

# Package ‘bfa’

September 29, 2016

**Maintainer** Jared Murray <jsmurray@stat.cmu.edu>

**License** GPL-3

**Title** Bayesian Factor Analysis

**LinkingTo** Rcpp, RcppArmadillo

**Type** Package

**LazyLoad** yes

**Author** Jared Murray

**Description** Provides model fitting for several Bayesian factor models including Gaussian, ordinal probit, mixed and semiparametric Gaussian copula factor models under a range of priors.

**SystemRequirements** GNU make

**Version** 0.4

**Date** 2016-09-07

**Imports** coda, Rcpp (>= 0.10.6)

**Collate** 'bfa-package.R' 'main.R' 'bfa\_copula.R' 'bfa\_gauss.R' 'bfa\_mixed.R' 'rng.R' 'postproc.R' 'class.R' 'get\_coda.R' 'plots.R' 'ratings08.R' 'wrappers.R'

**RoxygenNote** 5.0.1

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2016-09-29 08:04:45

## R topics documented:

bfa-package	2
bfa_copula	2
bfa_gauss	5
bfa_mixed	6
bfa_model	8

biplot.bfa . . . . .	9
coef.bfa . . . . .	9
cor_samp . . . . .	10
cov_samp . . . . .	10
get_coda . . . . .	11
get_posterior_loadings . . . . .	11
get_posterior_scores . . . . .	12
HPDinterval.bfa . . . . .	12
mean.bfa . . . . .	13
predict.bfa . . . . .	13
print.bfa . . . . .	14
ratings08 . . . . .	14
utri_restrict . . . . .	15
var.bfa . . . . .	16
woodbury . . . . .	16

<b>Index</b>	<b>17</b>
--------------	-----------

---

bfa-package	<i>Bayesian Factor Analysis</i>
-------------	---------------------------------

---

## Description

Bayesian Factor Analysis

## Details

Package:	bfa
Type:	Package
Version:	0.4
Date:	2016-9-07
License:	GPL-3
LazyLoad:	yes

This package provides model fitting for several Bayesian factor models including Gaussian, ordinal probit, mixed and semiparametric Gaussian copula factor models under a range of priors.

## Author(s)

Jared Murray <jsmurray@stat.cmu.edu>

---

bfa_copula	<i>Initialize and fit a Gaussian copula factor model</i>
------------	--

---

## Description

This function performs a specified number of MCMC iterations and returns an object containing summary statistics from the MCMC samples as well as the actual samples of factor scores if `keep.scores` is `TRUE`. Default behavior is to save only samples of the loadings.

## Usage

```
bfa_copula(x, data = NULL, num.factor = 1, restrict = NA,
  normal.dist = NA, nsim = 10000, nburn = 1000, thin = 1,
  print.status = 500, keep.scores = FALSE, loading.prior = c("gdp",
  "pointmass", "normal"), factor.scales = FALSE, px = TRUE,
  coda = "loadings", coda.scale = TRUE, imh = FALSE, imh.iter = 500,
  imh.burn = 500, ...)
```

