Package: glt (via r-universe)

October 29, 2026

Title Bayesian Factor Modelling via Generalized Lower Triangular Priors

Version 1.0

Description Sparse Bayesian factor models for multivariate data, built around a flexible Generalized Lower Triangular (GLT) process prior on the loading matrix by Canale and Frühwirth-Schnatter (2026) <doi:10.48550/arXiv.2610.09762>. The number of active factors, their sparsity pattern, and the loadings themselves are learned jointly via a Markov chain Monte Carlo sampler implemented in C++ with 'Rcpp' and 'RcppArmadillo'.

License GPL-2

Encoding UTF-8

LinkingTo Rcpp, RcppArmadillo

Imports Rcpp, sparvaride

Config/roxygen2/version 8.1.0

NeedsCompilation yes

Author Antonio Canale [aut, cre]

Maintainer Antonio Canale <antonio.canale@unipd.it>

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

Date/Publication 2026-10-29 16:17:56 UTC

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

RemoteRef HEAD

RemoteSha 3c27f9e9e89a40619de65b2c52b67cee7bad88bd

Contents

glt-package 2
facmatrix 3
get_pivots 3
get_variance 4
gltfa 4

gltfa_star 7
plot.facmatrix 9
plot_real_matrix 9
print.summary.gltfit 11
summary.gltfit 11

Index **13**

`glt-package` *gltfactor: Bayesian Factor Models with GLT Process Priors*

Description

Fits a Bayesian Gaussian factor model under a Generalized Lower Triangular (GLT) process prior using an MCMC sampler implemented in C++ via Rcpp. The package provides tools for model fitting, posterior summarisation, identification post-processing, and visualisation of loading matrices and variance decompositions.

Details

The main entry point is `gltfa()`, which accepts a data matrix `y` (T x m) and returns a `"gltfit"` object containing posterior draws and metadata.

Main functions

`gltfa()` Fit a GLT factor model via MCMC.

`get_variance()` Compute posterior draws of the factor and total covariance matrices.

`get_pivots()` Extract pivot row indices from a binary allocation matrix `Delta`.

`plot_real_matrix()` Visualise a numeric matrix as a colour image using a diverging palette.

Author(s)

Maintainer: Antonio Canale <antonio.canale@unipd.it>

Authors:

- Antonio Canale <antonio.canale@unipd.it>

References

Canale, A. and Frühwirth-Schnatter, S. (2026). "The generalized lower triangular process prior." *Technical Report*,

facmatrix *Construct a "facmatrix" object*

Description

A thin wrapper that tags a numeric matrix with class `"facmatrix"` so that it is plotted via `plot.facmatrix()` (a colour-image raster) rather than via the default matrix print/plot methods.

Usage

```
facmatrix(mat)
```

Arguments

`mat`	A numeric matrix.

Value

`mat` with class `"facmatrix"` prepended.

get_pivots *Extract pivot row indices from a Delta matrix*

Description

For each column of `Delta`, returns the row index of the first `1`, which defines the pivot row for that factor.

Usage

```
get_pivots(Delta)
```

Arguments

`Delta`	An `m x H` integer matrix with entries in `{0, 1}`. Each column must contain at least one `1`.

Value

An integer vector of length `H`.

get_variance *Posterior draws of variance decomposition*

Description

For each saved MCMC draw, computes the $m \times m$ matrices $\Lambda\Lambda^\top$ (factor covariance) and $\Lambda\Lambda^\top + \Sigma$ (total marginal covariance), where $\Sigma = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_m^2)$.

Usage

```
get_variance(fit)
```

Arguments

fit	A gltfit object returned by gltfa().

Value

A named list with two elements:

LLt An $m \times m \times S$ array of $\Lambda_s \Lambda_s^\top$ draws.

total An $m \times m \times S$ array of $\Lambda_s \Lambda_s^\top + \Sigma_s$ draws.

gltfa *Fit a Bayesian factor model with GLT process prior*

Description

Fits a Bayesian Gaussian factor model under a Generalized Lower Triangular (GLT) process prior using an MCMC sampler implemented in C++ via Rcpp. This is the main user-facing entry point of the package.

