Package 'factor.switching'

July 13, 2021

Type Package

Title Post-Processing MCMC Outputs of Bayesian Factor Analytic Models

Version 1.2

Date 2021-07-12

Maintainer Panagiotis Papastamoulis <papapast@yahoo.gr>

Description A well known identifiability issue in factor analytic models is the invariance with respect to orthogonal transformations. This problem burdens the inference under a Bayesian setup, where Markov chain Monte Carlo (MCMC) methods are used to generate samples from the posterior distribution. The package applies a series of rotation, sign and permutation transformations (Papastamoulis and Ntzoufras (2020) <arXiv:2004.05105>) into raw MCMC samples of factor loadings, which are provided by the user. The post-processed output is identifiable and can be used for MCMC inference on any parametric function of factor loadings. Comparison of multiple MCMC chains is also possible.

Imports coda, HDInterval, lpSolve , MCMCpack

License GPL-2

NeedsCompilation no

Author Panagiotis Papastamoulis [aut, cre]
(<https://orcid.org/0000-0001-9468-7613>)

Repository CRAN

Date/Publication 2021-07-12 22:10:07 UTC

R topics documented:

`factor.switching-package`

Post-Processing MCMC Outputs of Bayesian Factor Analytic Models

---

Description

A well known identifiability issue in factor analytic models is the invariance with respect to orthogonal transformations. This problem burdens the inference under a Bayesian setup, where Markov chain Monte Carlo (MCMC) methods are used to generate samples from the posterior distribution. The package applies a series of rotation, sign and permutation transformations (Papastamoulis and Ntzoufras (2020) <arXiv:2004.05105>) into raw MCMC samples of factor loadings, which are provided by the user. The post-processed output is identifiable and can be used for MCMC inference on any parametric function of factor loadings. Comparison of multiple MCMC chains is also possible.

There are three alternative schemes for minimizing the objective function.

1. Exact `rsp_exact`
2. Partial Simulated Annealing `rsp_partial_sa`
3. Full simulated annealing `rsp_full_sa`

The exact algorithm solves 2^q assignment problems per MCMC iteration, where q denotes the number of factors of the fitted model. For typical values of the number of factors (e.g. $q < 11$) the exact scheme should be preferred. Otherwise, the two approximate algorithms based on simulated annealing may be considered. The Partial simulated annealing is more efficient than the full simulated annealing scheme.

In cases of parallel MCMC chains, applying the RSP algorithm for each chain separately will identify the factor loadings within each chain. However, the results will not be comparable between chains. The comparison of multiple MCMC chains is doable via the `compareMultipleChains` function.

Details

The DESCRIPTION file:

Index of help topics:

```
compareMultipleChains   Compare multiple chains
credible.region         Compute a simultaneous credible region
                        (rectangle) from a sample for a vector valued
                        parameter.
factor.switching-package
                        Post-Processing MCMC Outputs of Bayesian Factor
                        Analytic Models
```

plot.rsp	Plot posterior means and credible regions
procrustes_switching	Orthogonal Procrustes rotations
rsp_exact	Rotation-Sign-Permutation (RSP) algorithm (Exact scheme)
rsp_full_sa	Rotation-Sign-Permutation (RSP) algorithm (Full Simulated Annealing)
rsp_partial_sa	Rotation-Sign-Permutation (RSP) algorithm (Partial Simulated Annealing)
small_posterior_2chains	Example data
switch_and_permute	Apply sign switchings and column permutations
weighted_procrustes_switching	Weighted Orthogonal Procrustes rotations

Author(s)

Panagiotis Papastamoulis

Maintainer: Panagiotis Papastamoulis

References

Papastamoulis, P. and Ntzoufras, I. (2020). On the identifiability of Bayesian Factor Analytic models. *arXiv:2004.05105 [stat.ME]*.

See Also

rsp_exact, plot.rsp, compareMultipleChains

Examples

```
# load 2 chains each one consisting of a
# small mcmc sample of 100 iterations
# with p=6 variables and q=2 factors.
data(small_posterior_2chains)
Nchains <- length(small_posterior_2chains)
reorderedPosterior <- vector('list',length=Nchains)
# post-process the 2 chains
for(i in 1:Nchains){
reorderedPosterior[[i]] <- rsp_exact( lambda_mcmc = small_posterior_2chains[[i]],
maxIter = 100,
threshold = 1e-6,
verbose=TRUE )
}
# plot posterior summary for chain 1:
plot(reorderedPosterior[[1]])
# plot posterior summary for chain 2:
plot(reorderedPosterior[[2]])
# make them comparable
makeThemSimilar <- compareMultipleChains(rspObjectList=reorderedPosterior)
# plot the traces of both chains
oldpar <- par(no.readonly =TRUE)
```

```
par(mfcol=c(2,6),mar=c(4,4,2,1))
plot(makeThemSimilar,auto.layout=FALSE,density=FALSE,
ylim=c(-1.1,1.1),smooth=FALSE,col=c('red','blue'))
legend('topright',c('post-processed chain 1',
'post-processed chain 2'),lty=1:2,col=c('red','blue'))
par(oldpar)
# you can also use the summary of mcmc.list
summary(makeThemSimilar)
```

