Package: spifa (via r-universe)

October 22, 2026

Type Package

Title Bayesian Spatial Item Factor Analysis

Version 0.1.0

Encoding UTF-8

Description Fits spatial item factor analysis models for binary responses, following Chacón-Montalván, E. A., Parry, L., Giorgi, E., Torres, P., Orellana, J. D. Y., Moraga, P., and Taylor, B. M. (2025) <doi:10.1214/25-AOAS2072>, using full Bayesian inference via adaptive Metropolis-Hastings within Gibbs sampling and auxiliary variables with a probit link function. The latent factors are modelled as the sum of a predictor effect, a multivariate Gaussian process capturing spatial dependence, and a multivariate non-spatial term, allowing users to fit and predict spatially-referenced constructs such as food insecurity or socioeconomic indices. Standard exploratory and confirmatory item factor analysis are supported as particular cases, simply by dropping the spatial structure.

License GPL-3

URL https://github.com/ErickChacon/spifa, https://ErickChacon.github.io/spifa/

Depends R (>= 4.1.0)

BugReports https://github.com/ErickChacon/spifa/issues

Imports Rcpp (>= 0.12.13), RcppTN, tibble, dplyr, tidyr, sf, ggplot2, posterior, ggridges, patchwork

LinkingTo Rcpp, RcppArmadillo, RcppTN

Suggests knitr, rmarkdown, testthat (>= 3.0.0), mirt, gstat

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author Erick A. Chacón-Montalván [aut, cre] (ORCID: <https://orcid.org/0000-0001-8068-1034>), Emanuele Giorgi [ctb], Benjamin M. Taylor [ctb]

Maintainer Erick A. Chacón-Montalván <erick.chaconmontalvan@wur.nl>

Repository https://cran.r-universe.dev

Date/Publication 2026-10-22 10:30:02 UTC

RemoteUrl https://github.com/cran/spifa

RemoteRef HEAD

RemoteSha 68e38b172e47c2020d5c8524644c86d6681ba22d

Contents

spifa-package 2
dic 3
dic.spifa 4
ipixuna 5
plot.spifa 6
plot_density 7
plot_interval 8
plot_predict 10
plot_trace 11
predict.spifa 13
print.spifa 15
spifa 16
summary.spifa 20
update.spifa 21

Index **23**

`spifa-package` *spifa: Bayesian Spatial Item Factor Analysis*

Description

Fits item factor analysis models for binary responses using full Bayesian inference via Gibbs sampling and auxiliary variables with a probit link function, optionally extending the latent factors with a multivariate Gaussian process to capture spatial dependence. See `spifa` for the main model-fitting function.

Author(s)

Maintainer: Erick A. Chacón-Montalván <erick.chaconmontalvan@wur.nl> (ORCID)

Authors:

- Erick A. Chacón-Montalván <erick.chaconmontalvan@wur.nl> (ORCID)

Other contributors:

- Emanuele Giorgi [contributor]
- Benjamin M. Taylor [contributor]

See Also

Useful links:

- https://github.com/ErickChacon/spifa
- https://ErickChacon.github.io/spifa/
- Report bugs at https://github.com/ErickChacon/spifa/issues

---

dic — *Deviance Information Criterion*

Description

Generic function for the Deviance Information Criterion (DIC), a Bayesian measure of model fit that penalises complexity. See `dic.spifa` for the `spifa` method.

Usage

```
dic(x, ...)
```

Arguments

`x`	A fitted model object.
`...`	Further arguments passed to methods.

Value

The DIC for `x`; see the specific method (e.g. `dic.spifa`) for its exact return value.

`dic.spifa` *Deviance Information Criterion for a spifa Model*

Description

Computes the Deviance Information Criterion (DIC) for a fitted `spifa` model, useful for comparing candidate models (e.g. different numbers of factors or different restriction structures) fitted to the same data.

Usage

```
## S3 method for class 'spifa'
dic(x, burnin = 0, thin = 1, ...)
```

Arguments

`x`	A fitted `spifa` object, as returned by `spifa`.
`burnin`	Number of initial iterations to discard.
`thin`	Thinning interval applied after discarding burn-in.
`...`	Further arguments passed to methods (currently unused).

Value

A one-row `tibble` with columns `mean_deviance` (posterior mean of the deviance), `p_eff` (effective number of parameters), and `dic`.

Author(s)

Erick A. Chacón-Montalván

Examples

```
data(ipixuna)
nitems <- ncol(ipixuna$items)
nfactors <- 3

# discrimination constraint: start with every item free to load on every
# factor, then restrict a few items per factor based on what each item is
# meant to measure (0 = no relationship, 1 = free parameter to estimate)
A <- matrix(1, nitems, nfactors)
A[c(4, 8), 1] <- 0
A[c(2, 4, 5, 6, 7, 8, 10), 2] <- 0
A[c(5, 6), 3] <- 0
samples <- spifa(
  items ~ 1, data = ipixuna, nfactors = nfactors, ngp = 0,
  niter = 20, standardize = FALSE,
  constraints = list(discrimination = A))
dic(samples)
```

`ipixuna` *Simulated Food Insecurity Survey Data for Ipixuna*

Description

Simulated household survey data for the rural area of Ipixuna (Para state, Brazil), used to illustrate spatial item factor analysis. Ten binary items, a household-level ‘wealth‘ covariate, and spatial coordinates are simulated for each household so that they reproduce the spatial multidimensional item response structure described in the SPIFA paper (Chacón-Montalván, E. A., Giorgi, E. and Taylor, B.). The data is provided as an ‘sf‘ object, already in the format [spifa()] requires.