Usage

```
gltfa(
  y,
  mcmc = list(),
  Hmax = ncol(y),
  prior = list(),
  init = list(),
  control = list(),
  fixed = list(),
  verbose = interactive()
)
```

Arguments

`y`	A numeric $T \times m$ data matrix with observations in rows and variables in columns. Missing values are not supported.
`mcmc`	A named list of MCMC settings. Recognised fields: `niter` Positive integer. Total number of MCMC iterations. Default `5000`. `nburn` Non-negative integer. Number of burn-in iterations to discard. Must be strictly less than `niter`. Default `2500`. `thin` Positive integer. Thinning interval. Default `1`.
`Hmax`	A positive integer giving a conservative upper bound for the number of active factors. Default `ncol(y)`.
`prior`	A named list of prior hyperparameters. Recognised fields: `a_nu` Positive numeric. First shape of the Beta prior on the global inclusion probability ν. Default `1`. `b_nu` Positive numeric. Second shape of the Beta prior on ν. Default `1`. `alpha` Positive numeric. Shape of the Gamma prior on pivot loadings. Default `1`. `beta` Positive numeric. Rate of the Gamma prior on pivot loadings. Default `1`. `kappa` Positive numeric. Precision of the normal prior on non-pivot loadings. Default `1`. `a_sigma` Positive numeric. Shape of the inverse-Gamma prior on idiosyncratic variances σ_i^2. Default `2`. `b_sigma` Positive numeric. Scale of the inverse-Gamma prior on σ_i^2. Default `2`.
`init`	A named list of initial values for the sampler. Recognised fields: `H` Single non-negative integer. Initial number of active factors. `Lambda` Numeric $m \times H$ loading matrix. `Delta` Integer $m \times H$ binary allocation matrix with entries in $\{0, 1\}$. `Eta` Numeric $H \times T$ factor score matrix. `ell` Integer vector of length H with pivot row indices. `tau` Numeric vector of length H with initial column-specific inclusion probabilities. `nu` Single numeric in $(0, 1)$. Initial global sparsity parameter. `sigma2` Positive numeric vector of length m. Initial idiosyncratic variances.
`control`	A named list of algorithmic and storage options. Recognised fields: `store_draws` Logical. Whether to store posterior draws of `Lambda`, `Delta`, `sigma2`, and `H`. Default `TRUE`. `store_eta` Logical. Whether to additionally store draws of the factor score matrix `Eta`. Default `FALSE`. `print_every` Positive integer. Print progress every this many iterations (only when `verbose = TRUE`). Default `100`. `seed` Single integer or `NULL`. Random seed for the C++ sampler. Default `NULL`.
`fixed`	A named list of logical flags indicating which MCMC updates to skip. For example, `list(H = TRUE)` keeps the number of factors fixed at its initial value throughout sampling.
`verbose`	Logical. Whether to print sampling progress to the console. Defaults to `interactive()`.

Details

The model is:

$$y_t = \Lambda \eta_t + \epsilon_t, \quad \epsilon_t \sim N_m(0, \Sigma)$$

where Λ is a sparse lower-triangular loading matrix controlled by a binary allocation matrix Δ, and $\Sigma = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_m^2)$.

Notation and dimensions:

- `y` is a numeric matrix of dimension $T \times m$.
- `Lambda` is a loading matrix of dimension $m \times H$.
- `Delta` is a binary allocation matrix of dimension $m \times H$.
- `Eta` is a factor score matrix of dimension $H \times T$.

Value

An object of class `"gltfit"`, a named list with components:

`draws` A named list of posterior draws (if `control$store_draws = TRUE`): `Lambda` ($m \times H_{\max} \times n_{\text{save}}$), `Delta` ($m \times H_{\max} \times n_{\text{save}}$), `sigma2` ($m \times n_{\text{save}}$), `H` ($n_{\text{save}}$), and optionally `Eta` ($H_{\max} \times T \times n_{\text{save}}$).

