_j \).
• beta A (J x 1) matrix of variance estimates for the item discrimination parameter \( \beta_j \).

runtime A list of fit results, with elements listed as follows:

• iters integer, number of iterations run.
• conv integer, convergence flag. Will return 1 if threshold reached, and 0 if maximum number of iterations reached.
• threads integer, number of threads used to estimated model.
• tolerance numeric, tolerance threshold for convergence. Identical to thresh argument in input to .control list.

N
Number of respondents in estimation, should correspond to number of rows in roll call matrix.

J
Number of items in estimation, should correspond to number of columns in roll call matrix.

T
Number of time periods fed into the estimation, identical to argument input from .data list.

call
Function call used to generate output.

Author(s)
Kosuke Imai <kimai@princeton.edu>
James Lo <lojames@usc.edu>
Jonathan Olmsted <jpolmsted@gmail.com>

References
Original model and the example is based off of Andrew Martin and Kevin Quinn, “Dynamic Ideal Point Estimation via Markov Chain Monte Carlo for the U.S. Supreme Court, 1953-1999.” Political Analysis 10(2) 134-153.

Variational model is described in Kosuke Imai, James Lo, and Jonathan Olmsted “Fast Estimation of Ideal Points with Massive Data.”

See Also
'mq_data'.

Examples

### Replication of Martin-Quinn Judicial Ideology Scores
### Based on July 23, 2014 (2014 Release 01) release of the Supreme Court Database
### Start values and priors based on replication code provided by Kevin Quinn

```r
data(mq_data)

# Estimate dynamic variational model using dynIRT()
lout <- dynIRT(.data = mq_data$data.mq,
 .starts = mq_data$cur.mq,
 .priors = mq_data$priors.mq,
 .control = {list(
   threads = 1,
   verbose = TRUE,
   thresh = 1e-6,
   maxit=500
 ))
```
getStarts

## Extract estimate from variational model
## Delete point estimates of 0, which are justices missing from that session
vi.out <- c(t(lout$means$x))
vi.out[vi.out==0] <- NA
vi.out <- na.omit(vi.out)

## Compare correlation against MCMC-estimated result
## Correlates at r=0.93 overall, and 0.96 when excluding Douglas
cor(vi.out, mq_data$mq_mcmc)
cor(vi.out[mq_data$justiceName != "Douglas"],
   mq_data$mq_mcmc[mq_data$justiceName != "Douglas"])

---

getStarts

Generate Starts for binIRT

Description

getStarts generates starting values for binIRT.

Usage

getStarts(.N, .J, .D, .type = "zeros")

Arguments

- `.N` integer, number of subjects/legislators to generate starts for.
- `.J` integer, number of items/bills to generate starts for.
- `.D` integer, number of dimensions.
- `.type` “zeros” and “random” are the only valid types, will generate starts accordingly.

Value

- `alpha` A (J x 1) matrix of starting values for the item difficulty parameter \( \alpha \).
- `beta` A (J x D) matrix of starting values for the item discrimination parameter \( \beta \).
- `x` An (N x D) matrix of starting values for the respondent ideal points \( x_i \).

Author(s)

Kosuke Imai <kimai@princeton.edu>
James Lo <lojames@usc.edu>
Jonathan Olmsted <jpolmsted@gmail.com>

References

hierIRT

Hierarchical IRT estimation via Variational Inference

Description

hierIRT estimates an hierarchical IRT model with two response categories, allowing the use of covariates to help determine ideal point estimates. Estimation is conducted using the variational EM algorithm described in the reference paper below. A special case of this model occurs when time/session is used as the covariate — this allows legislator ideal points to vary over time with a parametric time trend. Notably, the popular DW-NOMINATE model (Poole and Rosenthal, 1997) is one such example, in which legislator ideal points shift by a constant amount each period, and the error term in the hierarchical model is set to 0. In contrast to other functions in this package, this model does not assume a ‘rectangular’ roll call matrix, and all data are stored in vector form.

Usage

hierIRT(.data, .starts = NULL, .priors = NULL, .control = NULL)
Arguments

.data a list with the following items:
  • y A (L x 1) matrix of observed votes. ‘1’ and ‘-1’ are the yea and nay codes.
  • i A (L x 1) integer matrix of indexes of the ideal point i[l] linked to each observed vote l = 0 . . . L. Indexes begin at 0 and reach a maximum value of I - 1.
  • j A (L x 1) integer matrix of indexes of the bill/item j[l] linked to each observed vote l = 0 . . . L. Indexes begin at 0 and reach a maximum value of J - 1.
  • g A (I x 1) integer matrix of indexes of the group membership g[i[l]] linked to each ideal point i = 0 . . . I. Indexes begin at 0 and reach a maximum value of G - 1.
  • z A (I x D) numeric matrix of observed covariates. Rows correspond to ideal points i = 0 . . . I. The columns correspond to the D different covariates. Typically, the first column will be an intercept and fixed to 1, while other columns represent ideal point-specific covariates such as session.