Format

An ‘sf‘ tibble with one row per household (100 rows) and columns:

id Household identifier.

wealth A simulated household-level covariate.

items A 100x10 binary (0/1) matrix of responses to the food insecurity items.

geometry An ‘sfc_POINT‘ geometry column with the household’s spatial coordinates (longitude/latitude), concentrated towards the town centre.

It has an attribute ‘"parameters"‘ holding the true generating parameters used in the simulation (‘easiness‘, ‘discrimination‘, ‘abilities‘ (the true latent factor scores, useful for checking recovery), ‘effect‘, ‘resid_params‘ (list with ‘sd‘ and ‘corr‘, the residual covariance of the latent factors), ‘mgp_params‘ (list with ‘sd‘ and ‘phi‘, the spatial Gaussian process hyperparameters)), which is convenient for specifying sensible restrictions and checking recovery of the model in examples and tests.

Source

Simulated; see ‘data-raw/ipixuna.R‘ for the generating code and ‘attr(ipixuna, "parameters")‘ for the generating values.

Examples

```
data(ipixuna)
head(dplyr::select(ipixuna, -geometry, -items))
attr(ipixuna, "parameters")
```

plot.spifa *Default Plot of spifa Posterior Samples*

Description

The plot default for a fitted spifa object: a combined trace + density view, plot_trace on the left and plot_density on the right, one row per parameter and matched row for row, via wrap_plots – convergence and posterior shape at a glance, right after fitting. Defaults to the easiness (c) and discrimination (A) parameters. For a credible-interval view, or for any other parameter group, call plot_trace/plot_density/ plot_interval directly instead.

Usage

```
## S3 method for class 'spifa'
plot(x, select = c("c", "A"), nshow = 10, ...)
```

Arguments

x	A fitted spifa object, as returned by spifa.
select	Parameters to plot, as in plot_trace. Defaults to c("c", "A").
nshow	As in plot_trace, but applied once up front so both panels show the same parameters: calling plot_trace/ plot_density separately with the same select can otherwise each pick a different random subsample.
...	Further arguments passed to both plot_trace and plot_density.

Value

A patchwork object (a ggplot-like object).

Author(s)

Erick A. Chacón-Montalván

Examples

```
data(ipixuna)
samples <- spifa(items ~ 1, data = ipixuna, nfactors = 3, ngp = 0, niter = 1000)

plot(samples)
plot(samples, select = "c")
```

plot_density *Density Plot of spifa Posterior Samples*

Description

Draws posterior density curves directly from a fitted `spifa` model, in one of two formats: overlaid (`facet = FALSE`, the default; stacked, slightly-overlapping ridgeline plot via `geom_density_ridges` – the more insightful default for comparing many parameters' shapes and locations at a glance) or faceted (`facet = TRUE`; one panel per parameter). Shows the density shape only – for credible intervals and point estimates, see `plot_interval`.

Usage

```
plot_density(
  x,
  select,
  facet = FALSE,
  burnin = 0,
  thin = 1,
  nshow = 10,
  ncol = 1,
  facet_scales = "free",
  scale = 1.2,
  ...
)
```

Arguments

`x`	A fitted `spifa` model.
`select`	Parameters to plot, as in `plot_trace`.
`facet`	Logical; if `TRUE`, draw one panel per parameter; if `FALSE` (default), overlay every parameter as a ridgeline plot. See Description.
`burnin`	Number of initial iterations to discard.
`thin`	Thinning interval applied after `burnin`.
`nshow`	As in `plot_trace`.
`ncol`	Number of columns in the facet grid (`facet = TRUE` only); defaults to a single column.
`facet_scales`	The `scales` argument of `facet_wrap` (`facet = TRUE` only): one of `"free"` (default; each panel gets its own x/y-axis), `"free_y"` (one shared x-axis, line shown only on the bottom panel of each column, matching `plot_trace`'s facet – useful for comparing parameters on a similar scale), `"free_x"`, or `"fixed"`. Parameters with very different value ranges can render compressed/hard to read with a shared x-axis.
`scale`	Amount of vertical overlap between ridges (`facet = FALSE` only), passed to `geom_density_ridges`. Defaults to `1.2`.
`...`	Currently unused.

Value

A ggplot object.