`meta` A named list with `nsave`, `Hmax`, `T`, `m`.

`input` A named list echoing the processed `mcmc`, `prior`, `control`, and `fixed` arguments.

`call` The matched call.

References

Canale, A. and Frühwirth-Schnatter, S. (2026). "The generalized lower triangular process prior." *arXiv preprint arXiv:2610.09762*.

See Also

`get_variance()`, `get_pivots()`, `plot_real_matrix()`

Examples

```
set.seed(42)
Y <- matrix(rnorm(200 * 10), nrow = 200, ncol = 10)

fit <- gltfa(
  y       = Y,
  mcmc    = list(niter = 2000, nburn = 1000, thin = 1),
  Hmax    = 5,
  prior   = list(a_sigma = 2, b_sigma = 1),
  init    = list(H = 2),
  control = list(store_draws = TRUE, seed = 1L, random_scan = TRUE),
  verbose = FALSE
)
```

gltfa_star *Fit a STAR-extended GLT factor model for discrete/mixed-scale data*

Description

Fits the Kowal & Canale (2020) simultaneous transformation and rounding (STAR) extension of `gltfa()` to discrete-valued (e.g. counts, ordinal) observed data. A continuous latent matrix `z` is linked to the observed matrix `y` via a column-specific monotone transform `G_j` and a column-specific threshold sequence, and modelled with a regression mean plus the same sparse GLT factor model used by `gltfa()`:

Usage

```
gltfa_star(
  y,
  X = NULL,
  thresholds = NULL,
  g_type = "exp",
  mcmc = list(),
  Hmax = ncol(y),
  prior = list(),
  init = list(),
  control = list(),
  fixed = list(),
  verbose = interactive()
)
```

Arguments

`y`	An integer (or coercible) $T \times m$ matrix of observed, 0-indexed levels $(0, 1, \ldots, K_j - 1$ for column j). Missing values are not supported.
`X`	A numeric $T \times p$ covariate matrix, or `NULL` (default) for a zero-mean latent process ($p = 0$).
`thresholds`	A list of length `ncol(y)`. Element `j` is either `NULL` (use the default integer-spaced cutpoints $1, 2, \ldots, \max(y_{\cdot j})+1$, reproducing floor rounding) or a strictly increasing numeric vector of $\max(y_{\cdot j})+1$ interior cutpoints $(a_{j,1}, \ldots, a_{j,K_j-1})$. Columns may have different numbers of levels K_j. Default `NULL` uses the default for every column.
`g_type`	Character vector of length 1 or `ncol(y)`, each entry one of `"identity"` or `"exp"`, giving the column-specific monotone transform G_j. Default `"exp"` for every column (count-data STAR).
`mcmc, Hmax, control, fixed, verbose`	As in `gltfa()`.
`prior`	A named list of prior hyperparameters; see `gltfa()` for the factor-model hyperparameters (`a_nu`, `b_nu`, `alpha`, `beta`, `kappa`, `a_sigma`, `b_sigma`, ...), plus regression hyperparameters:

`b0_beta` Numeric vector of length `ncol(X)`. Prior mean for every column's regression coefficients. Default `0`.

`V_beta_inv` `p x p` prior precision matrix (common across columns), i.e. `beta_j ~ N(b0_beta, sigma2_j * V_beta_inv^{-1})`. Default `NULL`, in which case `V_beta_inv = I_p / c0_beta`.

`c0_beta` Positive numeric. Prior variance scale used when `V_beta_inv` is not supplied. Default `100`.

`init` A named list of initial values; see `gltfa()` for the factor-model fields, plus:

`Beta` Numeric $p \times m$ matrix. Default all zero.

`z` Numeric $T \times m$ matrix. Default a bound-consistent midpoint initialisation.