.starts a list containing several matrices of starting values for the parameters. The list should contain the following matrices:
  • alpha A (J x 1) matrix of starting values for the item difficulty parameter $\alpha_j$.
  • beta A (J x 1) matrix of starting values for the item discrimination parameter $\beta_j$.
  • gamma An (I x D) matrix of starting values for the group level coefficients $\gamma_m$.
  • eta An (I x 1) matrix of starting values for the ideal point error term $\eta_i$.
  • sigma An (G x 1) matrix of starting values for the group level variance parameter $\sigma^2_m$.

.priors list, containing several matrices of starting values for the parameters. The list should contain the following matrices:
  • gamma.mu A (D x 1) prior means matrix for all group level coefficients $\gamma_m$.
  • gamma.sigma A (D x D) prior covariance matrix for all group level coefficients $\gamma_m$.
  • beta.mu A (2 x 1) prior means matrix for all bill parameters $\alpha_j$ and $\beta_j$.
  • beta.sigma A (2 x 2) prior covariance matrix for all bill parameters $\alpha_j$ and $\beta_j$.
  • sigma.v A (1 x 1) matrix containing the group level variance prior parameter $\nu_\sigma$.
  • sigma.s A (1 x 1) matrix containing the group level variance prior parameter $s^2_\sigma$.

.control list, specifying some control functions for estimation. Options include the following:
  • threads integer, indicating number of cores to use. Default is to use a single core, but more can be supported if more speed is desired.
• verbose boolean, indicating whether output during estimation should be verbose or not. Set FALSE by default.

• thresh numeric. Algorithm will run until all parameters correlate at 1 - thresh across consecutive iterations. Set at 1e-6 by default.

• maxit integer. Sets the maximum number of iterations the algorithm can run. Set at 500 by default.

• checkfreq integer. Sets frequency of verbose output by number of iterations. Set at 50 by default.

Value

An object of class hierIRT.

means list, containing several matrices of point estimates for the parameters corresponding to the inputs for the priors. The list should contain the following matrices.

• alpha A (J x 1) matrix of point estimates for the item difficulty parameter $\alpha_j$.

• beta A (J x 1) matrix of point estimates for the item discrimination parameter $\beta_j$.

• gamma An (I x D) matrix of point estimates for the group level coefficients $\gamma_m$.

• eta An (I x 1) matrix of point estimates for the ideal point error term $\eta_n$.

• sigma An (G x 1) matrix of point estimates for the group level variance parameter $\sigma^2_m$.

• x_implied An (I x 1) matrix of the implied ideal point $x_i$, calculated as a function of gamma, z, and eta using the point estimates for those parameters.

vars list, containing several matrices of variance estimates for several parameters of interest for diagnostic purposes. Note that these variances are those recovered via variational approximation, and in most cases they are known to be far too small and generally unusable. The list should contain the following matrices:

• eta A (I x 1) matrix of variance estimates for the ideal point noise parameter $\eta_n$.

• gamma A (G x D x D) cube of covariance estimates for the gamma parameters for each group. Each of the G items is a matrix with a single covariance matrix for the m-th group’s D gamma parameters.

• beta2 A (J x 2 x 2) cube of covariance estimates for the item parameters $\alpha_j$ and $\beta_j$. Each of the J items is a matrix with a single covariance matrix for the j-th item.

runtime A list of fit results, with elements listed as follows:

• iters integer, number of iterations run.

• conv integer, convergence flag. Will return 1 if threshold reached, and 0 if maximum number of iterations reached.

• threads integer, number of threads used to estimated model.
tolerance numeric, tolerance threshold for convergence. Identical to thresh argument in input to .control list.

N

A list of counts of various items:

- D integer, number of dimensions (i.e. number of covariates, including intercept).
- G integer, number of groups.
- I integer, number of ideal points.
- J integer, number of items/bill parameters.
- L integer, number of observed votes.

call

Function call used to generate output.

Author(s)

Kosuke Imai <kimai@princeton.edu>
James Lo <lojames@usc.edu>
Jonathan Olmsted <jpolmsted@gmail.com>

References


See Also

'dwnom'.