Author(s)

Erick A. Chacón-Montalván

Examples

```
data(ipixuna)
samples <- spifa(items ~ 1, data = ipixuna, nfactors = 3, ngp = 0, niter = 1000)

plot_density(samples, select = "c", facet = FALSE)
plot_density(samples, select = "c", facet = TRUE)

# more than nshow (10) parameters: a random subsample is shown
plot_density(samples, select = "A", facet = FALSE, nshow = 6)
```

---

plot_interval *Interval Plot of spifa Posterior Samples*

Description

Draws a caterpillar/forest plot of posterior credible intervals directly from a fitted `spifa` model: one row per parameter, with a thin line for the `prob_outer` interval, a thick line for the `prob` interval, and a point at the `point_est`, computed directly from the raw draws (quantiles/median/mean). Unlike `plot_trace`/`plot_density`, this is a single combined view by design – comparing intervals side by side is the whole point, so there is no faceted alternative – but `sort` can reorder parameters by their point estimate, which a facet can't do usefully across independent panels.

Usage

```
plot_interval(
  x,
  select,
  horizontal = FALSE,
  burnin = 0,
  thin = 1,
  nshow = NULL,
  prob = 0.5,
  prob_outer = 0.9,
  point_est = c("median", "mean"),
  sort = FALSE,
  reference = NULL,
  ...
)
```

Arguments

x	A fitted spifa model.
select	Parameters to plot, as in plot_trace.
horizontal	Logical; if FALSE (default), parameters run along the x-axis and values run along the y-axis; if TRUE, the axes are swapped (a forest-plot layout).
burnin	Number of initial iterations to discard.
thin	Thinning interval applied after burnin.
nshow	As in plot_trace (a random subsample when select matches more than nshow parameters), but NULL (show every matched parameter) by default: unlike a faceted plot, a single interval plot stays readable with many more than 10 rows.
prob	Width of the thick (inner) credible interval (a central quantile interval). Defaults to 0.5.
prob_outer	Width of the thin (outer) credible interval. Defaults to 0.9.
point_est	Either "median" (default) or "mean".
sort	Logical; if TRUE, reorder parameters by their point estimate instead of their natural order. Defaults to FALSE.
reference	Optional reference values to overlay (e.g. the true values in a simulation study), as a fourth marker alongside the interval and point estimate. Only valid when select is a single group name (e.g. "A"): an unnamed vector or matrix matching that group's own shape (e.g. parameters$discrimination, an nitems x nfactors matrix). Structurally-restricted parameters (dropped internally before plotting) are silently ignored if present in reference.
...	Currently unused.

Value

A ggplot object.

Author(s)

Erick A. Chacón-Montalván

Examples

```
data(ipixuna)
samples <- spifa(items ~ 1, data = ipixuna, nfactors = 3, ngp = 0, niter = 1000)

plot_interval(samples, select = "c")
plot_interval(samples, select = "c", sort = TRUE)
plot_interval(samples, select = "A", horizontal = TRUE)

# overlay a reference set of discrimination values (e.g. from theory)
nitems <- ncol(ipixuna$items)
reference_A <- matrix(1, nitems, 3)
plot_interval(samples, select = "A", reference = reference_A)
```

plot_predict — *Map Posterior Predictions from a Fitted spifa Model*

Description

Maps a posterior summary of each latent factor (`stat`: the posterior mean by default, or any other function of the draws) at the locations `pred` was predicted at. Draws points if those locations are points (e.g. the observed households), or a filled polygon map if they are polygons (e.g. a prediction grid).

Usage

```
plot_predict(
  pred,
  grid = NULL,
  select = NULL,
  stat = mean,
  boundary = NULL,
  facet_scales = "fixed",
  ncol = NULL,
  ...
)
```

Arguments

`pred`	A `draws_array` as returned by `predict.spifa` for a spatial model.
`grid`	An `sf`/`sfc` object of the locations `pred` was predicted at (points or polygons), in the same row order as `pred`'s locations. Defaults to `attr(pred, "newdata")`, attached automatically by `predict.spifa`; only needed explicitly if that attribute is missing.
`select`	Factors to plot: an integer vector of factor indices, or `NULL` (default) for all.
`stat`	A function of one numeric vector (the draws at one location/factor) returning a single number – e.g. `mean` (the default), `sd`, `median`, or `function(v) mean(v > 1)` for an exceedance probability.
`boundary`	Optional `sf`/`sfc` polygon(s) drawn as an outline (e.g. the prediction area) on top of the map; has no effect on the computed/plotted values.
`facet_scales`	The `scales` argument of `facet_wrap`.
`ncol`	Number of columns in the facet grid; default lets `facet_wrap` choose.
`...`	Currently unused.

Details

`pred` is computed once via `predict.spifa` and can be reused across multiple `plot_predict()` calls – e.g. different `stat`/`select` values – without calling `predict()` again, which is normally the expensive step.

Value

A `ggplot` object with a clean map theme already applied (no axis titles, light dashed reference grid, small bottom legend). A fill (polygons) or colour (points) scale can be added as usual; a `theme`/`theme_*()` added afterwards overrides it as normal.