## Arguments

<code>x</code>	A formula or bfa object.
<code>data</code>	The data if <code>x</code> is a formula
<code>num.factor</code>	Number of factors
<code>restrict</code>	A matrix or list giving restrictions on factor loadings. A matrix should be the same size as the loadings matrix. Acceptable values are 0 (identically 0), 1 (unrestricted), or 2 (strictly positive). List elements should be character vectors of the form <code>c('variable', 1, "&gt;0")</code> where 'variable' is the manifest variable, 1 is the factor, and ">0" is the restriction. Acceptable restrictions are ">0" or "0".
<code>normal.dist</code>	A character vector specifying which variables should be treated as observed Gaussian. Defaults to none (a completely semiparametric copula model)
<code>nsim</code>	Number of iterations past burn-in
<code>nburn</code>	Number of initial (burn-in) iterations to discard
<code>thin</code>	Keep every thin'th MCMC sample (i.e. save <code>nsim/thin</code> samples)
<code>print.status</code>	How often to print status messages to console
<code>keep.scores</code>	Save samples of factor scores
<code>loading.prior</code>	Specify GDP ("gdp", default) point mass ("pointmass") or normal priors ("normal")
<code>factor.scales</code>	Include a separate scale parameter for each factor
<code>px</code>	Use parameter expansion (strongly recommended!)
<code>coda</code>	Create mcmc objects to allow use of functions from the coda package: "all" for loadings and scores, "loadings" or "scores" for one or the other, or "none" for neither
<code>coda.scale</code>	Put the loadings on the correlation scale when creating mcmc objects
<code>imh</code>	Use Independence Metropolis-Hastings step for discrete margins. If <code>FALSE</code> , use the semiparametric extended rank likelihood. If <code>TRUE</code> , uses a uniform prior on the cutpoints
<code>imh.iter</code>	Iterations used to build IMH proposal

imh.burn	Burn-in before collecting samples used to build IMH proposal (total burn-in is nburn+imh.iter+imh.burn)
...	Prior parameters and other (experimental) arguments (see details)

## Details

Additional parameters:

- loadings.var: Factor loading prior variance
- tau.a, tau.b: Gamma hyperparameters (scale=1/b) for factor precisions (if factor.scales=T). Default is a=b=1 (MV t w/ df=2)
- rho.a, rho.b: Beta hyperparameters for point mass prior
- gdp.alpha, gdp.beta: GDP prior parameters

## Value

A bfa object with posterior samples.

## Examples

```
## Not run:
require(MASS)
data(UScereal)
UScereal$shelf = factor(UScereal$shelf, ordered=TRUE)
UScereal$vitamins = factor(UScereal$vitamins, ordered=TRUE,
  levels=c("none", "enriched", "100%"))
obj = bfa_copula(~., data=UScereal[,-1], num.factor=2, nsim=10000, nburn=1000, thin=4,
  rest=list(c("sugars", 2, "0")), loading.prior="gdp", keep.scores=T,
  print.status=2500)

biplot(obj, cex=c(0.8, 0.8))
plot(get_coda(obj))
plot(get_coda(obj, loadings=F, scores=T))
hpd.int = HPDinterval(obj, scores=T)

#sample from posterior predictive
ps = predict(obj)

m=ggplot(UScereal, aes(x=calories, y=sugars))+geom_point(position='jitter', alpha=0.5)
m=m+stat_density2d(data=ps, aes(x=calories, y=sugars, color = ..level..), geom='contour')
print(m)

m=ggplot(UScereal, aes(x=calories))+geom_histogram()
m=m+stat_density(data=ps, aes(x=calories, y=..count..), color='red',fill=NA, adjust=1.3)
m=m+facet_grid(shelf~.)
print(m)

#we can compute conditional dist'n estimates directly as well by supplying cond.vars
cond.vars=list(shelf=1)
out = predict(obj, resp.var="calories", cond.vars=cond.vars)
plot(sort(unique(UScereal$calories)), apply(out, 2,mean), type='s')
lines(sort(unique(UScereal$calories)), apply(out, 2, quantile, 0.05), type='s', lty=2)
```

```

lines(sort(unique(UScereal$calories)), apply(out, 2,quantile, 0.95), type='s', lty=2)
lines(ecdf(UScereal$calories[UScereal$shelf==1]), col='blue')
text(400, 0.1, paste("n =", sum(UScereal$shelf==1)))

out2 = predict(obj, resp.var="calories", cond.vars=list(shelf=2))
out3 = predict(obj, resp.var="calories", cond.vars=list(shelf=3))
plot(sort(unique(UScereal$calories)), apply(out, 2,mean), type='s')
lines(sort(unique(UScereal$calories)), apply(out2, 2,mean), type='s', lty=2)
lines(sort(unique(UScereal$calories)), apply(out3, 2,mean), type='s', lty=3)

## End(Not run)

```

bfa\_gauss

*Initialize and fit a Gaussian factor model***Description**

This function performs a specified number of MCMC iterations and returns an object containing summary statistics from the MCMC samples as well as the actual samples of factor scores if `keep.scores` is TRUE. Default behavior is to save only samples of the loadings.