Details

$$y_{ij} = \ell \iff G_j(z_{ij}) \in [a_{j,\ell}, a_{j,\ell+1})$$
$$z_{ij} = x_i'\beta_j + \epsilon_{ij}, \quad \epsilon_t \sim N_m(0, \Lambda\Lambda'\text{-structured } \Sigma)$$

Two extra Gibbs steps run before the original 8-step sampler on each iteration: imputing `z` from its truncated-normal full conditional (Step 0a), and updating the regression coefficients `Beta` (Step 0b, conjugate normal). The residual `epsilon = z - X Beta` is then fed into the unchanged Delta/pivots/H/Lambda/sigma2/Eta updates.

Value

An object of class `c("gltfit_star", "gltfit")`. In addition to the components documented in `gltfa()`, `draws$Beta` holds `p x m` draws of the regression coefficients (always stored), and `last$z` / `last$epsilon` hold the final latent-z / residual draws (not stored across iterations).

References

Canale, A. and Frühwirth-Schnatter, S. (2026). "The generalized lower triangular process prior." *arXiv preprint arXiv:2610.09762*,

Kowal, D. and Canale, A. (2020). "Simultaneous transformation and rounding (STAR) models for integer-valued data." *Electronic Journal of Statistics*, 14(1), 1744-1772.

See Also

`gltfa()`

Examples

```
set.seed(42)
Ycounts <- matrix(rpois(100 * 6, lambda = 3), nrow = 100, ncol = 6)

fit <- gltfa_star(
  y       = Ycounts,
  Hmax    = 3,
  mcmc    = list(niter = 500, nburn = 250, thin = 1),
  init    = list(H = 1),
```

```
  control = list(store_draws = TRUE, seed = 1L),
  verbose = FALSE
)
```

`plot.facmatrix` *Plot a "facmatrix" object*

Description

S3 `plot` method for objects of class `"facmatrix"`. Renders the matrix as a colour-image raster using `signed_palette()` by default; see `plot_real_matrix()` for the full set of supported arguments.

Usage

```
## S3 method for class 'facmatrix'
plot(x, ...)
```

Arguments

`x`	A `"facmatrix"` object (a numeric matrix with that class).
`...`	Additional arguments passed on to `plot_real_matrix()`.

Value

Invisibly `x`; called for its side effect of producing a plot.

See Also

`plot_real_matrix()`, `facmatrix()`

`plot_real_matrix` *Plot a real-valued matrix as a colour image*

Description

Renders a numeric matrix as a raster image using a diverging colour palette (via `signed_palette()` by default), with optional grid lines and axis labels. Rows increase downward (matrix convention).

Usage

```
plot_real_matrix(
  mat,
  palette_fun = signed_palette,
  draw_grid = TRUE,
  grid_col = "grey80",
  axes = TRUE,
  xlab = "column",
  ylab = "row",
  asp = 1,
  main = "",
  legend = FALSE,
  legend_n = 100,
  legend_width = 1.4,
  legend_digits = 2,
  v_lines = NULL,
  h_lines = NULL,
  ...
)
```

Arguments

mat	A numeric matrix to plot.
palette_fun	A function with signature f(x, ...) that maps a numeric vector to a character vector of hex colour codes. Defaults to signed_palette().
draw_grid	Logical. Whether to draw cell grid lines. Default TRUE.
grid_col	Character. Colour of the grid lines. Default "grey80".
axes	Logical. Whether to draw row and column axes. Default TRUE.
xlab	Character. Label for the horizontal axis. Default "column".
ylab	Character. Label for the vertical axis. Default "row".
asp	Numeric. Aspect ratio passed to plot(). Default 1.
main	Character. Plot title. Default "".
legend	Logical. Whether to draw a colour-scale legend to the right of the matrix image. Default FALSE.
legend_n	Integer. Number of colour steps used to build the legend gradient. Default 100.
legend_width	Numeric. Width of the legend panel in inches. Default 1.4.
legend_digits	Integer. Number of decimal digits shown on the legend axis labels. Default 2.
v_lines	Numeric vector or NULL. Column positions (on the data scale) at which to draw dashed vertical reference lines. Default NULL (no lines).
h_lines	Numeric vector or NULL. Row positions (on the data scale) at which to draw dashed horizontal reference lines. Default NULL (no lines).
...	Additional arguments forwarded to palette_fun.