Author(s)

Erick A. Chacón-Montalván

Examples

```
library(sf)
data(ipixuna)

nitems <- ncol(ipixuna$items)
nfactors <- 3
A <- matrix(1, nitems, nfactors)
A[c(4, 8), 1] <- 0
A[c(2, 4, 5, 6, 7, 8, 10), 2] <- 0
A[c(5, 6), 3] <- 0

samples <- spifa(items ~ 1, data = ipixuna, nfactors = nfactors, niter = 5,
  constraints = list(discrimination = A))

# at the observed household locations (points)
pred <- predict(samples)
plot_predict(pred)

# on a grid (polygons); predict() takes the cell centroids for the
# spatial kernel but keeps the polygons for plot_predict() to map
grid <- st_sf(geometry = st_make_grid(ipixuna, n = c(3, 2)))
pred_grid <- predict(samples, newdata = grid)
plot_predict(pred_grid)
plot_predict(pred_grid, stat = sd)
plot_predict(pred_grid, stat = function (v) mean(v > 0)) # exceedance
```

`plot_trace` *Trace Plot of spifa Posterior Samples*

Description

Draws MCMC traceplots (iteration vs. value) directly from a fitted `spifa` model, in one of two formats: faceted (`facet = TRUE`, the default; one panel per parameter, with its own free y-scale, so a slow-mixing or small-variance parameter isn't visually flattened by others sharing the same axis) or overlaid (`facet = FALSE`; all series on a single panel, for a quick glance at overall convergence).

Usage

```
plot_trace(
  x,
  select,
  facet = TRUE,
  burnin = 0,
  thin = 1,
  nshow = 10,
  ncol = 1,
  legend = NULL,
  ...
)
```

Arguments

x	A fitted spifa model.
select	Parameters to plot, passed to the variable argument of subset_draws: either a group name (e.g. "A", matching every parameter in that group) or one or more full indexed names (e.g. "c[1]", paste0("A[", 1:10, ",1]")).
facet	Logical; if TRUE (default), draw one panel per parameter; if FALSE, overlay every series on a single panel. See Description.
burnin	Number of initial iterations to discard.
thin	Thinning interval applied after burnin.
nshow	If select matches more than nshow parameters, a random (sorted) subsample of nshow of them is shown instead of all of them. Set to NULL to always show every matched parameter.
ncol	Number of columns in the facet grid (facet = TRUE only); defaults to a single column.
legend	Legend position (facet = FALSE only): one of "bottom", "right", "none", or a logical. Defaults to NULL, which shows the legend unless there are more than 10 series, since a legend that large stops being readable and can dwarf the plot itself.
...	Currently unused.

Value

A ggplot object.

Author(s)

Erick A. Chacón-Montalván

Examples

```
data(ipixuna)
samples <- spifa(items ~ 1, data = ipixuna, nfactors = 3, ngp = 0, niter = 1000)
```

```
plot_trace(samples, select = "c", facet = TRUE)
plot_trace(samples, select = "c", facet = FALSE)

# more than nshow (10) parameters: a random subsample is shown
plot_trace(samples, select = "A", facet = TRUE, nshow = 6)

# explicit selection instead of a random subsample
plot_trace(samples, select = paste0("A[", 1:10, ",1]"), facet = TRUE)
```

`predict.spifa` *Predict the Latent Factors of a spifa Model*

Description

Predicts the latent factors of (spatial) item factor analysis for new subjects/locations and/or for new predictor values, using the posterior samples from a fitted `spifa` model.

Usage

```
## S3 method for class 'spifa'
predict(object, newdata = NULL, burnin = 0, thin = 1, joint = FALSE, ...)
```

Arguments

`object`	A fitted `spifa` object, as returned by `spifa`.
`newdata`	New data to predict at, mirroring `spifa`'s own `data` argument: an `sf`/`sfc` object (for spatial, i.e. `spifa`/`spifa_pred`, models – its geometry gives the new locations, and its CRS is used directly, so it need not match the training data's CRS) or a plain data frame (for `cifa_pred`), with columns matching the predictors used on the right-hand side of `formula` when the model was fitted. Its design matrix is built the same way `spifa` built the training one, using the same terms and factor levels. For a spatial model, its geometry may also be `POLYGON`/`MULTIPOLYGON` (e.g. a prediction grid): the centroid of each cell is used for the spatial kernel, but the original polygons are kept (see `Value`) so `plot_predict` can draw them as a filled map instead of points.
`burnin`	Number of initial (post-fitting) iterations to discard before using the posterior samples for prediction.
`thin`	Thinning interval applied to the posterior samples used for prediction.
`joint`	Logical; for spatial models (`spifa`/`spifa_pred`), whether the posterior predictive draws should respect the full predictive covariance across new locations and factors (`TRUE`), or be drawn marginally/independently per location-factor combination (`FALSE`, the default, cheaper). Each draw still propagates posterior parameter uncertainty (one draw per retained MCMC iteration) either way – this only controls whether, within a single draw, the values across new locations/factors are jointly correlated as the model implies. Marginal draws are

fine for per-location summaries (e.g. means, credible intervals computed independently per column); set TRUE when the samples themselves will be used as input to another model or computation that depends on their joint structure (e.g. a spatial contrast or aggregate across new locations). Ignored for cifa_pred, whose draws are already jointly correct across factors.