**Usage**

```

bfa_gauss(x, data = NULL, num.factor = 1, restrict = NA, nsim = 10000,
  nburn = 1000, thin = 1, print.status = 500, keep.scores = FALSE,
  loading.prior = c("gdp", "pointmass", "normal"), factor.scales = TRUE,
  coda = "loadings", ...)

```

**Arguments**

<code>x</code>	A formula or bfa object.
<code>data</code>	The data (if <code>x</code> is a formula)
<code>num.factor</code>	Number of factors
<code>restrict</code>	A matrix or list giving identifiability restrictions on factor loadings. A matrix should be the same size as the loadings matrix. Acceptable values are 0 (identically 0), 1 (unrestricted), or 2 (strictly positive). List elements should be character vectors of the form <code>c("variable",1, "&gt;0")</code> where 'variable' is the manifest variable, 1 is the factor, and ">0" is the restriction. Acceptable restrictions are ">0" or "0".
<code>nsim</code>	Number of iterations past burn-in
<code>nburn</code>	Number of initial (burn-in) iterations to discard
<code>thin</code>	Keep every thin'th MCMC sample (i.e. save <code>nsim/thin</code> samples)
<code>print.status</code>	How often to print status messages to console
<code>keep.scores</code>	Save samples of factor scores

loading.prior	Specify the prior on factor loadings - generalized double Pareto ("gdp", default), point mass mixtures (mixture of point mass at zero + mean zero normal) ("point-mass") or normal/Gaussian ("normal")
factor.scales	Include a shared precision parameter for each column of the factor loadings matrix. See details for setting hyperprior parameters. This is implemented as in PX-FA of Ghosh and Dunson (2009)
coda	Create mcmc objects to allow use of functions from the coda package: "all" for loadings and scores, "loadings" or "scores" for one or the other, or "none" for neither
...	Prior parameters and other (experimental) arguments (see details)

### Details

Note: All the priors in use assume that the manifest variables are on approximately the same scale.

Additional parameters:

- loadings.var: Factor loading prior variance
- tau.a, tau.b: Gamma hyperparameters (scale=1/b) for factor precisions (if factor.scales=T). Default is a=b=1 (MV t w/ df=2)
- rho.a, rho.b: Beta hyperparameters for point mass prior
- sigma2.a, sigma2.b: Gamma hyperparameters for error precisions
- gdp.alpha, gdp.beta: GDP prior parameters
- mu.mean, mu.var: (Scalar) prior mean and variance for  $\mu[j]$  (where  $E(y) = \mu$ ). Defaults are 0 and 1e4.

### Value

A bfa object with posterior samples.

---

bfa_mixed	<i>Initialize and fit a mixed-scale Gaussian factor model, with probit specifications for the discrete margins</i>
-----------	--

---

### Description

This function performs a specified number of MCMC iterations and returns an object containing summary statistics from the MCMC samples as well as the actual samples of factor scores if keep.scores is TRUE. Default behavior is to save only samples of the loadings.

### Usage

```
bfa_mixed(x, data = NULL, num.factor = 1, restrict = NA,
  normal.dist = NA, nsim = 10000, nburn = 1000, thin = 1,
  print.status = 500, keep.scores = FALSE, loading.prior = c("gdp",
  "pointmass", "normal"), factor.scales = FALSE, px = TRUE,
  coda = "loadings", coda.scale = TRUE, imh.iter = 500, imh.burn = 500,
  ...)
```

