Package 'BayesianDisaggregation'

October 16, 2025

Title Bayesian Methods for Economic Data Disaggregation

Version 0.1.2

Depends R (>= 4.1.0)

Description Implements a novel Bayesian disaggregation framework that combines Principal Component Analysis (PCA) and Singular Value Decomposition (SVD) dimension reduction of prior weight matrices with deterministic Bayesian updating rules. The method provides Markov Chain Monte Carlo (MCMC) free posterior estimation with built-in diagnostic metrics. While based on established PCA (Jolliffe, 2002) <doi:10.1007/b98835> and Bayesian principles (Gelman et al., 2013) <doi:10.1201/b16018>, the specific integration for economic disaggregation represents an original methodological contribution.

License MIT + file LICENSE

Encoding UTF-8

Imports readxl, dplyr, tidyr, stringr, foreach, doParallel, openxlsx, rlang, tibble, magrittr

Suggests knitr, rmarkdown, ggplot2, readr, testthat (>= 3.0.0)

RoxygenNote 7.3.3

VignetteBuilder knitr

NeedsCompilation no

Author José Mauricio Gómez Julián [aut, cre]

Maintainer José Mauricio Gómez Julián <isadore.nabi@pm.me>

Repository CRAN

Date/Publication 2025-10-16 11:40:21 UTC

Contents

`bayesian_disaggregate` *Run Bayesian disaggregation*

Description

Performs Bayesian disaggregation of an aggregated time series (e.g., CPI) into K components using one of four deterministic update rules: *weighted*, *multiplicative*, *dirichlet*, *adaptive*.

Usage

```
bayesian_disaggregate(
  path_cpi,
  path_weights,
  method = c("weighted", "multiplicative", "dirichlet", "adaptive"),
  lambda = 0.7,
  gamma = 0.1,
  coh_mult = 3,
  coh_const = 0.5,
  stab_a = 1000,
  stab_b = 10,
  stab_kappa = 50,
  likelihood_pattern = "recent"
)
```

Arguments

`path_cpi`	Path to the CPI Excel file. Must contain at least the columns `date` and `value` (case/locale tolerant in `read_cpi()`).
`path_weights`	Path to the Excel file with the baseline weight matrix (prior): either $T \times K$ (years in rows, sectors in columns) or a length-K vector (constant across time). Rows are renormalized to the simplex.
`method`	Disaggregation method: `"weighted"`, `"multiplicative"`, `"dirichlet"`, or `"adaptive"`.

`lambda` Weight for the `"weighted"` method in $[0, 1]$. Ignored otherwise.

`gamma` Uncertainty factor for the `"dirichlet"` method (> 0).

`coh_mult` Multiplier for the coherence increment $\Delta\rho$.

`coh_const` Constant offset for coherence, truncated to $[0, 1]$.

`stab_a` Sensitivity for row-sum deviation penalty $|\sum w - 1|$.

`stab_b` Sensitivity for negative-values penalty (count of negatives).

`stab_kappa` Sensitivity for temporal variation (average $|\Delta|$).

`likelihood_pattern` Temporal spreading pattern for the likelihood: `"constant"`, `"recent"`, `"linear"`, or `"bell"`.

Details

Assumptions: (i) prior/posterior rows lie on the simplex; (ii) no MCMC is used, updates are analytic/deterministic; (iii) `read_*` helpers coerce benign formatting issues and error on malformed inputs.

Value

A `list` with:

`years` Integer vector of years used.

`industries` Character vector of sector/column names.

`prior` Tibble prior $T \times (1 + K)$ with `Year` then sectors.

`likelihood_t` Tibble likelihood over time (same shape as `prior`).

`likelihood` Tibble `Sector`, `L` (length K).

`posterior` Tibble posterior $T \times (1 + K)$ (rows sum to 1).

`metrics` Tibble with hyperparameters + `coherence`, `stability`, `interpretability`, `efficiency`, `composite`, `T`, `K`.

See Also

`read_cpi`, `read_weights_matrix`, `compute_L_from_P`, `spread_likelihood`, `posterior_weighted`, `posterior_multiplicative`, `posterior_dirichlet`, `posterior_adaptive`, `coherence_score`, `stability_composite`, `interpretability_score`

Examples

```
# Minimal synthetic run (no files):
T <- 6; K <- 4
P <- matrix(rep(1/K, T*K), nrow = T)
L <- runif(K); L <- L/sum(L)
LT <- spread_likelihood(L, T, "recent")
W  <- posterior_weighted(P, LT, lambda = 0.7)
```