... Further arguments (currently unused).

Details

If the fitted model has no spatial or predictor structure (eifa or cifa), or if newdata is not supplied for a model that has one, there is nothing to predict beyond the latent abilities' own posterior samples already available from the fit, so those are returned directly (subject to burnin/thin) – this case is fast, since no new sampling is needed. Otherwise, new draws are sampled for the new locations and/or predictor values, which takes longer.

If the fitted model has predictors (cifa_pred/spifa_pred), newdata must include those predictor columns, the same as predict.lm and similar methods require – this is an error otherwise. There's no synthesized reference-level fallback for missing predictors (e.g. an sf/sfc object holding only new locations, with no predictor columns at all): for a factor predictor under the no-intercept encoding spifa uses, an automatically-filled all-zero row wouldn't correspond to any real category, so any reference profile – including all zeros for numeric predictors – must be supplied explicitly in newdata, matching the original data's format.

Value

A draws_array of posterior predictive samples of the latent abilities (theta) for the requested new locations and/or predictor values (or, if no prediction was requested, for the originally observed subjects). For a spatial model, the locations used (newdata, or the training locations if newdata was omitted) are attached as the "newdata" attribute, so plot_predict can map the result without needing it supplied again.

Author(s)

Erick A. Chacón-Montalván

Examples

```
library(sf)
data(ipixuna)

nitems <- ncol(ipixuna$items)
nfactors <- 3
A <- matrix(1, nitems, nfactors)
A[c(4, 8), 1] <- 0
A[c(2, 4, 5, 6, 7, 8, 10), 2] <- 0
A[c(5, 6), 3] <- 0

# Spifa model
samples <- spifa(items ~ 1, data = ipixuna, nfactors = nfactors, niter = 5,
  constraints = list(discrimination = A))
# latent abilities for observed locations
```

```
predict(samples)
# latent abilities for new locations
newdata <- st_make_grid(ipixuna, n = c(3, 2), what = "centers")
predict(samples, newdata = newdata)

# Spifa model with predictors
samples_pred <- spifa(items ~ wealth, data = ipixuna, nfactors = nfactors, niter = 5,
  constraints = list(discrimination = A))
newdata_pred <- st_sf(wealth = rnorm(6), geometry = newdata)
predict(samples_pred, newdata = newdata_pred)
```

`print.spifa` *Print spifa Posterior Samples*

Description

Prints a fitted `spifa` object (the output of `spifa`): model type, formula, data dimensions, MCMC settings, and a posterior summary table (via `summary.spifa`).

Usage

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

Arguments

`x`	An object of class `spifa`, as returned by `spifa`.
`...`	Further arguments passed to methods (currently unused).

Value

Invisibly returns `x`.

Author(s)

Erick A. Chacón-Montalván

Examples

```
data(ipixuna)
samples <- spifa(items ~ 1, data = ipixuna, nfactors = 3, ngp = 0, niter = 20)
samples
```

spifa *Bayesian Spatial Item Factor Analysis*

Description

Fits exploratory, confirmatory, and spatial item factor analysis (IFA) models for binary responses using full Bayesian inference. The model represents each binary response as a thresholded continuous auxiliary variable explained by `nfactors` latent abilities, optionally extended with linear predictors and/or a multivariate Gaussian process to capture spatial dependence in the latent factors (see `vignette("spifa")` for a full worked example). Inference is done via Gibbs sampling with adaptive Metropolis-Hastings updates for the spatial range and correlation parameters.

Usage

```
spifa(
  formula,
  data,
  nfactors,
  ngp = nfactors,
  niter = 100,
  thin = 1,
  burnin = 0,
  standardize = TRUE,
 constraints = list(discrimination = NULL, loading = NULL, sd = rep(1, nfactors)),
 priors = list(easiness = list(initial = NULL, mean = NULL, sd = NULL), discrimination =
   list(initial = NULL, mean = NULL, sd = NULL), effect = list(initial = NULL, mean =
   NULL, sd = NULL), corr = list(initial = NULL, eta = 1.5), loading = list(initial =
   NULL, mean = NULL, sd = NULL), range = list(initial = NULL, mean = NULL, sd = NULL)),
 adaptive = list(Sigma = NULL, Sigma_corr = NULL, Sigma_loading = NULL, Sigma_range =
    NULL, scale = 1, C = 0.7, alpha = 0.8, accep_prob = 0.234),
  execute = TRUE
)
```

Arguments

`formula`	A two-sided formula `items ~ predictors`. The left-hand side must be a single symbol naming a matrix-valued column of `data` holding the binary item responses (see Details). The right-hand side specifies predictors for the latent factors (e.g. `~ x1 + x2`); use `items ~ 1` for no predictors.
`data`	A data frame containing the item-response matrix column named on the left-hand side of `formula` and, if used, the predictor columns named on its right-hand side. If `data` is an `sf` object, its geometry (`st_geometry`) is used as the spatial coordinates and spatial Gaussian processes are added to the model.
`nfactors`	Number of latent factors (dimensions of the ability construct).
`ngp`	Number of independent Gaussian processes used to build the (possibly restricted) multivariate Gaussian process for the latent factors. Defaults to `nfactors` (one

GP per factor). Only relevant when `data` is an `sf` object; set to `0` to fit a non-spatial model even when `data` has a geometry column (the geometry and `ngp` are otherwise ignored in that case).