Examples

### Real data example of US Senate 80-110 (not run)
### Based on voteview.com example of DW-NOMINATE (ftp://voteview.com/dw-nominate.htm)
### We estimate a hierarchical model without noise and a linear time covariate
### This model corresponds very closely to the DW-NOMINATE model

## Not run:
data(dwnom)

## This takes about 10 minutes to run on 8 threads
## You may need to reduce threads depending on what your machine can support
lout <- hierIRT(.data = dwnom$data.in,
                   .starts = dwnom$cur,
                   .priors = dwnom$priors,
                   .control = (list(
                   threads = 8,
                   verbose = TRUE,
                   thresh = 1e-4,
                   maxit=200,
## Bind ideal point estimates back to legislator data
final <- cbind(dwnom$legis, idealpt.hier=lout$means$x_implied)

## These are estimates from DW-NOMINATE as given on the Voteview example
## From file "SL80110C21.DAT"
nomres <- dwnom$nomres

## Merge the DW-NOMINATE estimates to model results by legislator ID
## Check correlation between hierIRT() and DW-NOMINATE scores
res <- merge(final, nomres, by=c("senate","id"),all.x=TRUE,all.y=FALSE)
cor(res$idealpt.hier, res$dwnom1d)

## End(Not run)

---

**makePriors**  
*Generate Priors for binIRT*

**Description**

makePriors generates diffuse priors for binIRT.

**Usage**

```
makePriors(.N = 20, .J = 100, .D = 1)
```

**Arguments**

- `.N` integer, number of subjects/legislators to generate priors for.
- `.J` integer, number of items/bills to generate priors for.
- `.D` integer, number of dimensions.

**Value**

- `x$mu` A (D x D) prior means matrix for respondent ideal points $x_i$.
- `x$sigma` A (D x D) prior covariance matrix for respondent ideal points $x_i$.
- `beta$mu` A (D+1 x 1) prior means matrix for $\alpha_j$ and $\beta_j$.
- `beta$sigma` A (D+1 x D+1) prior covariance matrix for $\alpha_j$ and $\beta_j$.

**Author(s)**

- Kosuke Imai <kimai@princeton.edu>
- James Lo <lojames@usc.edu>
- Jonathan Olmsted <jpolmsted@gmail.com>
manifesto

References


See Also

'binIRT', 'getStarts', 'convertRC'.

Examples

```r
## Data from 109th US Senate
data(s109)

## Convert data and make starts/priors for estimation
rc <- convertRC(s109)
p <- makePriors(rc$n, rc$m, 1)
s <- getStarts(rc$n, rc$m, 1)

## Conduct estimates
lout <- binIRT(.rc = rc,
               .starts = s,
               .priors = p,
               .control = {
                 list(threads = 1,
                   verbose = FALSE,
                   thresh = 1e-6
                 )
               })

## Look at first 10 ideal point estimates
lout$means$x[1:10]
```

manifesto

German Manifesto Data

Description

A word frequency matrix containing word frequencies from 25 German party manifestos between 1990-2005. Obtained from Slapin and Proksch AJPS paper, also used in Lo, Slapin and Proksch.

Usage

data(manifesto)
Value

manifesto list, containing the following elements:

• data.manif Term-document matrix, formatted for input to ordIRT().
• starts.manif Start values, formatted for input to ordIRT().
• priors.manif Priors, formatted for input to ordIRT().

References


See Also

'poisIRT'.

Examples

```r
## Not run:
## Load German Manifesto data
data(manifesto)

## Estimate variational Wordfish model
lout <- poisIRT(.rc = manifesto$data.manif,
               i = 0:(ncol(manifesto$data.manif)-1),
               NI=ncol(manifesto$data.manif),
               .starts = manifesto$starts.manif,
               .priors = manifesto$priors.manif,
               .control = (list(
                   threads = 1,
                   verbose = TRUE,
                   thresh = 1e-6,
                   maxit=1000
                   )))

## Positional Estimates for Parties
lout$means$x

## End(Not run)
```
**mq_data**  
*Martin-Quinn Judicial Ideology Scores*

**Description**

This data set contains materials related to the Martin-Quinn measures of judicial ideology, estimated for every justice serving from October 1937 to October 2013. The materials are based on the July 23, 2014 (2014 Release 01) release of the Supreme Court Database, which contain the votes of each Supreme Court justice on each case heard in the court. The data is set up for input to dynIRT(), and also includes point estimates of the same model obtained using standard MCMC techniques. Start values and priors input to this model are identical to those used in the MCMC estimates, and were provided by Kevin Quinn.

**Usage**

```r
data(mq_data)
```

**Value**

- `mq_data` list, containing the following elements:
  - `data.mq` Justice voting data, formatted for input to dynIRT().
  - `cur.mq` Start values, formatted for input to dynIRT().
  - `priors.mq` Priors, formatted for input to dynIRT().
  - `mq_mcmc` Ideal point estimates with data via MCMC.
  - `justiceName` A vector of names identifying the justice that goes with each estimated ideal point.

