Package 'bfa'

February 19, 2015

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

License GPL-3

Title Bayesian Factor Analysis

LinkingTo Rcpp, RcppArmadillo

Type Package

LazyLoad yes

Author Jared Murray

Description 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.

SystemRequirements GNU make

Version 0.3.1

Date 2014-2-10

Imports coda, Rcpp (>= 0.10.6), RcppArmadillo (>= 0.2.35)

Collate 'bfa-package.R' 'main.R' 'bfa_copula.R' 'bfa_gauss.R' 'bfa_mixed.R' 'class.R' 'get_coda.R' 'plots.R' 'rng.R' 'postproc.R' 'ratings08.R' 'wrappers.R'

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-02-11 15:30:23

R topics documented:

`bfa-package` *Bayesian Factor Analysis*

Description

Bayesian Factor Analysis

Details

Package:	bfa
Type:	Package
Version:	0.3.1
Date:	2014-2-10
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 <jared.murray@stat.duke.edu>

`bfa_copula` *Initialize and fit a Gaussian copula factor model*

Description

Perform MCMC model fitting for a semiparametric Gaussian copula factor model

Usage

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

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 c('variable',1, ">0") 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 none (a completely semiparametric copula model)
center.data	Center continuous variables (only if normal.dist includes one or more variables)
scale.data	Scale continuous variables (only if normal.dist includes one or more variables)
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 (strongly recommended!)
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
coda.scale	Put the loadings on the correlation scale when creating mcmc objects
imh	Use Independence Metropolis-Hastings step for discrete margins. If FALSE, use the semiparametric extended rank likelihood. If TRUE, uses a uniform prior on the cutpoints
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

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)
plot_loadings(obj)
#plot_loadings returns ggplot objects you can tweak
m = plot_loadings(obj, type='credint')
print(m)
print(m+opts(title="HPD intervals (p=0.95)")+theme_bw())
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$calories)), apply(out, 2,quantile, 0.95), type='s', lt=2)
lines(ecdf(UScereal$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

Performs MCMC model fitting for a Gaussian factor model.

Usage

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

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 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 c("variable",1, ">0") where 'variable' is the manifest variable, 1 is the factor, and ">0" is the restriction. Acceptable restrictions are ">0" or "0".
center.data	Center data (recommended; this model doesn't include a mean vector!)
scale.data	Scale data (also recommended)
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 the prior on factor loadings - generalized double Pareto ("gdp", default), point mass mixtures (mixture of point mass at zero + mean zero normal) ("pointmass") 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

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

Value

A bfa object with posterior samples.

---

bfa_mixed *Initialize and fit a mixed-scale Gaussian factor model, with probit specifications for the discrete margins*

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 if keep.scores or keep.loadings are TRUE. Default behavior is to save only the loadings.

Usage

```
bfa_mixed(x, data = NULL, num.factor = 1, restrict = NA,
  normal.dist = NA, center.data = TRUE, scale.data = TRUE, 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 c('variable',1, ">0") 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.
center.data	Center data (for continuous variables - recommended!)
scale.data	Scale data (for continuous variables)
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 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
coda.scale	Put the loadings on the correlation scale when creating mcmc 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

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

Value

A bfa object with posterior samples.

---

bfa_model *Initialize a bfa model*

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 = TRUE, scale.data = FALSE, 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 c('variable',1, '>0') 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
scale.data	Scale each margin to unit mean/variance
init	Initialize the factor loadings
...	ignored

Value

An S3 object of class bfa.

biplot.bfa *Display a biplot*

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

Shows a biplot

coef.bfa *Extract samples of implied regression coefficients from a bfa object*

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 length(index) x p-length(index) x (no. of mcmc samples) with posterior samples of regression coefficients

corr_samp *Compute samples of the correlation matrix*

Description

Compute samples of the correlation matrix

Usage

```
corr_samp(model)
```

Arguments

model	bfa model object

Value

A p x p x (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 x p x (no. of mcmc samples) array containing samples of the covariance matrix

get_coda *Get coda object*

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 = TRUE,
  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)
positive	Post-process to enforce positivity constraints

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 x k or n x k, or NA's if loadings or scores is FALSE

`mean.bfa` *Extract posterior means from bfa object*

Description

Extract posterior means from bfa object

Usage

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

Arguments

`x`	A bfa object
`...`	Ignored

Value

A list with elements loadings and scores containing MCMC sample means

---

`predict.bfa` *Posterior predictive and univariate conditional posterior predictive distributions*

Description

Posterior predictive and univariate conditional posterior predictive distributions, currently implemented only for Gaussian and 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 *Print method for bfa object*

Description

Print method for bfa object

Usage

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

Arguments

x	A bfa object
...	Ignored

---

ratings08 *Special Interest Group Ratings (2008)*

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.projectvotesmart.org

---

`utri_restrict` *Upper triangular loadings restriction*

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 *Extract posterior means from bfa object*

Description

Extract posterior means from bfa object

Usage

```
var.bfa(x, ...)
```

Arguments

x	A bfa object
...	Ignored

Value

A list with elements loadings and scores containing MCMC sample variances

woodbury *Invert the matrix lam*t(lam)+diag(u) via Woodbury identity*

Description

Invert the matrix lam*t(lam)+diag(u) via Woodbury identity

Usage

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

Arguments

lam	p by k matrix
u	p dimensional vector

Value

The matrix inverse

Index