`niter` Number of MCMC iterations to run (after `burnin`) and store.

`thin` Thinning interval for the stored MCMC samples.

`burnin` Number of initial MCMC iterations to discard. These iterations still run (and the adaptive Metropolis-Hastings proposals for `loading`/`range`/`corr` still adapt through them), but they are never stored, so `niter` counts only the iterations that end up in the returned samples. `0` by default (no iterations discarded during fitting – the previous behaviour). Prefer this over discarding a prefix of the samples afterwards (e.g. via `summary(..., burnin = )`): iterations dropped here were never stored, so they don't cost memory or thinning-index arithmetic, and the adaptive proposals get to keep converging across the burnin/niter boundary rather than someone accidentally analysing them as if they were post-adaptation draws.

`standardize` Logical; if `TRUE` (default), the stored posterior draws are rescaled after fitting so the latent factors have unit variance: `theta` is divided by its posterior SD per factor, and `discrimination`, `effect`, `loading`, and the residual SD (`constraints$sd`) are compensated by the same factor so the fitted response probabilities are unchanged. This only applies to models with a spatial Gaussian process and/or predictor effects on the latent factors (`cifa_pred`/`spifa`/`spifa_pred`; ignored for `eifa`/`cifa`, where the residual SD is the only source of `theta`'s variance and there's nothing to normalize against). Those model types have a multiplicative scale non-identifiability between `theta` and `discrimination`/the GP variance/the predictor effect – an equally good fit can shrink one and inflate the other – so leaving `TRUE` keeps draws on an interpretable, comparable scale. Set to `FALSE` to keep the raw, unscaled posterior, e.g. when checking recovery of known simulated parameters (see `dev/simulated/analyze-ipixuna.R`), where an extra rescale would make draws harder to compare directly against the true simulated values. No predictors are standardized by this argument – despite the name, it does not touch `formula`'s right-hand side at all.

`constraints` Named list of constraints associated to the factor model. Accepted names are 'discrimination', 'loading', and 'sd'. The restrictions on the discrimination parameter should be placed in the element 'discrimination' with same dimensions as the discrimination matrix (nitems x nfactors). A value of 0 indicates that the link between the item and the factor is disabled and 1 indicates that it remains active and the coefficient associated will be estimated. If left unspecified ('NULL'), a lower-triangular restriction is applied by default (the item on row 'i' can only load on factors '1:i') – this is a standard identifiability trick for exploratory factor analysis, not a fully unrestricted model, and gives EIFA (see Details). The restrictions for the multivariate Gaussian process loading matrix should be placed in the element 'loading' with dimensions nfactors x ngp, such as a value of 0 indicates a link disconnected between the factor and the (independent) GP while 1 indicates that it remains active. The restrictions with respect to the standard deviation of the latent factors' residual term should be placed in the element 'sd', which should be a vector (length nfactors) providing the fixed values for that standard deviation (paired with 'priors$corr', together

they parameterize the residual covariance). If the model includes predictors or a Gaussian process, it is recommended to be lower than 1.

`priors` Named list of initial values and prior hyperparameters, one element per parameter group: ‘easiness‘, ‘discrimination‘, ‘effect‘ (predictor effect on the latent factors), ‘corr‘ (correlation of the latent factors’ residual term, paired with ‘constraints$sd‘), ‘loading‘ (multivariate Gaussian process loading matrix, paired with ‘constraints$loading‘), and ‘range‘ (multivariate Gaussian process scale parameters). Each element (except ‘corr‘) accepts ‘initial‘, ‘mean‘, and ‘sd‘; ‘corr‘ accepts ‘initial‘ and ‘eta‘ (the LKJ prior shape parameter). See the parameter glossary above for how these names map to the fitted model’s sampled output.

For ‘eifa‘ specifically, ‘corr‘ is never estimated (see Details) – ‘priors$corr$initial‘ (an ‘nfactors x nfactors‘ correlation matrix, 1s on the diagonal; identity if left unspecified) instead fixes the correlation between the latent factors at that value for the whole fit, and ‘priors$corr$eta‘ has no effect at all.

`adaptive` Named list of hyperparameters associated with the adaptive sampling. The adaptive sampling is done jointly for the ‘correlation‘ parameters, ‘standard deviation of the gps‘ and ‘scale parameter of the gps‘. The matrix ‘Sigma‘ can be provided as the full covariance matrix of these parameters for the proposal distribution. Otherwise, part of this matrix can be provided by using the elements ‘Sigma‘, ‘Sigma_corr‘, ‘Sigma_loading‘ and ‘Sigma_range‘. Additional elements are ‘scale‘, ‘C‘, ‘alpha‘ and ‘accep_prob‘ which are hyperparameters of the adaptive sampling proposed in Andrieu and Thomas (2008).