Details

This is the internal workhorse used by `plot.facmatrix()`; it remains available for plotting plain numeric matrices directly.

Value

Invisibly `NULL`; called for its side effect of producing a plot.

See Also

`signed_palette()`

`print.summary.gltfit` *Print a "summary.gltfit" object*

Description

Print a "summary.gltfit" object

Usage

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

Arguments

`x`	A `"summary.gltfit"` object, as returned by `summary.gltfit()`.
`...`	Currently unused.

Value

Invisibly `x`.

`summary.gltfit` *Summarize a "gltfit" object*

Description

Computes posterior averages (across admissible saved MCMC draws) of `Lambda`, `Delta`, `sigma2`, and `H`, along with the posterior average of $\Omega = \Lambda\Lambda^\top + \Sigma$, where $\Sigma = \mathrm{diag}(\sigma^2)$. The posterior-mean `Lambda`, `Delta`, and `Omega` matrices are returned with class `"facmatrix"` so they inherit `plot.facmatrix()`.

Usage

```
## S3 method for class 'gltfit'
summary(object, ...)
```

Arguments

`object`	An object of class `"gltfit"`, as returned by `gltfa()`.
`...`	Currently unused.

Details

Before averaging, draws are filtered using `admissible_draws()`: a draw is dropped when, after removing spurious columns (loadings on a single variable) and any resulting all-zero rows, the allocation matrix fails `sparvaride::counting_rule_holds()`. This mirrors the usual post-processing step for GLT-type factor models, where unidentified draws (e.g. those with too few active variables per factor) should not contribute to posterior summaries of `Lambda`/`Delta`.

`object$draws$Lambda` and `object$draws$Delta` are stored as ragged lists of length n_{save}, each element an $m \times H_s$ matrix (H_s is the active number of factors for that draw, see `object$draws$H`). Kept draws are zero-padded to $m \times H_{\max}$ before elementwise averaging. `sigma2` draws are stored as an $n_{\text{save}} \times m$ matrix, averaged over kept rows. Ω is computed per kept draw as $\Lambda_s \Lambda_s^\top + \operatorname{diag}(\sigma_s^2)$ (using the draw's own, unpadded Λ_s) and then averaged.

Value

A named list with class `"summary.gltfit"` containing:

`Lambda` Posterior mean loading matrix ($m \times H_{\max}$), class `"facmatrix"`.

`Delta` Posterior mean allocation matrix ($m \times H_{\max}$), class `"facmatrix"`.

`sigma2` Posterior mean idiosyncratic variances (length m).

`H` Posterior mean number of active factors (scalar).

`Omega` Posterior mean of $\Lambda\Lambda^\top + \Sigma$ ($m \times m$), class `"facmatrix"`.

`n_admissible` Number of draws retained after filtering.

`n_total` Total number of saved draws before filtering.

See Also

`gltfa()`, `plot.facmatrix()`, `sparvaride::counting_rule_holds()`

Index

facmatrix, 3
facmatrix(), *9*

get_pivots, 3
get_pivots(), *2*, *6*
get_variance, 4
get_variance(), *2*, *6*
glt *(*glt-package*)*, 2
glt-package, 2
gltfa, 4
gltfa(), *2*, *4*, *7, 8*, *12*
gltfa_star, 7

plot(), *10*
plot.facmatrix, 9
plot.facmatrix(), *3*, *11, 12*
plot_real_matrix, 9
plot_real_matrix(), *2*, *6*, *9*
print.summary.gltfit, 11

sparvaride::counting_rule_holds(), *12*
summary.gltfit, 11
summary.gltfit(), *11*