

Package ‘glt’

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>](https://doi.org/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 CRAN

Date/Publication 2026-10-29 11:10:01 UTC

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

`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 ($T \times m$) 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 Δ .

`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	<i>Construct a "facmatrix" object</i>
-----------	---------------------------------------

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	<i>Extract pivot row indices from a Delta matrix</i>
------------	--

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 $m \times H$ integer matrix with entries in $\{0, 1\}$. Each column must contain at least one 1.
-------	--

Value

An integer vector of length H.

get_variance	<i>Posterior draws of variance decomposition</i>
--------------	--

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 $\Sigma = \text{diag}(\sigma_1^2, \dots, \sigma_m^2)$.

Usage

```
get_variance(fit)
```

Arguments

`fit` A gltf object returned by `gltf()`.

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.

gltf	<i>Fit a Bayesian factor model with GLT process prior</i>
------	---

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

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

Arguments

<code>y</code>	A numeric $T \times m$ data matrix with observations in rows and variables in columns. Missing values are not supported.
<code>mcmc</code>	A named list of MCMC settings. Recognised fields: <ul style="list-style-type: none"> <code>niter</code> Positive integer. Total number of MCMC iterations. Default 5000. <code>nburn</code> Non-negative integer. Number of burn-in iterations to discard. Must be strictly less than <code>niter</code>. Default 2500. <code>thin</code> Positive integer. Thinning interval. Default 1.
<code>Hmax</code>	A positive integer giving a conservative upper bound for the number of active factors. Default <code>ncol(y)</code> .
<code>prior</code>	A named list of prior hyperparameters. Recognised fields: <ul style="list-style-type: none"> <code>a_nu</code> Positive numeric. First shape of the Beta prior on the global inclusion probability ν. Default 1. <code>b_nu</code> Positive numeric. Second shape of the Beta prior on ν. Default 1. <code>alpha</code> Positive numeric. Shape of the Gamma prior on pivot loadings. Default 1. <code>beta</code> Positive numeric. Rate of the Gamma prior on pivot loadings. Default 1. <code>kappa</code> Positive numeric. Precision of the normal prior on non-pivot loadings. Default 1. <code>a_sigma</code> Positive numeric. Shape of the inverse-Gamma prior on idiosyncratic variances σ_i^2. Default 2. <code>b_sigma</code> Positive numeric. Scale of the inverse-Gamma prior on σ_i^2. Default 2.
<code>init</code>	A named list of initial values for the sampler. Recognised fields: <ul style="list-style-type: none"> <code>H</code> Single non-negative integer. Initial number of active factors. <code>Lambda</code> Numeric $m \times H$ loading matrix. <code>Delta</code> Integer $m \times H$ binary allocation matrix with entries in $\{0, 1\}$. <code>Eta</code> Numeric $H \times T$ factor score matrix. <code>ell</code> Integer vector of length H with pivot row indices. <code>tau</code> Numeric vector of length H with initial column-specific inclusion probabilities. <code>nu</code> Single numeric in $(0, 1)$. Initial global sparsity parameter. <code>sigma2</code> Positive numeric vector of length m. Initial idiosyncratic variances.
<code>control</code>	A named list of algorithmic and storage options. Recognised fields: <ul style="list-style-type: none"> <code>store_draws</code> Logical. Whether to store posterior draws of <code>Lambda</code>, <code>Delta</code>, <code>sigma2</code>, and <code>H</code>. Default TRUE. <code>store_eta</code> Logical. Whether to additionally store draws of the factor score matrix <code>Eta</code>. Default FALSE. <code>print_every</code> Positive integer. Print progress every this many iterations (only when <code>verbose = TRUE</code>). Default 100. <code>seed</code> Single integer or NULL. Random seed for the C++ sampler. Default NULL.
<code>fixed</code>	A named list of logical flags indicating which MCMC updates to skip. For example, <code>list(H = TRUE)</code> keeps the number of factors fixed at its initial value throughout sampling.
<code>verbose</code>	Logical. Whether to print sampling progress to the console. Defaults to <code>interactive()</code> .

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 $\Sigma = \text{diag}(\sigma_1^2, \dots, \sigma_m^2)$.

Notation and dimensions:

- y is a numeric matrix of dimension $T \times m$.
- Λ is a loading matrix of dimension $m \times H$.
- Δ is a binary allocation matrix of dimension $m \times H$.
- η is a factor score matrix of dimension $H \times T$.

Value

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

draws A named list of posterior draws (if `control$store_draws = TRUE`): Λ ($m \times H_{\max} \times n_{\text{save}}$), Δ ($m \times H_{\max} \times n_{\text{save}}$), σ^2 ($m \times n_{\text{save}}$), H (n_{save}), and optionally η ($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 <- gltf(
  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
)
```

gltf_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 [gltf\(\)](#) to discrete-valued (e.g. counts, ordinal) observed data. A continuous latent matrix $\mathbf{z}$ is linked to the observed matrix $\mathbf{y}$ 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 [gltf\(\)](#):

Usage

```
gltf_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

- | | |
|-------------------------------------|--|
| $\mathbf{y}$ | An integer (or coercible) $T \times m$ matrix of observed, 0-indexed levels $(0, 1, \dots, K_j - 1)$ for column j . Missing values are not supported. |
| $\mathbf{X}$ | A numeric $T \times p$ covariate matrix, or NULL (default) for a zero-mean latent process ($p = 0$). |
| thresholds | A list of length <code>ncol(y)</code> . Element j is either NULL (use the default integer-spaced cutpoints $1, 2, \dots, \max(y_{\cdot j}) + 1$, reproducing floor rounding) or a strictly increasing numeric vector of $\max(y_{\cdot j}) + 1$ interior cutpoints $(a_{j,1}, \dots, a_{j,K_j-1})$. 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 <code>ncol(y)</code> , 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 gltf() . |
| prior | A named list of prior hyperparameters; see gltf() for the factor-model hyperparameters (<code>a_nu</code> , <code>b_nu</code> , <code>alpha</code> , <code>beta</code> , <code>kappa</code> , <code>a_sigma</code> , <code>b_sigma</code> , ...), plus regression hyperparameters: |

b0_beta Numeric vector of length $\text{ncol}(X)$. Prior mean for every column's regression coefficients. Default 0.

V_beta_inv $p \times p$ prior precision matrix (common across columns), i.e. $\text{beta}_j \sim N(\text{b0_beta}, \text{sigma2_j} * \text{V_beta_inv}^{-1})$. Default NULL, in which case $\text{V_beta_inv} = \text{I}_p / \text{c0_beta}$.

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 `gltf()` 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 $\text{epsilon} = z - X \text{Beta}$ is then fed into the unchanged $\text{Delta/pivots/H/Lambda/sigma2/Eta}$ updates.

Value

An object of class `c("gltf_star", "gltf")`. In addition to the components documented in `gltf()`, `draws$Beta` holds $p \times m$ 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

`gltf()`

Examples

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

fit <- gltf_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	<i>Plot a "facmatrix" object</i>
----------------	----------------------------------

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 <code>plot_real_matrix()</code> .

Value

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

See Also

`plot_real_matrix()`, `facmatrix()`

plot_real_matrix	<i>Plot a real-valued matrix as a colour image</i>
------------------	--

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

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

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 $\Sigma = \text{diag}(\sigma^2)$. 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 $m \times H_{\text{max}}$ 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 $\Lambda_s \Lambda_s^\top + \text{diag}(\sigma_s^2)$ (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 ($m \times H_{\text{max}}$), class "facmatrix".

Delta Posterior mean allocation matrix ($m \times H_{\text{max}}$), 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](#)