`execute` Logical value to run sampler or not. TRUE by default.

Details

The type of model fitted is determined automatically from `formula` and the class of `data`: a one-sided right-hand side (`items ~ 1`) with `constraints$discrimination` left unspecified gives exploratory IFA (EIFA) – a default lower-triangular restriction on `discrimination` is applied automatically for identifiability in this case, not something the user chooses (see `constraints` below). EIFA also never estimates the correlation between the latent factors: it stays fixed at `priors$corr$initial` (identity by default) for the whole fit – see `priors` below. Supplying your own (typically theory-driven) restricted `constraints$discrimination` instead gives confirmatory IFA (CIFA); adding predictors to the right-hand side (e.g. `items ~ x1`) gives CIFA with predictors; `data` being an `sf` object adds a spatial Gaussian process on the latent factors, giving spatial IFA (SPIFA), with or without predictors.

The left-hand side of `formula` must be a single symbol naming a matrix-valued column of `data` (`nobs` x `nitems`, one row per respondent, one column per binary item) — the same mechanism base R uses for multivariate `lm`. Build it with `I` (or ordinary `$<-` assignment) so it survives as a matrix column rather than being flattened into separate columns, e.g.:

```
items <- as.matrix(dplyr::select(data, `Item 1`:`Item 10`))
data$items <- items
spifa(items ~ x1, data = data, nfactors = 2)
```

Missing values are handled differently depending on where they occur. A missing item response (`NA` in the response matrix) does not drop that respondent: it is treated as an unobserved auxiliary

variable and sampled natively along with everything else. A missing predictor value (right-hand side of `formula`), by contrast, drops that respondent entirely, the same way `lm` and friends do.

Parameter glossary. `priors/constraints` use descriptive names; the fitted model's sampled output (as seen via `summary.spifa`, `plot_trace`, and friends) instead uses short internal names matching the underlying model notation. The two are deliberately different vocabularies (what you configure vs. what the sampler produced) – this table maps between them:

`priors/constraints` name	output group name	meaning
`easiness`	`c`	item easiness (intercept)
`discrimination`	`A`	item-factor discrimination (loading) matrix
`effect`	`B`	predictor effect on the latent factors
`corr`	`Corr`, `Chol`	residual correlation matrix, and its Cholesky factor
`sd`	*(fixed, not sampled)*	residual standard deviation
`loading`	`T`	multivariate Gaussian process loading matrix
`range`	`phi`	multivariate Gaussian process spatial range
(not user-set)	`Theta`	latent abilities
(not user-set)	`Z`	augmented latent response

Value

An object of class `spifa`: a `draws_array` of the MCMC samples (one variable per parameter, e.g. `c[1]`, `A[1,1]`, `Theta[1,1]`, ...), with an attribute `"fit_args"` recording the data and options used to fit the model (needed by `predict.spifa` and `dic`). See `summary.spifa`, `plot_trace`, `plot_density`, and `plot_interval` for working with it.

Author(s)

Erick A. Chacón-Montalván

References

Chacón-Montalván, E. A., Parry, L., Giorgi, E., Torres, P., Orellana, J. D. Y., Moraga, P., and Taylor, B. M. (2025). Mapping food insecurity in the Brazilian Amazon using a spatial item factor analysis model. The Annals of Applied Statistics, 19(4), 3438-3463. doi:10.1214/25AOAS2072

Examples

```
data(ipixuna)
nitems <- ncol(ipixuna$items)
nfactors <- 3

# EIFA: default discrimination constraints, non-spatial
samples_eifa <- spifa(items ~ 1, data = ipixuna, nfactors = nfactors, ngp = 0,
  niter = 20)

# discrimination constraint for cifa/spifa
A <- matrix(1, nitems, nfactors)
A[c(4, 8), 1] <- 0
A[c(2, 4, 5, 6, 7, 8, 10), 2] <- 0
A[c(5, 6), 3] <- 0
```

```
# CIFA with predictors: custom discrimination constraints, non-spatial
samples_cifa <- spifa(items ~ poly(wealth, 2), data = ipixuna, nfactors = nfactors,
  ngp = 0, niter = 20, burnin = 5, constraints = list(discrimination = A))

# SPIFA: custom discrimination constraints, default spatial components
samples_spifa <- spifa(items ~ 1, data = ipixuna, nfactors = nfactors,
  niter = 20, burnin = 5, thin = 2, constraints = list(discrimination = A))

# SPIFA with predictors and a custom loading matrix: factors 1 and 2
# share one Gaussian process, factor 3 gets its own
ngp <- 2
T <- matrix(c(1, 1, 0, 0, 0, 1), nfactors, ngp)
samples_spifa_pred <- spifa(items ~ wealth, data = ipixuna, nfactors = nfactors,
  ngp = 2, niter = 20, constraints = list(discrimination = A, loading = T))
```

