0.1 `factor.mix`: Mixed Data Factor Analysis

Mixed data factor analysis takes both continuous and ordinal dependent variables and estimates a model for a given number of latent factors. The model is estimated using a Markov Chain Monte Carlo algorithm (Gibbs sampler with data augmentation). Alternative models include Bayesian factor analysis for continuous variables (Section **??**) and Bayesian factor analysis for ordinal variables (Section **??**).

Syntax

```
> z.out <- zelig(cbind(Y1 ,Y2, Y3) ~ NULL, factors = 1,
                 model = "factor.mix", data = mydata)
```

Inputs

`zelig()` accepts the following arguments for `factor.mix`:

- `Y1`, `Y2`, `Y3`, `...`: The dependent variables of interest, which can be a mix of ordinal and continuous variables. You must have more dependent variables than factors.
- `factors`: The number of the factors to be fitted.

Additional Inputs

The model accepts the following additional arguments to monitor convergence:

- `lambda.constraints`: A list that contains the equality or inequality constraints on the factor loadings.
  - `varname = list()`: by default, no constraints are imposed.
  - `varname = list(d, c)`: constrains the dth loading for the variable named `varname` to be equal to `c`.
  - `varname = list(d, "+")`: constrains the dth loading for the variable named `varname` to be positive;
  - `varname = list(d, "-")`: constrains the dth loading for the variable named `varname` to be negative.

  Unlike Bayesian factor analysis for continuous variables (Section **??**), the first column of Λ corresponds to negative item difficulty parameters and should not be constrained in general.

- `std.mean`: defaults to `TRUE`, which rescales the continuous manifest variables to have mean 0.
- `std.var`: defaults to `TRUE`. which rescales the continuous manifest variables to have unit variance.

`factor.mix` accepts the following additional arguments to monitor the sampling scheme for the Markov chain:

- `burnin`: number of the initial MCMC iterations to be discarded. The default value is 1,000.
- `mcmc`: number of the MCMC iterations after burnin. The default value is 20,000.
- `thin`: thinning interval for the Markov chain. Only every `thin`-th draw from the Markov chain is kept. The value of `mcmc` must be divisible by this value. The default value is 1.
- `tune`: tuning parameter, which can be either a scalar or a vector of length K. The value of the tuning parameter must be positive. The default value is `1.2`.
- `verbose`: defaults to `FALSE`. If `TRUE`, the progress of the sampler (every 10%) is printed to the screen. The default is `FALSE`.
- `seed`: seed for the random number generator. The default is `NA` which corresponds to a random seed 12345.
- `lambda.start`: starting values of the factor loading matrix Λ for the Markov chain, either a scalar (starting values of the unconstrained loadings will be set to that value), or a matrix with compatible dimensions. The default is `NA`, where the start values for the first column of Λ are set based on the observed pattern, while for the rest of the columns of Λ, the start values are set to be 0 for unconstrained factor loadings, and 1 or -1 for constrained factor loadings (depending on the nature of the constraints).
- `psi.start`: starting values for the diagonals of the error variance (uniquenesses) matrix. Since the starting values for the ordinal variables are constrained to 1 (to identify the model), you may only specify the starting values for the continuous variables. For the continuous variables, you may specify `psi.start` as a scalar or a vector with length equal to the number of continuous variables. If a scalar, that starting value is recycled for all continuous variables. If a vector, the starting values should correspond to each of the continuous variables. The default value is `NA`, which means the starting values of all the continuous variable uniqueness are set to 0.5.
- `store.lambda`: defaults to `TRUE`, storing the posterior draws of the factor loadings.

- `store.scores`: defaults to `FALSE`. If `TRUE`, the posterior draws of the factor scores are stored. (Storing factor scores may take large amount of memory for a a large number of draws or observations.)

Use the following additional arguments to specify prior parameters used in the model:

- `l0`: mean of the Normal prior for the factor loadings, either a scalar or a matrix with the same dimensions as Λ. If a scalar value, then that value will be the prior mean for all the factor loadings. The default value is 0.

- `L0`: precision parameter of Normal prior for the factor loadings, either a scalar or a matrix with the same dimensions as Λ. If a scalar value, then the precision matrix will be a diagonal matrix with the diagonal elements set to that value. The default value is 0 which leads to an improper prior.

- `a0`: `a0/2` is the shape parameter of the Inverse Gamma priors for the uniquenesses. It can take a scalar value or a vector. The default value is 0.001.

- `b0`: `b0/2` is the shape parameter of the Inverse Gamma priors for the uniquenesses. It can take a scalar value or a vector. The default value is 0.001.

Zelig users may wish to refer to `help(MCMCmixfactanal)` for more information.

Convergence

Users should verify that the Markov Chain converges to its stationary distribution. After running the `zelig()` function but before performing `setx()`, users may conduct the following convergence diagnostics tests:

- `geweke.diag(z.out$coefficients)`: The Geweke diagnostic tests the null hypothesis that the Markov chain is in the stationary distribution and produces z-statistics for each estimated parameter.

- `heidel.diag(z.out$coefficients)`: The Heidelberger-Welch diagnostic first tests the null hypothesis that the Markov Chain is in the stationary distribution and produces p-values for each estimated parameter. Calling `heidel.diag()` also produces output that indicates whether the mean of a marginal posterior distribution can be estimated with sufficient precision, assuming that the Markov Chain is in the stationary distribution.