**Arguments**

x	A formula or bfa object.
data	The data if x is a formula
num.factor	Number of factors
restrict	A matrix or list giving restrictions on factor loadings. A matrix should be the same size as the loadings matrix. Acceptable values are 0 (identically 0), 1 (unrestricted), or 2 (strictly positive). List elements should be character vectors of the form <code>c('variable', 1, "&gt;0")</code> where 'variable' is the manifest variable, 1 is the factor, and ">0" is the restriction. Acceptable restrictions are ">0" or "0".
normal.dist	A character vector specifying which variables should be treated as observed Gaussian. Defaults to all numeric variables if x is a formula.
nsim	Number of iterations past burn-in
nburn	Number of initial (burn-in) iterations to discard
thin	Keep every thin'th MCMC sample (i.e. save nsim/thin samples)
print.status	How often to print status messages to console
keep.scores	Save samples of factor scores
loading.prior	Specify GDP ("gdp", default) point mass ("pointmass") or normal priors ("normal")
factor.scales	Include a separate scale parameter for each factor
px	Use parameter expansion for ordinal variables (recommended)
coda	Create coda objects to allow use of functions from the coda package: "all" for loadings and scores, "loadings" or "scores" for one or the other, or "none" for neither
coda.scale	Put the loadings on the correlation scale when creating coda objects
imh.iter	Iterations used to build IMH proposal
imh.burn	Burn-in before collecting samples used to build IMH proposal (total burn-in is nburn+imh.iter+imh.burn)
...	Prior parameters and other (experimental) arguments (see details)

**Details**

Note: All the priors in use assume that the manifest Gaussian variables are on approximately the same scale.

Additional parameters:

- `loadings.var`: Factor loading prior variance
- `tau.a`, `tau.b`: Gamma hyperparameters (scale=1/b) for factor precisions (if `factor.scales=T`). Default is `a=b=1` (MV t w/ `df=2`)
- `rho.a`, `rho.b`: Beta hyperparameters for point mass prior
- `sigma2.a`, `sigma2.b`: Gamma hyperparameters for error precisions (for numeric variables)
- `gdp.alpha`, `gdp.beta`: GDP prior parameters
- `mu.mean`, `mu.var`: (Scalar) prior mean and variance for  $\mu[j]$  (where  $E(y) = \mu$ ). Defaults are 0 and  $1e4$ .

**Value**

A bfa object with posterior samples.

---

bfa_model	<i>Initialize a bfa model</i>
-----------	-------------------------------

---

**Description**

This function accepts a data matrix  $D$  and specified options, returning an S3 object of class bfa.

**Usage**

```
bfa_model(x, data = NULL, num.factor = 1, restrict = NA,
  normal.dist = NA, center.data = NULL, scale.data = NULL, init = TRUE,
  ...)
```

**Arguments**

x	A formula or matrix
data	The data if x is a formula
num.factor	number of factors
restrict	A matrix or list giving restrictions on factor loadings. A matrix should be the same size as the loadings matrix. Acceptable values are 0 (identically 0), 1 (unrestricted), or 2 (strictly positive). List elements should be character vectors of the form <code>c('variable', 1, '&gt;0')</code> where 'variable' is the manifest variable, 1 is the factor, and '>0' is the restriction. Acceptable restrictions are '>0' or '0'.
normal.dist	A character vector specifying which variables should be treated as observed Gaussian
center.data	Center each margin (deprecated, ignored!)
scale.data	Scale each margin to unit mean/variance (deprecated, ignored!)
init	Initialize the factor loadings
...	ignored

**Value**

An S3 object of class bfa.