`summary.spifa` *Summarise spifa Posterior Samples*

Description

Computes posterior summary statistics for every parameter in a fitted `spifa` object, via `summarise_draws`: mean, median, sd, mad, the 2.5%/10%/50%/90%/97.5% quantiles, effective sample size (bulk and tail), and `rhat`. `rhat` is computed via split-chain R-hat (Vehtari et al. 2021), which splits each chain in half and compares the halves – so it remains a meaningful convergence diagnostic even though `spifa` only ever fits a single chain. Discrimination parameters (`A`) structurally restricted to zero (via `constraints$discrimination`) are excluded, since they are fixed by construction rather than estimated.

Usage

```
## S3 method for class 'spifa'
summary(object, burnin = 0, thin = 1, select = NULL, ...)
```

Arguments

`object`	A fitted `spifa` object, as returned by `spifa`.
`burnin`	Number of initial iterations to discard.
`thin`	Thinning interval applied after discarding burn-in.
`select`	Character vector of parameter groups to summarise (defaults to all of them). An error if any requested group does not exist in the fitted model (e.g. `"T"` for a model with no spatial process).
`...`	Further arguments passed to methods (currently unused).

Value

A tibble with one row per parameter; see summarise_draws for column details.

Author(s)

Erick A. Chacón-Montalván

Examples

```
data(ipixuna)
nitems <- ncol(ipixuna$items)
nfactors <- 3

# discrimination constraint: start with every item free to load on every
# factor, then restrict a few items per factor based on what each item is
# meant to measure (0 = no relationship, 1 = free parameter to estimate)
A <- matrix(1, nitems, nfactors)
A[c(4, 8), 1] <- 0
A[c(2, 4, 5, 6, 7, 8, 10), 2] <- 0
A[c(5, 6), 3] <- 0
samples <- spifa(
  items ~ 1, data = ipixuna, nfactors = nfactors, ngp = 0,
  niter = 20, standardize = FALSE,
  constraints = list(discrimination = A))
summary(samples, select = "c")
```

---

update.spifa — *Continue Sampling from the Posterior of a spifa Model*

Description

Runs niter additional MCMC iterations, warm-started from the last posterior draw of object (the easiness/discrimination/residual correlation/predictor-effect/Gaussian process parameters), reusing the data, priors, and constraints of the original fit. Returns only the new draws, not concatenated with object's – see Details.

Usage

```
## S3 method for class 'spifa'
update(object, niter = 100, thin = 1, burnin = 0, ...)
```

Arguments

object	A fitted spifa object, as returned by spifa with execute = TRUE.
niter	Number of additional MCMC iterations to run and store.
thin	Thinning interval for the newly stored MCMC samples.

burnin	Number of initial iterations of this continuation to discard before storing (see spifa's burnin).
...	Further arguments (currently unused).

Details

The adaptive Metropolis-Hastings proposal tuning (for the residual correlation and, for spifa/spifa_pred, the Gaussian process parameters) resumes from wherever object's own run left it off, rather than restarting from object's original adaptive settings – so a chain of update() calls keeps refining its proposal instead of re-paying for adaptation each time.

update() always uses the same standardize setting object itself was originally fit with (see spifa); it isn't an argument here. standardize's rescale is still computed independently for each call, from that call's own posterior draws, so object and the object returned here can end up on slightly different absolute scales even though both represent the same continuous chain. Combine them yourself (e.g. rbind() on their as_draws_matrix form) if you want one continuous chain; fit with standardize = FALSE in the first place if you want every continuation on a genuinely identical, unrescaled scale.

Value

A new spifa object holding only the continuation draws (see Description).

Author(s)

Erick A. Chacón-Montalván

Examples

```
data(ipixuna)
samples <- spifa(items ~ 1, data = ipixuna, nfactors = 3, ngp = 0, niter = 1000)
more_samples <- update(samples, niter = 1000)
```

Index

as_draws_matrix, *22*

dic, 3, *19*
dic.spifa, *3*, 4
draws_array, *10*, *14*, *19*

facet_wrap, *7*, *10*

geom_density_ridges, *7*

I, *18*
ipixuna, 5

lm, *18, 19*

plot, *6*
plot.spifa, 6
plot_density, *6*, 7, *8*, *19*
plot_interval, *6, 7*, 8, *19*
plot_predict, 10, *13, 14*
plot_trace, *6–9*, 11, *19*
predict.lm, *14*
predict.spifa, *10*, 13, *19*
print.spifa, 15

sf, *13*, *16*, *18*
sfc, *13*
spifa, *2*, *4*, *6*, *13–15*, 16, *20–22*
spifa-package, 2
st_geometry, *16*
subset_draws, *12*
summarise_draws, *20, 21*
summary.spifa, *15*, *19*, 20

theme, *11*
tibble, *4*, *21*

update.spifa, 21

wrap_plots, *6*