**References**

Original model and the example is based off of Andrew Martin and Kevin Quinn, “Dynamic Ideal Point Estimation via Markov Chain Monte Carlo for the U.S. Supreme Court, 1953-1999.” Political Analysis 10(2) 134-153.

Variational model is described in Kosuke Imai, James Lo, and Jonathan Olmsted “Fast Estimation of Ideal Points with Massive Data.”

**See Also**

- `dynIRT`

**Examples**

```r
### Replication of Martin-Quinn Judicial Ideology Scores
### Based on July 23, 2014 (2014 Release 01) release of the Supreme Court Database
### Start values and priors based on replication code provided by Kevin Quinn

data(mq_data)
```
## Estimate dynamic variational model using dynIRT()

```r
lout <- dynIRT(.data = mq_data$data.mq,
               .starts = mq_data$cur.mq,
               .priors = mq_data$priors.mq,
               .control = list(
                 threads = 1,
                 verbose = TRUE,
                 thresh = 1e-6,
                 maxit=500
               ))
```

## Extract estimate from variational model

## Delete point estimates of 0, which are justices missing from that session

```r
vi.out <- c(t(lout$means$x))
vi.out[vi.out==0] <- NA
vi.out <- na.omit(vi.out)
```

## Compare correlation against MCMC-estimated result

```r
cor(vi.out, mq_data$mq_mcmc)
cor(vi.out[mq_data$justiceName != "Douglas"],
    mq_data$mq_mcmc[mq_data$justiceName != "Douglas"])
```

### networkIRT

**Network IRT estimation via EM**

**Description**

networkIRT estimates an IRT model with network in cells. Estimation is conducted using the EM algorithm described in the reference paper below. The algorithm generalizes a model by Slapin and Proksch (2009) that is commonly applied to manifesto data.

**Usage**

```r
networkIRT(.y, .starts = NULL, .priors = NULL, .control = NULL,
            .anchor_subject = NULL, .anchor_item = NULL)
```

**Arguments**

- `.y` matrix, with 1 indicating a valid link and 0 otherwise. Followers (usually voters) are on rows, elites are on columns. No NA values are permitted.
- `.starts` a list containing several matrices of starting values for the parameters. The list should contain the following matrices:
  - alpha A (J x 1) matrix of starting values for politician propensity to be followed $\alpha$.
  - beta A (N x 1) matrix of starting values for follower propensity to follow others $\beta$. 
• w An (J x 1) matrix of starting values for politician ideal points z.
• theta An (N x 1) matrix of starting values for the follower ideal points x.
• gamma An (1 x 1) matrix, should generally be fixed to be 1.

.priors list, containing several matrices of starting values for the parameters. The list should contain the following matrices (1x1) matrices:
• alpha$mu prior mean for α.
• alpha$sigma prior variance for α
• beta$mu prior mean for β.
• beta$sigma prior variance for β.
• w$mu prior mean for z.
• w$sigma prior variance for z
• theta$mu prior mean for x.
• theta$sigma prior variance for x.
• gamma$mu Should be fixed to equal 1.
• gamma$sigma Should be fixed to equal 1.

.control list, specifying some control functions for estimation. Options include the following:
• threads integer, indicating number of cores to use. Default is to use a single core, but more can be supported if more speed is desired.
• verbose boolean, indicating whether output during estimation should be verbose or not. Set FALSE by default.
• thresh numeric. Algorithm will run until all parameters correlate at 1 - thresh across consecutive iterations. Set at 1e-6 by default.
• maxit integer. Sets the maximum number of iterations the algorithm can run. Set at 500 by default.
• checkfreq integer. Sets frequency of verbose output by number of iterations. Set at 50 by default.

.anchor_subject integer, specifying subject to use as identification anchor.

.anchor_item integer, specifying item to use as identification anchor.

Value
An object of class networkIRT.

.means list, containing several matrices of point estimates for the parameters corresponding to the inputs for the priors. The list should contain the following matrices.
• alpha A (J x 1) matrix of point estimates for politician propensity to be followed alpha.
• beta A (N x 1) matrix of point estimates for follower propensity to follow others β.
• w An (J x 1) matrix of point estimates for politician ideal points z.
• theta An (N x 1) matrix of point estimates for the follower ideal points x.
vars list, containing several matrices of variance estimates for parameters corresponding to the inputs for the priors. Note that these variances are those recovered via variational approximation, and in most cases they are known to be far too small and generally unusable. Better estimates of variances can be obtained manually via the parametric bootstrap. The list should contain the following matrices:

- alpha A (J x 1) matrix of variance estimates for politician propensity to be followed $\alpha$.
- beta A (N x 1) matrix of variance estimates for follower propensity to follow others $\beta$.
- w An (J x 1) matrix of variance estimates for politician ideal points $z$.
- theta An (N x 1) matrix of variance estimates for the follower ideal points $x$.

runtime A list of fit results, with elements listed as follows:

- iters integer, number of iterations run.
- conv integer, convergence flag. Will return 1 if threshold reached, and 0 if maximum number of iterations reached.
- threads integer, number of threads used to estimated model.
- tolerance numeric, tolerance threshold for convergence. Identical to thresh argument in input to .control list.