`compareMultipleChains` *Compare multiple chains*

Description

Compares multiples chains after each one of them has been post-processed by the RSP algorithm, so that all of them are switched into a similar labelling.

Usage

```
compareMultipleChains(rspObjectList, scheme, sa_loops, maxIter, threshold)
```

Arguments

`rspObjectList`	A list consisting of `rsp` objects.
`scheme`	Character argument with possible values: "exact" (default), "partial" or "full".
`sa_loops`	Number of simulated annealing loops (only applicable when "exact" is disabled).
`maxIter`	Max number of iterations.
`threshold`	Threshold for convergence.

Value

`reorderedChains`: an object of class `mcmc.list` containing all simultaneously processed chains.

Author(s)

Panagiotis Papastamoulis

Examples

```
# load 2 chains each one consisting of a
# small mcmc sample of 100 iterations
# with p=6 variables and q=2 factors.
data(small_posterior_2chains)
Nchains <- length(small_posterior_2chains)
reorderedPosterior <- vector('list',length=Nchains)
for(i in 1:Nchains){
reorderedPosterior[[i]] <- rsp_exact( lambda_mcmc = small_posterior_2chains[[i]],
```

```
maxIter = 100,
threshold = 1e-6,
verbose=TRUE )
}
# make them comparable
makeThemSimilar <- compareMultipleChains(rspObjectList=reorderedPosterior)
```

`credible.region` *Compute a simultaneous credible region (rectangle) from a sample for a vector valued parameter.*

Description

See references below for more details. The function has been originally written for the archived `bayesSurv` package.

Usage

```
credible.region(sample, probs=c(0.90, 0.975))
```

Arguments

`sample`	a data frame or matrix with sampled values (one column = one parameter)
`probs`	probabilities for which the credible regions are to be computed

Value

A list (one component for each confidence region) of length equal to `length(probs)`. Each component of the list is a matrix with two rows (lower and upper limit) and as many columns as the number of parameters giving the confidence region.

Author(s)

Arnost Komarek

References

Besag, J., Green, P., Higdon, D. and Mengersen, K. (1995). Bayesian computation and stochastic systems (with Discussion). *Statistical Science,* **10**, 3 - 66, page 30

Held, L. (2004). Simultaneous inference in risk assessment; a Bayesian perspective *In: COMPSTAT 2004, Proceedings in Computational Statistics (J. Antoch, Ed.),* 213 - 222, page 214

Held, L. (2004b). Simultaneous posterior probability statements from Monte Carlo output. *Journal of Computational and Graphical Statistics,* **13**, 20 - 35.

Examples

```
m <- 10000
sample <- data.frame(x1=rnorm(m), x2=rnorm(m), x3=rnorm(m))
probs <- c(0.70, 0.90, 0.95)
CR <- credible.region(sample, probs=probs)

for (kk in 1:length(CR)){
 suma <- sum(sample$x1 >= CR[[kk]]["Lower", "x1"] & sample$x1 <= CR[[kk]]["Upper", "x1"] &
          sample$x2 >= CR[[kk]]["Lower", "x2"] & sample$x2 <= CR[[kk]]["Upper", "x2"] &
           sample$x3 >= CR[[kk]]["Lower", "x3"] & sample$x3 <= CR[[kk]]["Upper", "x3"])
  show <- c(suma/m, probs[kk])
  names(show) <- c("Empirical", "Desired")
  print(show)
}
```

`plot.rsp` *Plot posterior means and credible regions*

Description

This function plot posterior mean estimates per factor along with Highest Density Intervals, as well as simultaneous credible regions.