---

biplot.bfa	<i>Display a biplot</i>
------------	-------------------------

---

**Description**

Display a biplot

**Usage**

```
## S3 method for class 'bfa'
biplot(x, factors = c(1, 2), ...)
```

**Arguments**

x	A bfa object
factors	Integer vector giving indices of the factors to plot
...	Additional arguments to biplot; see ?biplot

**Value**

Displays a biplot

---

coef.bfa	<i>Extract samples of implied regression coefficients from a bfa object</i>
----------	---

---

**Description**

Extract samples of implied regression coefficients from a bfa object

**Usage**

```
## S3 method for class 'bfa'
coef(object, responses, scale = attr(object, "type") != "gauss",
...)
```

**Arguments**

object	A bfa object
responses	A character vector containing one or more response variables
scale	Whether to compute regression coefficients from factor loadings on the correlation scale; recommended if object is a copula or mixed factor model.
...	Ignored

**Value**

An array of dimension  $\text{length}(\text{index}) \times \text{p-length}(\text{index}) \times (\text{no. of mcmc samples})$  with posterior samples of regression coefficients

---

`cor_samp`*Compute samples of the correlation matrix*

---

**Description**

Compute samples of the correlation matrix

**Usage**

```
cor_samp(model)
```

**Arguments**

`model`            bfa model object

**Value**

A  $p \times p \times$  (no. of mcmc samples) array containing samples of the correlation matrix

---

`cov_samp`*Compute samples of the covariance matrix*

---

**Description**

Compute samples of the covariance matrix

**Usage**

```
cov_samp(model)
```

**Arguments**

`model`            bfa model object

**Value**

A  $p \times p \times$  (no. of mcmc samples) array containing samples of the covariance matrix

---

get_coda	<i>Get coda object</i>
----------	------------------------

---

**Description**

Returns an mcmc object for use in coda functions for convergence diagnostics, HPD intervals, etc.

**Usage**

```
get_coda(model, loadings = TRUE, scores = FALSE, scale = attr(model,
  "type") != "gauss", positive = FALSE)
```

**Arguments**

model	a bfa model
loadings	Return samples of the factor loadings? (default is TRUE)
scores	Return samples of the factor scores? (default is FALSE)
scale	Return factor loadings on the correlation scale? (default is TRUE for mixed/copula models, FALSE for Gaussian models)
positive	Post-process to enforce positivity constraints

**Value**

An mcmc object

---

get_posterior_loadings	<i>Get posterior samples of factor loadings</i>
------------------------	---

---

**Description**

Returns an array of posterior samples of factor loadings

**Usage**

```
get_posterior_loadings(model, scale = attr(model, "type") != "gauss")
```

**Arguments**

model	a bfa model
scale	Return factor loadings on the correlation scale? (default is TRUE for mixed/copula models, FALSE for Gaussian models)

**Value**

A  $p \times k \times n$  (no. of mcmc samples) array containing samples of the  $p \times k$  loadings matrix

---

`get_posterior_scores` *Get posterior samples of factor scores*

---

**Description**

Returns an array of posterior samples of factor scores

**Usage**

```
get_posterior_scores(model)
```

**Arguments**

`model`            a bfa model

**Value**

An mcmc object

---

`HPDinterval.bfa`        *HPD intervals from a bfa object*

---

**Description**

HPD intervals from a bfa object

**Usage**

```
## S3 method for class 'bfa'
HPDinterval(obj, prob = 0.95, loadings = TRUE,
             scores = FALSE, ...)
```

**Arguments**

`obj`                A bfa object  
`prob`               Target probability content (see ?HPDinterval)  
`loadings`           Compute interval for the loadings  
`scores`             Compute interval for the scores  
`...`               Ignored

**Value**

A list with elements `loadings.lower`, `loadings.upper`, `scores.lower`, `scores.upper` which are matrices of dimension  $p \times k$  or  $n \times k$ , or NA's if `loadings` or `scores` is FALSE

---

mean.bfa	<i>Extract posterior means from bfa object</i>
----------	--

---

**Description**

Extract posterior means from bfa object

**Usage**

```
## S3 method for class 'bfa'
mean(x, scale = attr(x, "type") != "gauss", ...)
```

**Arguments**

x	A bfa object
scale	Return factor loadings on the correlation scale? (default is TRUE for mixed/copula models, FALSE for Gaussian models)
...	Ignored

**Value**

A list with elements loadings and scores containing MCMC sample means

---

predict.bfa	<i>Posterior predictive and univariate conditional posterior predictive distributions</i>
-------------	---

---

**Description**

Posterior predictive and univariate conditional posterior predictive distributions, currently implemented only for Gaussian copula models. If resp.var is not NA, returns an estimate of the conditional cdf at every observed data point for each MCMC iterate. If resp.var is NA, returns draws from the joint posterior predictive.