N Number of followers in estimation, should correspond to number of rows in data matrix .y

J Number of politicians in estimation, should correspond to number of columns in data matrix .y

call Function call used to generate output.

Author(s)

Kosuke Imai <kimai@princeton.edu>
James Lo <lojames@usc.edu>
Jonathan Olmsted <jpolmsted@gmail.com>

References


See Also

'ustweet'
Examples

```r
## Not run:
data(ustweet)

## A ridiculously short run to pass CRAN
## For a real test, set maxit to a more reasonable number to reach convergence
lout <- networkIRT(.y = ustweet$data,
      .starts = ustweet$starts,
      .priors = ustweet$priors,
      .control = list(verbose = TRUE,
                      maxit = 3,
                      convtype = 2,
                      thresh = 1e-6,
                      threads = 1)
      ),
      .anchor_item = 43)

## End(Not run)
```

ordIRT

Two-parameter Ordinal IRT estimation via EM

Description

ordIRT estimates an ordinal IRT model with three ordered response categories. Estimation is conducted using the EM algorithm described in the reference paper below. The algorithm will produce point estimates that are comparable to those of MCMCordfactanal, but will do so much more rapidly and also scale better with larger data sets.

Usage

```r
ordIRT(.rc, .starts = NULL, .priors = NULL, .D = 1L, .control = NULL)
```

Arguments

- `.rc` matrix of numeric values containing the data to be scaled. Respondents are assumed to be on rows, and items assumed to be on columns, so the matrix is assumed to be of dimension (N x J). For each item, only 3 ordered category responses are accepted, and the only allowable responses are ‘1’, ‘2’, and ‘3’, with ‘0’ as a missing data record. If data of more than 3 categories are to be rescaled, they should be collapsed into 3 categories and recoded accordingly before proceeding.
starts

A list containing several matrices of starting values for the parameters. Note that the parameters here correspond to the re-parameterized version of the model (i.e. alpha is $\alpha^*$, not the original $\alpha_{1j}$). The list should contain the following matrices:

- beta A (J x 1) matrix of starting values for the reparameterized item discrimination parameter $\beta^*$.
- x An (N x 1) matrix of starting values for the respondent ideal points $x_t$.
- tau A (J x 1) matrix of starting values for the bill cutpoint $\tau_j$.
- Delta_sq A (J x 1) matrix of starting values for the squared bill cutpoint difference $\tau_j^2$.

priors

A list, containing several matrices of starting values for the parameters. Note that the parameters here correspond to the re-parameterized version of the model (i.e. alpha is $\alpha^*$, not the original $\alpha_{1j}$). The list should contain the following matrices:

- x$mu$ A (1 x 1) prior means matrix for respondent ideal points $x_t$.
- x$sigma$ A (1 x 1) prior covariance matrix for respondent ideal points $x_t$.
- beta$mu$ A (2 x 1) prior means matrix for $\tau_j$ and $\beta^*$.
- beta$sigma$ A (2 x 2) prior covariance matrix for $\tau_j$ and $\beta^*$.

D

An integer, indicates number of dimensions. Only one dimension is implemented and this argument is ignored.

.control

A list, specifying some control functions for estimation. Options include the following:

- threads integer, indicating number of cores to use. Default is to use a single core, but more can be supported if more speed is desired.
- verbose boolean, indicating whether output during estimation should be verbose or not. Set FALSE by default.
- thresh numeric. Algorithm will run until all parameters correlate at 1 - thresh across consecutive iterations. Set at 1e-6 by default.
- maxit integer. Sets the maximum number of iterations the algorithm can run. Set at 500 by default.
- checkfreq integer. Sets frequency of verbose output by number of iterations. Set at 50 by default.