`coherence_score` *Coherence score (prior → posterior alignment improvement)*

Description

Measures how much the posterior W improves alignment with the sectoral signal L relative to the prior P. We compute the correlation increment $\Delta\rho = \max(0, \rho(W, L) - \rho(P, L))$ using `robust_cor` (chooses Pearson/Spearman by larger absolute value), then map it to $[0, 1]$ with `mult` and `const`:

$$\text{score} = \min\{1,\ \text{const} + \text{mult} \cdot \Delta\rho\}.$$

Usage

```
coherence_score(P, W, L, mult = 3, const = 0.5)
```

Arguments

`P`	Prior matrix ($T \times K$); rows should sum to 1 (approximately).
`W`	Posterior matrix ($T \times K$); rows should sum to 1 (approximately).
`L`	Likelihood vector (length K), non-negative and summing to 1.
`mult`	Non-negative multiplier applied to the correlation increment (default `3.0`).
`const`	Constant offset in $[0, 1]$ (default `0.5`).

Value

Scalar coherence score in $[0, 1]$.

Examples

```
T <- 6; K <- 4
P <- matrix(runif(T*K), T); P <- P/rowSums(P)
W <- matrix(runif(T*K), T); W <- W/rowSums(W)
L <- runif(K); L <- L/sum(L)
coherence_score(P, W, L)
```

`compute_L_from_P` *Compute likelihood vector from a prior matrix via SVD (center-only, robust)*

Description

Builds a sectoral likelihood L (length K) from the prior weights matrix $P \in \mathbb{R}^{T \times K}$ by taking the absolute value of the first right singular vector of the centered matrix (no scaling), then normalizing to the unit simplex. Includes input validation, optional row renormalization, and a safe fallback when PC1 is degenerate.

Usage

```
compute_L_from_P(P, renormalize_rows = TRUE, tol = 1e-12)
```

Arguments

`P`	Numeric matrix $T \times K$; prior weights per period.
`renormalize_rows`	Logical; if `TRUE` (default), rows of `P` that are within tolerance of summing to 1 are renormalized. Otherwise an error is thrown.
`tol`	Numeric tolerance for simplex checks (default `1e-12`).

Details

Validation: `P` must be a finite, non-negative numeric matrix. Each row must either (i) already sum to 1 within `tol` or (ii) be renormalizable within `tol`. Rows with (near-)zero sums are not renormalizable and raise an error. Missing values are not allowed.

Algorithm (exactly as implemented):

1. Center columns over time: `X <- scale(P, center = TRUE, scale = FALSE)`.
2. Compute SVD: `sv <- svd(X)`.
3. Take the first right singular vector (first column of V matrix); set $l = |v|$.
4. If $\sum l \leq tol$ or PC1 is degenerate, fall back to column means of `P` (over time) and renormalize.
5. Otherwise, $L = l / \sum l$.

Value

Numeric vector L of length K (non-negative, sums to 1). Attributes:

- `"pc1_loadings"`: signed PC1 loadings v.
- `"explained_var"`: fraction of variance explained by PC1.
- `"fallback"`: `TRUE` if column-mean fallback was used.

See Also

spread_likelihood, bayesian_disaggregate

Examples

```
set.seed(123)
T <- 10; K <- 4
P <- matrix(rexp(T*K), nrow = T); P <- P / rowSums(P)
L <- compute_L_from_P(P)
stopifnot(length(L) == K, all(L >= 0), abs(sum(L) - 1) < 1e-12)
```

---

interpretability_score

Interpretability score (structure preservation + plausibility)

Description

Balances two ideas: (i) *preservation* of the average sectoral structure (correlation between `colMeans(P)` y `colMeans(W)`; truncated at 0), and (ii) *plausibility* of relative changes $|W\_bar - P\_bar|/(P\_bar + \varepsilon)$, summarized by the 90th percentile (or the maximum). The score is

$$0.6 \cdot \text{preservation} + 0.4 \cdot \frac{1}{1 + 2 \cdot \text{change}}.$$

Usage

```
interpretability_score(P, W, use_q90 = TRUE)
```

Arguments

`P`	Prior matrix ($T \times K$).
`W`	Posterior matrix ($T \times K$).
`use_q90`	If `TRUE` (default), use 90th percentile of relative changes; if `FALSE`, use the maximum.

Value

Scalar interpretability score in $[0, 1]$.

Examples

```
T <- 6; K <- 5
P <- matrix(runif(T*K), T); P <- P/rowSums(P)
W <- matrix(runif(T*K), T); W <- W/rowSums(W)
interpretability_score(P, W)
```

log_enable *Enable logging at a specific level*

Description

Sets the package-wide logging verbosity.