Usage

```
## S3 method for class 'rsp'
plot(x, prob, myCol, mfrow, subSet, simCR, HDI, ...)
```

Arguments

`x`	An object of class `rsp`.
`prob`	Coverage probability of credible regions.
`myCol`	Vector of colours.
`mfrow`	Number of rows and columns in the resulting graphic.
`subSet`	Enable to plot a subset of factors.
`simCR`	Logical value for plotting simultaneous credible regions. Default: True.
`HDI`	Logical value for plotting Highest Density Intervals per factor loading. Default: True.
`...`	Ignored

Value

A plot.

Author(s)

Panagiotis Papastasmoulis

Examples

```
# load small mcmc sample of 100 iterations
# with p=6 variables and q=2 factors.
data(small_posterior_2chains)
# post-process it
reorderedPosterior <- rsp_exact(
lambda_mcmc = small_posterior_2chains[[1]])
# plot it
plot(reorderedPosterior, mfrow = c(1,2), prob=0.95)
```

`procrustes_switching` *Orthogonal Procrustes rotations*

Description

Orthogonal Procrustes (OP) post-processing (Assmann et al. 2016) augmented with a final varimax rotation as implemented in Papastamoulis and Ntzoufras (2020). The algorithm uses the `procrustes` function of the MCMCpack package.

Usage

```
procrustes_switching(lambda_mcmc, maxIter, threshold, verbose, rotate, printIter)
```

Arguments

`lambda_mcmc`	Input matrix containing a MCMC sample of factor loadings. The column names should read as 'LambdaV1_1',..., 'LambdaV1_q', ..., 'LambdaVp_1',..., 'LambdaVp_q', where p and q correspond to the number of variables and factors, respectively.
`maxIter`	Maximum number of iterations of the RSP algorithm. Default: 100.
`threshold`	Positive threshold for declaring convergence. The actual convergence criterion is `threshold` m p q with m denoting the number of MCMC iterations. Default: 1e-6.
`verbose`	Logical value indicating whether to print intermediate output or not.
`rotate`	This is argument is always set to FALSE.
`printIter`	Print the progress of the algorithm when processing `printIter` MCMCdraws, per iteration. Default: 1000.

Details

If necessary, more details than the description above.

Value

`lambda_reordered_mcmc`
: Post-processed MCMC sample of factor loadings.

`lambda_hat`
: The resulting average of the post-processed MCMC sample of factor loadings.

`objective_function`
: A two-column matrix containing the time-to-reach and the value of the objective function for each iteration.

Author(s)

Panagiotis Papastamoulis

References

Assmann, C., Boysen-Hogrefem J. and Pape M. (2016). Bayesian analysis of static and dynamic factor models: An ex-post approach towards the rotation problem. *Journal of Econometrics*: 192(1): Pages 190-206.

Martin AD, Quinn KM, Park JH (2011). MCMCpack: Markov Chain Monte Carlo in R. *Journal of Statistical Software*: 42(9), 22.

Papastamoulis, P. and Ntzoufras, I. (2020). On the identifiability of Bayesian Factor Analytic models. *arXiv:2004.05105 [stat.ME]*.

Examples

```
# load small mcmc sample of 100 iterations
# with p=6 variables and q=2 factors.
data(small_posterior_2chains)
# post-process it
reorderedPosterior <- procrustes_switching(
lambda_mcmc = small_posterior_2chains[[1]])
# summarize the post-processed MCMC sample with coda
summary(reorderedPosterior$lambda_reordered_mcmc)
```

---

rsp_exact *Rotation-Sign-Permutation (RSP) algorithm (Exact scheme)*

Description

Rotation-Sign-Permutation (RSP) algorithm (exact).

Usage

```
rsp_exact(lambda_mcmc, maxIter, threshold, verbose, rotate, printIter)
```

Arguments

`lambda_mcmc`	Input matrix containing a MCMC sample of factor loadings. The column names should read as 'LambdaV1_1',..., 'LambdaV1_q', ..., 'LambdaVp_1',..., 'LambdaVp_q', where p and q correspond to the number of variables and factors, respectively.
`maxIter`	Maximum number of iterations of the RSP algorithm. Default: 100.
`threshold`	Positive threshold for declaring convergence. The actual convergence criterion is `threshold` m p q with m denoting the number of MCMC iterations. Default: 1e-6.
`verbose`	Logical value indicating whether to print intermediate output or not.
`rotate`	Logical. Default: TRUE.
`printIter`	Print the progress of the algorithm when processing `printIter` MCMCdraws, per iteration. Default: 1000.

Details

If necessary, more details than the description above.

Value

`lambda_reordered_mcmc`	Post-processed MCMC sample of factor loadings.
`sign_vectors`	The final sign-vectors.
`permute_vectors`	The final permutations.
`lambda_hat`	The resulting average of the post-processed MCMC sample of factor loadings.
`objective_function`	A two-column matrix containing the time-to-reach and the value of the objective function for each iteration.