Value

An object of class ordIRT.

means

A list, containing several matrices of point estimates for the parameters corresponding to the inputs for the priors. The list should contain the following matrices:

- x A (N x 1) matrix of point estimates for the respondent ideal points $x_t$.
- beta A (J x 1) matrix of point estimates for the reparameterized item discrimination parameter $\beta^*$.
- tau A (J x 1) matrix of point estimates for the bill cutpoint $\alpha^*$.
- Delta_sq A (J x 1) matrix of point estimates for the squared bill cutpoint difference $\tau_j^2$. 
Delta A (J x 1) matrix of point estimates for the bill cutpoint difference $\tau_j$.

vars list, containing several matrices of variance estimates for parameters corresponding to the inputs for the priors. Note that these variances are those recovered via variational approximation, and in most cases they are known to be far too small and generally unusable. Better estimates of variances can be obtained manually via the parametric bootstrap. The list should contain the following matrices:

- $x$ A (N x 1) matrix of variances for the respondent ideal points $x_t$.
- beta $A$ (J x 1) matrix of variances for the reparameterized item discrimination parameter $\beta^*$. 
- tau $A$ (J x 1) matrix of variances for the bill cutpoint $\tau_j$.

runtime A list of fit results, with elements listed as follows:

- iters integer, number of iterations run.
- conv integer, convergence flag. Will return 1 if threshold reached, and 0 if maximum number of iterations reached.
- threads integer, number of threads used to estimated model.
- tolerance numeric, tolerance threshold for convergence. Identical to thresh argument in input to .control list.

n Number of respondents in estimation, should correspond to number of rows in roll call matrix.

j Number of items in estimation, should correspond to number of columns in roll call matrix.

call Function call used to generate output.

Author(s)
Kosuke Imai <kimai@princeton.edu>
James Lo <lojames@usc.edu>
Jonathan Olmsted <jpolmsted@gmail.com>

References

See Also
'AsahiTodai'.

Examples

## Not run:
### Real data example: Asahi-Todai survey (not run)
## Collapses 5-category ordinal survey items into 3 categories for estimation
data(AsahiTodai)
out.varinf <- ordIRT(.rc = AsahiTodai$dat.all, .starts = AsahiTodai$start.values, .priors = AsahiTodai$priors, .D = 1, .control = {list(verbose = TRUE, thresh = 1e-6, maxit = 500)})

## Compare against MCMC estimates using 3 and 5 categories
cor(ideal3, out.varinf$means$x)
cor(ideal5, out.varinf$means$x)

## End(Not run)

### Monte Carlo simulation of ordIRT() model vs. known parameters

## Set number of legislators and items
set.seed(2)
NN <- 500
JJ <- 100

## Simulate true parameters from original model
x.true <- runif(NN, -2, 2)
beta.true <- runif(JJ, -1, 1)
tau1 <- runif(JJ, -1.5, -0.5)
tau2 <- runif(JJ, 0.5, 1.5)
ystar <- x.true %o% beta.true + rnorm(NN *JJ)

## These parameters are not needed, but correspond to reparameterized model
#d.true <- tau2 - tau1
#dd.true <- d.true^2
#tau_star <- -tau1/d.true
#beta_star <- beta.true/d.true

## Generate roll call matrix using simulated parameters
newrc <- matrix(0, NN, JJ)
for(j in 1:JJ) newrc[,j] <- cut(ystar[,j], c(-100, tau1[j], tau2[j],100), labels=FALSE)

## Generate starts and priors
cur <- vector(mode = "list")
cur$DD <- matrix(rep(0.5,JJ), ncol=1)
cur$tau <- matrix(rep(-0.5,JJ), ncol=1)
cur$beta <- matrix(runif(JJ,-1,1), ncol=1)
cur$x <- matrix(runif(NN,-1,1), ncol=1)
priors <- vector(mode = "list")
priors$x <- list(mu = matrix(0,1,1), sigma = matrix(1,1,1) )
priors$beta <- list(mu = matrix(0,2,1), sigma = matrix(diag(25),2,2))

## Call ordIRT() with inputs
#time <- system.time({
#  lout <- ordIRT(.rc = newrc, .starts = cur, .priors = priors, .control = {list(thread = 1, verbose = TRUE),

## Monte Carlo simulation of ordIRT() model vs. known parameters

## Simulate true parameters from original model
x.true <- runif(NN, -2, 2)
beta.true <- runif(JJ, -1, 1)
tau1 <- runif(JJ, -1.5, -0.5)
tau2 <- runif(JJ, 0.5, 1.5)
ystar <- x.true %o% beta.true + rnorm(NN *JJ)

## These parameters are not needed, but correspond to reparameterized model
#d.true <- tau2 - tau1
#dd.true <- d.true^2
#tau_star <- -tau1/d.true
#beta_star <- beta.true/d.true

## Generate roll call matrix using simulated parameters
newrc <- matrix(0, NN, JJ)
for(j in 1:JJ) newrc[,j] <- cut(ystar[,j], c(-100, tau1[j], tau2[j],100), labels=FALSE)