Usage

```
log_enable(level = "INFO")
```

Arguments

level	Character scalar. One of "TRACE", "DEBUG", "INFO", "WARN", "ERROR".

Value

No return value, called for side effects

numerical_stability_exp *Numerical stability (exponential penalty)*

Description

Penalizes numerical issues in W: (i) deviation of each row sum from 1, measured by the mean absolute deviation, and (ii) count of negative entries. The score is $\exp(-(a\,\mathrm{msd} + b\,\#\mathrm{neg}))$.

Usage

```
numerical_stability_exp(W, a = 1000, b = 10)
```

Arguments

W	Posterior matrix ($T \times K$).
a	Sensitivity for row-sum deviation (default 1000).
b	Sensitivity for negative counts (default 10).

Value

Scalar numerical stability score in $[0, 1]$.

Examples

```
W <- matrix(runif(20), 5); W <- W/rowSums(W)
numerical_stability_exp(W)
```

`posterior_adaptive` *Adaptive posterior based on sector volatility*

Description

Ajusta el *mixing* por sector según la volatilidad histórica del prior:

$$\phi_k = \min\left(\frac{\sigma_k/\mu_k}{\overline{\sigma/\mu}}, 0.8\right)$$

, donde σ_k/μ_k es la desviación estándar relativa de la columna k. Sectores más volátiles reciben más peso de LT. Devuelve $W = (1-\phi) \odot P + \phi \odot LT$ renormalizado por filas.

Usage

```
posterior_adaptive(P, LT)
```

Arguments

`P`	Prior matrix ($T \times K$); filas no negativas.
`LT`	Likelihood matrix ($T \times K$); filas no negativas.

Value

Posterior matrix W ($T \times K$), filas suman 1.

See Also

`posterior_weighted`, `posterior_multiplicative`, `posterior_dirichlet`

Examples

```
set.seed(1)
T <- 8; K <- 5
P  <- matrix(runif(T*K), T);  P  <- P  / rowSums(P)
LT <- matrix(runif(T*K), T); LT <- LT / rowSums(LT)
W <- posterior_adaptive(P, LT)
```

`posterior_dirichlet` *Dirichlet-conjugate posterior (analytical mean)*

Description

Interpreta P y LT como proporciones convertidas a pseudo-cuentas con concentración total $\alpha_{base} = 1/\gamma$. El posterior medio es $W = (P\,\alpha_{base} + LT\,\alpha_{base})/\mathrm{rowSums}(\cdot)$.

Usage

```
posterior_dirichlet(P, LT, gamma = 0.1)
```

Arguments

`P`	Prior matrix ($T \times K$); filas no negativas que suman 1 (o cercanas).
`LT`	Likelihood matrix ($T \times K$); filas no negativas que suman 1 (o cercanas).
`gamma`	Positive uncertainty factor (default `0.1`); menor γ implica mayor concentración (más "seguro").

Value

Posterior matrix W ($T \times K$), filas suman 1.

See Also

`posterior_weighted`, `posterior_multiplicative`, `posterior_adaptive`

Examples

```
T <- 6; K <- 3
P  <- matrix(runif(T*K), T);  P  <- P  / rowSums(P)
LT <- matrix(runif(T*K), T); LT <- LT / rowSums(LT)
W <- posterior_dirichlet(P, LT, gamma = 0.1)
```

`posterior_multiplicative`
Multiplicative posterior (Hadamard product + renormalization)

Description

Computes $W \propto P \odot LT$ (producto elemento a elemento) y renormaliza cada fila a la simplex. Útil cuando prior y likelihood deben reforzarse mutuamente.

Usage

```
posterior_multiplicative(P, LT)
```

Arguments

P	Prior matrix ($T \times K$); filas no negativas.
LT	Likelihood matrix ($T \times K$); filas no negativas.

Value

Posterior matrix W ($T \times K$), filas suman 1.

See Also

posterior_weighted, posterior_dirichlet, posterior_adaptive

Examples

```
T <- 4; K <- 4
P  <- matrix(runif(T*K), T); P  <- P / rowSums(P)
LT <- matrix(runif(T*K), T); LT <- LT / rowSums(LT)
W <- posterior_multiplicative(P, LT)
```

---

posterior_weighted *Weighted-average posterior (convex combination)*

Description

Computes $W = \lambda P + (1 - \lambda)\, LT$ row-wise and renormalizes each row to the unit simplex (suma 1). Expects matrices no negativas con las mismas dimensiones $T \times K$.

Usage

```
posterior_weighted(P, LT, lambda = 0.7)
```

Arguments

P	Prior matrix ($T \times K$); filas no negativas.
LT	Likelihood matrix ($T \times K$); filas no negativas.
lambda	Mixing weight in $[0, 1]$ (default 0.7).