**Usage**

```
## S3 method for class 'bfa'
predict(object, resp.var = NA, cond.vars = NA,
        numeric.as.factor = TRUE, ...)
```

**Arguments**

object	bfa model object
resp.var	Either a character vector (length 1) with name of the response variable for conditional, or NA for draws from the joint posterior predictive.
cond.vars	Conditioning variables; either a list like list(X1=val1, X2=val2) with X1, X2 variables in the original data frame, or a P length vector with either the conditioning value or NA (for marginalized variables). Ignored if resp.var is NA
numeric.as.factor	Treat numeric variables as ordinal when conditioning
...	Ignored

**Value**

A matrix where each row is either a sample of the conditional posterior predictive cdf at each datapoint, or a single sample from the joint posterior predictive.

---

print.bfa	<i>Print method for bfa object</i>
-----------	------------------------------------

---

**Description**

Print method for bfa object

**Usage**

```
## S3 method for class 'bfa'
print(x, ...)
```

**Arguments**

x	A bfa object
...	Ignored

---

ratings08	<i>Special Interest Group Ratings (2008)</i>
-----------	--

---

**Description**

This dataset consists of special interest group ratings of members of the US House and Senate. It contains two objects: ratings, a data frame with the ratings information, and groups which contains information about the special interest groups

**Usage**

```
ratings08
```

**Format**

List of two data frames: ratings and groups (see above)

**Author(s)**

Jared Murray <jared.murray@stat.duke.edu>

**Source**

Accessed through the Project VoteSmart API: <http://www.votesmart.org>

---

utri_restrict	<i>Upper triangular loadings restriction</i>
---------------	--

---

**Description**

Returns the restriction matrix corresponding to Geweke & Zhou (1996) upper-triangular positive diagonal condition '

**Usage**

```
utri_restrict(p, k)
```

**Arguments**

p	number of rows (variables)
k	number of columns (factors)

**Value**

p by k matrix

---

var.bfa	<i>Extract posterior variances from bfa object</i>
---------	--

---

**Description**

Extract posterior variances from bfa object

**Usage**

```
var.bfa(x, scale = attr(x, "type") != "gauss", ...)
```

**Arguments**

x	A bfa object
scale	Return factor loadings on the correlation scale? (default is TRUE for mixed/copula models, FALSE for Gaussian models)
...	Ignored

**Value**

A list with elements loadings and scores containing MCMC sample variances

---

woodbury	<i>Invert the matrix <math>\text{lam} * \text{t}(\text{lam}) + \text{diag}(u)</math> via Woodbury identity</i>
----------	--

---

**Description**

Invert the matrix  $\text{lam} * \text{t}(\text{lam}) + \text{diag}(u)$  via Woodbury identity

**Usage**

```
woodbury(lam, u)
```

**Arguments**

lam	p by k matrix
u	p dimensional vector

**Value**

The matrix inverse



# Index

## \*Topic **package**

bfa-package, 2

bfa-package, 2

bfa\_copula, 2

bfa\_gauss, 5

bfa\_mixed, 6

bfa\_model, 8

biplot.bfa, 9

coef.bfa, 9

cor\_samp, 10

cov\_samp, 10

get\_coda, 11

get\_posterior\_loadings, 11

get\_posterior\_scores, 12

HPDinterval.bfa, 12

mean.bfa, 13

predict.bfa, 13

print.bfa, 14

ratings08, 14

utri\_restrict, 15

var.bfa, 16

woodbury, 16