Author(s)

Panagiotis Papastamoulis

References

Papastamoulis, P. and Ntzoufras, I. (2020). On the identifiability of Bayesian Factor Analytic models. *arXiv:2004.05105 [stat.ME]*.

Examples

```
# load small mcmc sample of 100 iterations
# with p=6 variables and q=2 factors.
data(small_posterior_2chains)
# post-process it
reorderedPosterior <- rsp_exact(
lambda_mcmc = small_posterior_2chains[[1]])
# summarize the post-processed MCMC sample with coda
summary(reorderedPosterior$lambda_reordered_mcmc)
```

`rsp_full_sa` *Rotation-Sign-Permutation (RSP) algorithm (Full Simulated Annealing)*

Description

Rotation-Sign-Permutation (RSP) algorithm (Full Simulated Annealing).

Usage

```
rsp_full_sa(lambda_mcmc, maxIter = 1000, threshold = 1e-06, verbose = TRUE,
sa_loops, rotate = TRUE, increaseIter = FALSE,
temperatureSchedule = NULL, printIter = 1000)
```

Arguments

`lambda_mcmc`	Input matrix containing a MCMC sample of factor loadings. The column names should read as 'LambdaV1_1',..., 'LambdaV1_q', ..., 'LambdaVp_1',..., 'LambdaVp_q', where p and q correspond to the number of variables and factors, respectively.
`maxIter`	Maximum number of iterations of the RSP algorithm. Default: 1000.
`threshold`	Positive threshold for declaring convergence. The actual convergence criterion is `threshold` m p q with m denoting the number of MCMC iterations. Default: 1e-6.
`verbose`	Logical value indicating whether to print intermediate output or not.
`sa_loops`	Number of simulated annealing loops per MCMC draw.
`rotate`	Logical. Default: TRUE.
`increaseIter`	Logical.
`temperatureSchedule`	Single valued function describing the temperature cooling schedule for the simulated annealing loops.
`printIter`	Print the progress of the algorithm when processing `printIter` MCMCdraws, per iteration. Default: 1000.

Details

If necessary, more details than the description above.

Value

`lambda_reordered_mcmc`	Post-processed MCMC sample of factor loadings.
`sign_vectors`	The final sign-vectors.
`permute_vectors`	The final permutations.

`lambda_hat` The resulting average of the post-processed MCMC sample of factor loadings.

`objective_function`
A two-column matrix containing the time-to-reach and the value of the objective function for each iteration.

Author(s)

Panagiotis Papastamoulis

References

Papastamoulis, P. and Ntzoufras, I. (2020). On the identifiability of Bayesian Factor Analytic models. *arXiv:2004.05105 [stat.ME]*.

Examples

```
# load small mcmc sample of 100 iterations
# with p=6 variables and q=2 factors.
data(small_posterior_2chains)
# post-process it
reorderedPosterior <- rsp_partial_sa(
lambda_mcmc = small_posterior_2chains[[1]], sa_loops=5)
# sa_loops should be larger in general
# summarize the post-processed MCMC sample with coda
summary(reorderedPosterior$lambda_reordered_mcmc)
```

---

rsp_partial_sa *Rotation-Sign-Permutation (RSP) algorithm (Partial Simulated Annealing)*

Description

Rotation-Sign-Permutation (RSP) algorithm (Partial Simulated Annealing).

Usage

```
rsp_partial_sa(lambda_mcmc, maxIter = 1000, threshold = 1e-06,
verbose = TRUE, sa_loops, rotate = TRUE, increaseIter = FALSE,
temperatureSchedule = NULL, printIter = 1000)
```

Arguments

`lambda_mcmc` Input matrix containing a MCMC sample of factor loadings. The column names should read as 'LambdaV1_1',..., 'LambdaV1_q', ..., 'LambdaVp_1',..., 'LambdaVp_q', where p and q correspond to the number of variables and factors, respectively.

`maxIter` Maximum number of iterations of the RSP algorithm. Default: 1000.

`threshold`	Positive threshold for declaring convergence. The actual convergence criterion is `threshold` m p q with m denoting the number of MCMC iterations. Default: 1e-6.
`verbose`	Logical value indicating whether to print intermediate output or not.
`sa_loops`	Number of simulated annealing loops per MCMC draw.
`rotate`	Logical. Default: TRUE.
`increaseIter`	Logical.
`temperatureSchedule`	Single valued function describing the temperature cooling schedule for the simulated annealing loops.
`printIter`	Print the progress of the algorithm when processing `printIter` MCMCdraws, per iteration. Default: 1000.