Value

Posterior matrix W ($T \times K$), filas suman 1.

See Also

posterior_multiplicative, posterior_dirichlet, posterior_adaptive

Examples

```
T <- 5; K <- 3
P  <- matrix(runif(T*K), T); P  <- P / rowSums(P)
LT <- matrix(runif(T*K), T); LT <- LT / rowSums(LT)
W <- posterior_weighted(P, LT, 0.6)
stopifnot(all(abs(rowSums(W)-1) < 1e-12))
```

---

read_cpi *Read CPI data from an Excel file*

Description

Loads and normalizes a CPI time series from an Excel worksheet. The function detects the date/year column and the CPI/value column by pattern-matching on lower-cased header names, parses localized numerics (via `to_num_commas()`), collapses duplicate years by averaging, and returns a clean, sorted data frame.

Usage

```
read_cpi(path_cpi)
```

Arguments

`path_cpi`	Character path to the CPI Excel file.

Details

Column detection. Headers are lower-cased and matched with:

- Date/year: patterns `"date|fecha|year|anio|ano"`.
- CPI/value: patterns `"cpi|indice|price"`.

If either column cannot be identified, the function errors.

Cleaning.

- Year is extracted as the first 4 digits of the date-like column.
- CPI is parsed with `to_num_commas()` (handles commas/thousands).
- `NA` rows are dropped; duplicates in `Year` are averaged.
- Output is sorted by `Year` ascending.

Value

A `data.frame` with two columns:

- `Year` (`integer`)
- `CPI` (`numeric`)

See Also

`read_weights_matrix`, `bayesian_disaggregate`

Examples

```
# Create a temporary Excel file with sample CPI data
temp_file <- tempfile(fileext = ".xlsx")
df_sample <- data.frame(
  Fecha = c("2019-01-01", "2020-01-01", "2021-01-01", "2022-01-01"),
  Indice = c(95.5, 100.0, 103.2, 108.7)
)
openxlsx::write.xlsx(df_sample, temp_file)

# Read the CPI data
df <- read_cpi(temp_file)
print(df)

# Verify structure
stopifnot(
  is.data.frame(df),
  all(c("Year", "CPI") %in% names(df)),
  nrow(df) == 4
)

# Clean up
unlink(temp_file)
```

`read_weights_matrix` *Read a weights matrix from an Excel file*

Description

Loads a sector-by-year weight table, normalizes weights to the simplex per year, and returns a list with the $T \times K$ prior matrix `P`, the sector names, and the year vector. The first column is assumed to contain sector names (renamed to `Industry`); all other columns are treated as years.

Usage

```
read_weights_matrix(path_weights)
```

Arguments

`path_weights` Character path to the weights Excel file.

Details

Expected layout. One sheet with:

- First column: sector names (any header; renamed to `Industry`).
- Remaining columns: years; the function extracts a 4-digit year from each header using `stringr::str_extract(Year, "\\d{4}")`.

Values are parsed with `to_num_commas()`, missing rows are dropped, and weights are normalized within each year to sum to 1. Any absent (sector, year) entry becomes 0 when pivoting wide. Finally, rows are re-normalized with `row_norm1()` for numerical safety.

Safeguards.

- Rows with all-missing/zero after parsing are dropped by the filters.
- If no valid year columns are found, the function errors.

Value

A `list` with:

`P` $T \times K$ numeric matrix of prior weights (rows sum to 1).

`industries` Character vector of sector names (length K).

`years` Integer vector of years (length T).

See Also

`read_cpi`, `bayesian_disaggregate`

Examples

```
# Create a temporary Excel file with sample weights
temp_file <- tempfile(fileext = ".xlsx")
df_sample <- data.frame(
  Sector = c("Agriculture", "Manufacturing", "Services", "Construction"),
  "2019" = c(0.20, 0.35, 0.30, 0.15),
  "2020" = c(0.18, 0.37, 0.32, 0.13),
  "2021" = c(0.17, 0.38, 0.33, 0.12),
  "2022" = c(0.16, 0.39, 0.34, 0.11),
  check.names = FALSE
)
openxlsx::write.xlsx(df_sample, temp_file)

# Read the weights matrix
w <- read_weights_matrix(temp_file)

# Inspect structure
str(w)
```

```
print(w$P)

# Verify properties
stopifnot(
  is.matrix(w$P),
  nrow(w$P) == 4,  # 4 years
  ncol(w$P) == 4,  # 4 sectors
  all(abs(rowSums(w$P) - 1) < 1e-10),  # rows sum to 1
  length(w$industries) == 4,
  length(w$years) == 4
)

# Clean up
unlink(temp_file)
```