- `raftery.diag(z.out$coefficients)`: The Raftery diagnostic indicates how long the Markov Chain should run before considering draws from the marginal posterior distributions sufficiently representative of the stationary distribution.

If there is evidence of non-convergence, adjust the values for `burnin` and `mcmc` and rerun `zelig()`.

Advanced users may wish to refer to `help(geweke.diag)`, `help(heidel.diag)`, and `help(raftery.diag)` for more information about these diagnostics.

Examples

1. Basic Example
   Attaching the sample dataset:

   ```
   > data(PErisk)
   ```

   Factor analysis for mixed data using `factor.mix`:

   ```
   > z.out <- zelig(cbind(courts, barb2, prsexp2, prscorr2, gdpw2) ~
   +     NULL, data = PErisk, model = "factor.mix", factors = 1, burnin = 5000,
   +     mcmc = 1e+05, thin = 50, verbose = TRUE, L0 = 0.25, tune = 1.2)
   ```

   Checking for convergence before summarizing the estimates:

   ```
   > geweke.diag(z.out$coefficients)

   > heidel.diag(z.out$coefficients)

   > summary(z.out)
   ```

Model

Let Y_i be a K-vector of observed variables for observation i, The kth variable can be either continuous or ordinal. When Y_{ik} is an ordinal variable, it takes value from 1 to J_k for $k = 1, \ldots, K$ and for $i = 1, \ldots, n$. The distribution of Y_{ik} is assumed to be governed by another K-vector of unobserved continuous variable Y^*_{ik}. There are d underlying factors. When Y_{ik} is continuous, we let $Y^*_{ik} = Y_{ik}$.

- The *stochastic component* is described in terms of Y^*_i:

  $$Y^*_i \sim \text{Normal}_K(\mu_i, I_K),$$

  where $Y^*_i = (Y^*_{i1}, \ldots, Y^*_{iK})$, and $\mu_i = (\mu_{i1}, \ldots, \mu_{iK})$.

  For ordinal Y_{ik},

  $$Y_{ik} = j \quad \text{if} \quad \gamma_{(j-1),k} \leq Y^*_{ik} \leq \gamma_{jk} \quad \text{for} \quad j = 1, \ldots, J_k; k = 1, \ldots, K.$$

  where $\gamma_{jk}, j = 0, \ldots, J$ are the threshold parameters for the kth variable with the following constraints, $\gamma_{lk} < \gamma_{mk}$ for $l < m$, and $\gamma_{0k} = -\infty, \gamma_{J_k k} = \infty$ for any $k = 1, \ldots, K$. It follows that the probability of observing Y_{ik} belonging to category j is,

  $$\Pr(Y_{ik} = j) = \Phi(\gamma_{jk} \mid \mu_{ik}) - \Phi(\gamma_{(j-1),k} \mid \mu_{ik}) \quad \text{for } j = 1, \ldots, J_k$$

  where $\Phi(\cdot \mid \mu_{ik})$ is the cumulative distribution function of the Normal distribution with mean μ_{ik} and variance 1.

- The *systematic component* is given by,

$$\mu_i \quad = \quad \Lambda\phi_i,$$

  where Λ is a $K \times d$ matrix of factor loadings for each variable, ϕ_i is a d-vector of factor scores for observation i. Note both Λ and ϕ are estimated..

- The independent conjugate *prior* for each Λ_{ij} is given by

$$\Lambda_{ij} \quad \sim \quad \text{Normal}(l_{0_{ij}}, L_{0_{ij}}^{-1}) \text{ for } i = 1, \ldots, k; \quad j = 1, \ldots, d.$$

- The *prior* for ϕ_i is,

$$\phi_i \sim \text{Normal}(0, I_{d-1}), \quad \text{for} \quad i = 2, \ldots, n.$$

  where I_{d-1} is a $(d-1) \times (d-1)$ identity matrix. Note the first element of ϕ_i is 1.

Output Values

The output of each Zelig command contains useful information which you may view. For example, if you run:

```
z.out <- zelig(cbind(Y1, Y2, Y3), model = "factor.mix", data)
```

then you may examine the available information in `z.out` by using `names(z.out)`, see the draws from the posterior distribution of the `coefficients` by using `z.out$coefficients`, and view a default summary of information through `summary(z.out)`. Other elements available through the `$` operator are listed below.

- From the `zelig()` output object `z.out`, you may extract:
  - `coefficients`: draws from the posterior distributions of the estimated factor loadings, the estimated cut points γ for each variable. Note the first element of γ is normalized to be 0. If `store.scores = TRUE`, the estimated factors scores are also contained in `coefficients`.
  - `zelig.data`: the input data frame if `save.data = TRUE`.
  - `seed`: the random seed used in the model.

- Since there are no explanatory variables, the `sim()` procedure is not applicable for factor analysis models.

How to Cite

To cite the *factor.mix* Zelig model:

> Ben Goodrich and Ying Lu. 2007. "factor.mix: Mixed Data Factor Analysis ," in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software," `http://gking.harvard.edu/zelig`.

To cite Zelig as a whole, please reference these two sources:

> Kosuke Imai, Gary King, and Olivia Lau. 2007. "Zelig: Everyone's Statistical Software," `http://GKing.harvard.edu/zelig`.

> Kosuke Imai, Gary King, and Olivia Lau. 2007. "Toward A Common Framework for Statistical Analysis and Development," `http://gking.harvard.edu/files/abs/z-abs.shtml`.