## Generate starts and priors
cur <- vector(mode = "list")
cur$DD <- matrix(rep(0.5,JJ), ncol=1)
cur$tau <- matrix(rep(-0.5,JJ), ncol=1)
cur$beta <- matrix(runif(JJ,-1,1), ncol=1)
cur$x <- matrix(runif(NN,-1,1), ncol=1)
priors <- vector(mode = "list")
priors$x <- list(mu = matrix(0,1,1), sigma = matrix(1,1,1) )
priors$beta <- list(mu = matrix(0,2,1), sigma = matrix(diag(25),2,2))

## Call ordIRT() with inputs
#time <- system.time({
#  lout <- ordIRT(.rc = newrc, .starts = cur, .priors = priors, .control = {list(thread = 1, verbose = TRUE),

## Monte Carlo simulation of ordIRT() model vs. known parameters

## Simulate true parameters from original model
x.true <- runif(NN, -2, 2)
beta.true <- runif(JJ, -1, 1)
tau1 <- runif(JJ, -1.5, -0.5)
tau2 <- runif(JJ, 0.5, 1.5)
ystar <- x.true %o% beta.true + rnorm(NN *JJ)

## These parameters are not needed, but correspond to reparameterized model
#d.true <- tau2 - tau1
#dd.true <- d.true^2
#tau_star <- -tau1/d.true
#beta_star <- beta.true/d.true

## Generate roll call matrix using simulated parameters
newrc <- matrix(0, NN, JJ)
for(j in 1:JJ) newrc[,j] <- cut(ystar[,j], c(-100, tau1[j], tau2[j],100), labels=FALSE)

## Generate starts and priors
cur <- vector(mode = "list")
cur$DD <- matrix(rep(0.5,JJ), ncol=1)
cur$tau <- matrix(rep(-0.5,JJ), ncol=1)
cur$beta <- matrix(runif(JJ,-1,1), ncol=1)
cur$x <- matrix(runif(NN,-1,1), ncol=1)
priors <- vector(mode = "list")
priors$x <- list(mu = matrix(0,1,1), sigma = matrix(1,1,1) )
priors$beta <- list(mu = matrix(0,2,1), sigma = matrix(diag(25),2,2))

## Call ordIRT() with inputs
#time <- system.time({
Description

poisIRT estimates an IRT model with count (usually word counts) in cells. Estimation is conducted using the EM algorithm described in the reference paper below. The algorithm generalizes a model by Slapin and Proksch (2009) that is commonly applied to manifesto data.

Usage

poisIRT (.rc, i = 0:(nrow(.rc)-1), NI = nrow(.rc), .starts = NULL, .priors = NULL, .control = NULL)

Arguments

.rc matrix, usually with unique words along the J rows and different documents across K columns. Each cell will contain a count of words. There should be no NA values, so documents missing a particular word should list 0 in the cell.

.i vector of length K, indicating for each of the K documents which actor it belongs to. Assignment of actors begins at actor 0. If set to 0:(K-1), and NI=K below, then each assignment is assigned its own ideal point, and we get the Slapin and Proksch Wordfish model.

.NI integer, number of unique actors. Must be less than or equal to K. If NI=K, then each document is assigned its own ideal point, and we get the Slapin and Proksch Wordfish model.

.starts a list containing several matrices of starting values for the parameters. The list should contain the following matrices:

- alpha A (J x 1) matrix of starting values for the word frequency parameter \( \alpha \).
- psi A (K x 1) matrix of starting values for the document verbosity parameter \( \psi \).
- beta A (J x 1) matrix of starting values for the word discrimination parameter \( \beta \).
- x An (NI x 1) matrix of starting values for the actor ideal points \( x_i \).
.priors list, containing several matrices of starting values for the parameters. The list should contain the following matrices:

- $x$\textit{mu} numeric, prior mean for actor ideal points $x_i$.
- $x$\textit{sigma2} numeric, prior variance for actor ideal points $x_i$.
- $\beta$\textit{mu} numeric, prior mean for $\beta_j$.
- $\beta$\textit{sigma2} numeric, prior variance for $\beta_j$.
- $\alpha$\textit{mu} numeric, prior mean for $\alpha_j$.
- $\alpha$\textit{sigma2} numeric, prior variance for $\alpha_j$.
- $\psi$\textit{mu} numeric, prior mean for $\psi_k$.
- $\psi$\textit{sigma2} numeric, prior variance for $\psi_k$.

.control list, specifying some control functions for estimation. Options include the following:

- threads integer, indicating number of cores to use. Default is to use a single core, but more can be supported if more speed is desired.
- verbose boolean, indicating whether output during estimation should be verbose or not. Set FALSE by default.
- thresh numeric. Algorithm will run until all parameters correlate at 1 - thresh across consecutive iterations. Set at 1e-6 by default.
- maxit integer. Sets the maximum number of iterations the algorithm can run. Set at 500 by default.
- checkfreq integer. Sets frequency of verbose output by number of iterations. Set at 50 by default.