Details

If necessary, more details than the description above.

Value

`lambda_reordered_mcmc`	Post-processed MCMC sample of factor loadings.
`sign_vectors`	The final sign-vectors.
`permute_vectors`	The final permutations.
`lambda_hat`	The resulting average of the post-processed MCMC sample of factor loadings.
`objective_function`	A two-column matrix containing the time-to-reach and the value of the objective function for each iteration.

Author(s)

Panagiotis Papastamoulis

References

Papastamoulis, P. and Ntzoufras, I. (2020). On the identifiability of Bayesian Factor Analytic models. *arXiv:2004.05105 [stat.ME]*.

Examples

```
# load small mcmc sample of 100 iterations
# with p=6 variables and q=2 factors.
data(small_posterior_2chains)
# post-process it
reorderedPosterior <- rsp_partial_sa(
lambda_mcmc = small_posterior_2chains[[1]],
sa_loops=5)
# sa_loops should be larger in general
```

```
# summarize the post-processed MCMC sample with coda
summary(reorderedPosterior$lambda_reordered_mcmc)
```

`small_posterior_2chains` *Example data*

Description

A list consisting of two small MCMC chains.

Usage

```
data(small_posterior_2chains)
```

Format

List of length 2. Each entry contains a matrix of 20 MCMC draws.

`switch_and_permute` *Apply sign switchings and column permutations*

Description

Help function, not really meant to be used by the average user.

Usage

```
switch_and_permute(lambda_mcmc, switch_vectors, permute_vectors)
```

Arguments

`lambda_mcmc`	MCMC input.
`switch_vectors`	Sign vectors.
`permute_vectors`	Permutation vectors.

Value

reordered `lambda_mcmc` according to sign and permutations provided.

Author(s)

Panagiotis Papastamoulis

`weighted_procrustes_switching` *Weighted Orthogonal Procrustes rotations*

Description

Weighted Orthogonal Procrustes (WOP) post-processing (Assmann et al. 2016) augmented with a final varimax rotation as implemented in Papastamoulis and Ntzoufras (2020). The algorithm uses the `procrustes` function of the MCMCpack package.

Usage

```
weighted_procrustes_switching(lambda_mcmc, maxIter, threshold, verbose,
    weight, printIter)
```

Arguments

`lambda_mcmc`	Input matrix containing a MCMC sample of factor loadings. The column names should read as 'LambdaV1_1',..., 'LambdaV1_q', ..., 'LambdaVp_1',..., 'LambdaVp_q', where p and q correspond to the number of variables and factors, respectively.
`maxIter`	Maximum number of iterations of the RSP algorithm. Default: 100.
`threshold`	Positive threshold for declaring convergence. The actual convergence criterion is `threshold` m p q with m denoting the number of MCMC iterations. Default: 1e-6.
`verbose`	Logical value indicating whether to print intermediate output or not.
`weight`	This is argument is always set to TRUE.
`printIter`	Print the progress of the algorithm when processing `printIter` MCMCdraws, per iteration. Default: 1000.

Details

If necessary, more details than the description above.

Value

`lambda_reordered_mcmc`	Post-processed MCMC sample of factor loadings.
`lambda_hat`	The resulting average of the post-processed MCMC sample of factor loadings.
`objective_function`	A two-column matrix containing the time-to-reach and the value of the objective function for each iteration.

Author(s)

Panagiotis Papastamoulis

References

Assmann, C., Boysen-Hogrefem J. and Pape M. (2016). Bayesian analysis of static and dynamic factor models: An ex-post approach towards the rotation problem. *Journal of Econometrics*: 192 (1): Pages 190-206.

Martin AD, Quinn KM, Park JH (2011). MCMCpack: Markov Chain Monte Carlo in R. *Journal of Statistical Software*: 42(9), 22.

Papastamoulis, P. and Ntzoufras, I. (2020). On the identifiability of Bayesian Factor Analytic models. *arXiv:2004.05105 [stat.ME]*.

Examples

```
# load small mcmc sample of 100 iterations
# with p=6 variables and q=2 factors.
data(small_posterior_2chains)
# post-process it
reorderedPosterior <- weighted_procrustes_switching(
lambda_mcmc = small_posterior_2chains[[1]])
# summarize the post-processed MCMC sample with coda
summary(reorderedPosterior$lambda_reordered_mcmc)
```

Index