0.1 `factor.bayes`: Bayesian Factor Analysis

Given some unobserved explanatory variables and observed dependent variables, the Normal theory factor analysis model estimates the latent factors. The model is implemented using a Markov Chain Monte Carlo algorithm (Gibbs sampling with data augmentation). For factor analysis with ordinal dependent variables, see ordered factor analysis (Section **??**), and for a mix of types of dependent variables, see the mixed factor analysis model (Section **??**).

Syntax

```
> z.out <- zelig(cbind(Y1 ,Y2, Y3) ~ NULL, factors = 2,
                 model = "factor.bayes", data = mydata)
```

Inputs

`zelig()` takes the following functions for `factor.bayes`:

- `Y1`, `Y2`, and `Y3`: variables of interest in factor analysis (manifest variables), assumed to be normally distributed. The model requires a minimum of three manifest variables.
- `factors`: number of the factors to be fitted (defaults to 2).

Additional Inputs

In addition, `zelig()` accepts the following additional arguments for model specification:

- `lambda.constraints`: list containing the equality or inequality constraints on the factor loadings. Choose from one of the following forms:
  - `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.
- `std.var`: defaults to `FALSE` (manifest variables are rescaled to zero mean, but retain observed variance). If `TRUE`, the manifest variables are rescaled to be mean zero and unit variance.

In addition, `zelig()` accepts the following additional inputs for `bayes.factor`:

- `burnin`: number of the initial MCMC iterations to be discarded (defaults to 1,000).
- `mcmc`: number of the MCMC iterations after burnin (defaults to 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.

- `verbose`: defaults to `FALSE`. If `TRUE`, the progress of the sampler (every 10%) is printed to the screen.

- `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, either a scalar (all unconstrained loadings are set to that value), or a matrix with compatible dimensions. The default is `NA`, where the start value are set to be 0 for unconstrained factor loadings, and 0.5 or -0.5 for constrained factor loadings (depending on the nature of the constraints).

- `Psi.start`: starting values for the uniquenesses, either a scalar (the starting values for all diagonal elements of Ψ are set to be this value), or a vector with length equal to the number of manifest variables. In the latter case, the starting values of the diagonal elements of Ψ take the values of `Psi.start`. The default value is `NA` where the starting values of the all the uniquenesses are set to be 0.5.

- `store.lambda`: defaults to `TRUE`, which stores the posterior draws of the factor loadings.

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

The model also accepts the following additional arguments to specify prior parameters:

- `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, that value will be the prior mean for all the factor loadings. Defaults to 0.

- `L0`: precision parameter of the Normal prior for the factor loadings, either a scalar or a matrix with the same dimensions as Λ. If `L0` takes 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`: the shape parameter of the Inverse Gamma prior for the uniquenesses is `a0/2`. It can take a scalar value or a vector. The default value is 0.001.

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

Zelig users may wish to refer to `help(MCMCfactanal)` 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(swiss)
   > names(swiss) <- c("Fert", "Agr", "Exam", "Educ", "Cath", "InfMort")
   ```

   Factor analysis:

   ```
   > z.out <- zelig(cbind(Agr, Exam, Educ, Cath, InfMort) ~ NULL,
   +     model = "factor.bayes", data = swiss, factors = 2, verbose = TRUE,
   +     a0 = 1, b0 = 0.15, burnin = 5000, mcmc = 50000)
   ```

   Checking for convergence before summarizing the estimates:

   ```
   > algor <- try(geweke.diag(z.out$coefficients), silent = T)
   > if (class(algor) == "try-error") print(algor)
   ```

   Since the algorithm did not converge, we now add some constraints on Λ.

2. Putting Constraints on Λ
   Put constraints on Lambda to optimize the algorithm:

```
> z.out <- zelig(cbind(Agr, Exam, Educ, Cath, InfMort) ~ NULL,
+     model = "factor.bayes", data = swiss, factors = 2, lambda.constraints = list
+         "+"), Exam = list(2, "-"), Educ = c(2, 0), InfMort = c(1,
+         0)), verbose = TRUE, a0 = 1, b0 = 0.15, burnin = 5000,
+     mcmc = 50000)
> geweke.diag(z.out$coefficients)
> heidel.diag(z.out$coefficients)
> raftery.diag(z.out$coefficients)
> summary(z.out)
```

Model

Suppose for observation i we observe K variables and hypothesize that there are d underlying factors such that:

$$Y_i = \Lambda\phi_i + \epsilon_i$$

where Y_i is the vector of K manifest variables for observation i. Λ is the $K \times d$ factor loading matrix and ϕ_i is the d-vector of latent factor scores. Both Λ and ϕ need to be estimated.

- The *stochastic component* is given by:

  $$\epsilon_i \sim \text{Normal}(0, \Psi).$$

  where Ψ is a diagonal, positive definite matrix. The diagonal elements of Ψ are referred to as uniquenesses.

- The *systematic component* is given by

  $$\mu_i = E(Y_i) = \Lambda\phi_i$$

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

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

- The independent conjugate *prior* for each Ψ_{ii} is given by

  $$\Psi_{ii} \sim \text{InverseGamma}(\frac{a_0}{2}, \frac{b_0}{2}), \text{ for } i = 1, \ldots, k.$$

- The *prior* for ϕ_i is

  $$\phi_i \sim \text{Normal}(0, I_d), \text{ for } i = 1, \ldots, n.$$

  where I_d is a $d \times d$ identity matrix.

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.bayes", 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 and the uniquenesses. If `store.scores = TRUE`, the estimated factors scores are also contained in `coefficients`.
  - `data`: the name of the input data frame.
  - `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.bayes* Zelig model:

> Ben Goodrich and Ying Lu. 2007. "factor.bayes: Bayesian 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`.