Value

An object of class poisIRT.

means list, containing several matrices of point estimates for the parameters corresponding to the inputs for the priors. The list should contain the following matrices.

- alpha A (J x 1) matrix of point estimates for the word frequency parameter $\alpha$.
- psi A (K x 1) matrix of point estimates for the document verbosity parameter $\psi$.
- beta A (J x 1) matrix of point estimates for the word discrimination parameter $\beta$.
- $x$ An (NI x 1) matrix of point estimates for the actor ideal points $x_i$.

vars list, containing several matrices of variance estimates for parameters corresponding to the inputs for the priors. Note that these variances are those recovered via variational approximation, and in most cases they are known to be far too small and generally unusable. Better estimates of variances can be obtained manually via the parametric bootstrap. The list should contain the following matrices:

- beta A (J x 1) matrix of variational variance estimates for the word discrimination parameter $\beta$. 
poisIRT

• \( \mathbf{x} \) An \((NI \times 1)\) matrix of variational variance estimates for the actor ideal points \( x_i \).

runtime

A list of fit results, with elements listed as follows:

• iters integer, number of iterations run.
• conv integer, convergence flag. Will return 1 if threshold reached, and 0 if maximum number of iterations reached.
• threads integer, number of threads used to estimated model.
• tolerance numeric, tolerance threshold for convergence. Identical to thresh argument in input to .control list.

N

A list of sizes, with elements listed as follow:

• K Number of unique words in term-document matrix.
• J Number of documents in term-document matrix.
• I Number of actors in model, always less than or equal to J.
• call Function call used to generate output.

i_of_k

A copy of input for argument ‘i’, which allows the J documents to be linked to I actors.

Author(s)

Kosuke Imai <kimai@princeton.edu>
James Lo <lojames@usc.edu>
Jonathan Olmsted <jpolmsted@gmail.com>

References


See Also

'manifesto'

Examples

```r
## Not run:
## Load German Manifesto data
data(manifesto)

## Estimate variational Wordfish model
lout <- poisIRT(.rc = manifesto$data.manif,
i = 0:(ncol(manifesto$data.manif)-1),
NI=ncol(manifesto$data.manif),
   .starts = manifesto$starts.manif,
   .priors = manifesto$priors.manif,
   .control = list(
       threads = 1,
```
ustweet

```r
verbose = TRUE,
thresh = 1e-6,
maxit=1000
})
```

## Positional Estimates for Parties
lout$means$x

## End(Not run)

---

**ustweet**  
*U.S. Twitter Following Data*

**Description**

Data from U.S. Twitter follower data, obtained from Barbera’s (2015) replication archive.

**Usage**

data(ustweet)

**Value**

- **ustweet** list, containing the following elements:
  - *data* Term-document matrix, formatted for input to `networkIRT()`.
  - *starts* Start values, formatted for input to `networkIRT()`.
  - *priors* Priors, formatted for input to `networkIRT()`.

**References**


**See Also**

`networkIRT`.

**Examples**

```r
## Not run:
data(ustweet)

## A ridiculously short run to pass CRAN
## For a real test, set maxit to a more reasonable number to reach convergence
```
lout <- networkIRT(.y = ustweet$data,
                   .starts = ustweet$starts,
                   .priors = ustweet$priors,
                   .control = {list(verbos = TRUE,
                                 maxit = 3,
                                 convtype = 2,
                                 thresh = 1e-6,
                                 threads = 1
                               )
                   },
                   .anchor_item = 43
)

## End(Not run)
Index

*Topic datasets
  AsahiTodai, 2
dwnom, 10
manifesto, 21
mq_data, 23
ustweet, 34

*Topic multivariate
  binIRT, 3
  boot_emIRT, 7
  convertRC, 8
dynIRT, 11
getStarts, 15
hierIRT, 16
makePriors, 20
networkIRT, 24
ordIRT, 27
poisIRT, 31
AsahiTodai, 2, 29
binIRT, 3, 8, 9, 16, 21
boot_emIRT, 7
convertRC, 5, 8, 16, 21
dwnom, 10, 19
dynIRT, 8, 11, 23
getStarts, 5, 9, 15, 21
hierIRT, 8, 10, 16
ideal, 3
makePriors, 5, 9, 16, 20
manifesto, 21, 33
MCMCdynamicIRT1d, 11
MCMCordfactanal, 27
mq_data, 14, 23
networkIRT, 24, 34
ordIRT, 3, 8, 27
poisIRT, 22, 31
rollcall, 8, 9
ustweet, 26, 34