`run_grid_search` *Run grid search for parameter optimization (parallel PSOCK)*

Description

Run grid search for parameter optimization (parallel PSOCK)

Usage

```
run_grid_search(
  path_cpi,
  path_weights,
  grid_df,
  n_cores = parallel::detectCores() - 1
)
```

Arguments

`path_cpi`	Path to CPI Excel file
`path_weights`	Path to weights Excel file
`grid_df`	Data frame with parameter combinations (one row = one config)
`n_cores`	Number of cores for parallel processing

Value

Data frame with results for all parameter combinations (ordered by composite desc)

save_results *Save disaggregation results to disk*

Description

Writes CSV extracts and a single Excel workbook with the key outputs from `bayesian_disaggregate`.

Usage

```
save_results(res, out_dir)
```

Arguments

`res`	A result object returned by `bayesian_disaggregate()`.
`out_dir`	Directory where files will be written. Created if missing.

Value

(Invisibly) the path to the Excel file written.

Examples

```
# Usar archivos de ejemplo incluidos en el paquete
cpi_file <- system.file("extdata", "CPI.xlsx",
                        package = "BayesianDisaggregation")
weights_file <- system.file("extdata", "WEIGHTS.xlsx",
                            package = "BayesianDisaggregation")

if (nzchar(cpi_file) && nzchar(weights_file)) {
  res <- bayesian_disaggregate(cpi_file, weights_file)
  # Use tempdir() for CRAN compliance
  save_results(res, file.path(tempdir(), "results"))
}
```

spread_likelihood *Spread a likelihood vector across time with a chosen temporal pattern*

Description

Expands a sectoral likelihood L (length K) into a $T \times K$ matrix by applying a temporal weight profile and then row-normalizing to the simplex.

Usage

```
spread_likelihood(
  L,
  T_periods,
  pattern = c("constant", "recent", "linear", "bell")
)
```

Arguments

L	Numeric vector (length K); sectoral likelihood. It is normalized internally to sum to 1 before spreading.
T_periods	Integer; number of time periods T.
pattern	Temporal pattern for the weights across time; one of "constant", "recent", "linear", "bell".

Details

Given L and a non-negative time-weight vector w_t, the function replicates L across rows and applies w elementwise, then *row-normalizes* using row_norm1():

$$LT_{t,k} \propto w_t \cdot L_k, \qquad \sum_k LT_{t,k} = 1 \; \forall t.$$

Patterns:

- "constant": $w_t = 1$.
- "recent": linearly increasing in t (more weight to later periods).
- "linear": affine ramp from first to last period.
- "bell": symmetric bell-shaped profile centered at $T/2$.

Value

Numeric matrix $T \times K$; each row sums to 1.

See Also

compute_L_from_P, bayesian_disaggregate

Examples

```
set.seed(1)
K <- 5; T <- 8
L <- runif(K); L <- L / sum(L)
LT <- spread_likelihood(L, T, "recent")
stopifnot(nrow(LT) == T, ncol(LT) == K, all(abs(rowSums(LT) - 1) < 1e-12))
```

stability_composite *Composite stability score (numerical and temporal)*

Description

Convex combination of numerical and temporal stability:

$$0.6 \cdot \text{numerical\_stability\_exp}(W) + 0.4 \cdot \text{temporal\_stability}(W).$$

Usage

```
stability_composite(W, a = 1000, b = 10, kappa = 50)
```

Arguments

W	Posterior matrix ($T \times K$).
a	Sensitivity for row-sum deviation in numerical part (default 1000).
b	Sensitivity for negatives in numerical part (default 10).
kappa	Sensitivity for temporal smoothness (default 50).

Value

Scalar composite stability score in $[0, 1]$.

See Also

numerical_stability_exp, temporal_stability

Examples

```
W <- matrix(runif(20), 5); W <- W/rowSums(W)
stability_composite(W)
```

temporal_stability *Temporal stability (smoothness over time)*

Description

Rewards smooth posteriors by penalizing average absolute period-to-period changes per sector. Maps to $[0, 1]$ via $1/(1 + \kappa \cdot \text{mv})$.

Usage

```
temporal_stability(W, kappa = 50)
```

Arguments

W	Posterior matrix ($T \times K$).
kappa	Sensitivity to average absolute change (default 50).

Value

Scalar temporal stability score in $[0, 1]$. If $T < 2$, returns 0.8.

Examples

```
W <- matrix(runif(30), 6); W <- W/rowSums(W)
temporal_stability(W, kappa = 40)
```

Index