Package: MMEFM (via r-universe)

November 4, 2026

Type Package

Title Multilevel Main Effects Matrix Factor Model Tools

Version 0.1.1

Description Provides estimation, rank selection, and simulation for grouped matrix-valued time series using the Multilevel Main Effects Matrix Factor Model. The model separates global and local grand means, row and column main effects, and bilinear interaction components, allowing nonstationary main effects under suitable conditions. A circular-shift bootstrap is used to detect a shared global common component and screen participating groups on a stationary segment. The underlying methodology is developed in Liu, Lam and Cen (2026) <doi:10.48550/arXiv.2610.18335>.

License MIT + file LICENSE

URL https://github.com/kx-liu/MMEFM

BugReports https://github.com/kx-liu/MMEFM/issues

Depends R (>= 4.1.0)

Imports parallel, stats

Suggests testthat (>= 3.2.0), knitr, rmarkdown

VignetteBuilder knitr

Encoding UTF-8

Roxygen list(markdown = TRUE)

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

Repository https://kx-liu.r-universe.dev

Date/Publication 2026-10-21 15:58:50 UTC

RemoteUrl https://github.com/kx-liu/mmefm

RemoteRef HEAD

RemoteSha ff2da4f848fac04ebea05732ab323859e52d512a

Contents

detect_MMEFM_global 2
est_MMEFM 4
fitted.mmefm_fit 7
gen_MMEFM 8
gen_piecewise_arfima 11
print.mmefm_global_detection 13
residuals.mmefm_fit 14
select_MMEFM_rank 15
summary.mmefm_fit 17

Index **18**

`detect_MMEFM_global` *Detect and screen global common factors in MMEFM*

Description

Test whether a global common component is shared by at least two groups, and screen the groups that share it. Group statistics are calibrated by a circular-shift bootstrap.

Usage

```
detect_MMEFM_global(
  Xt,
  rank = NULL,
  B = 199L,
  alpha = 0.05,
  reference_group = 1L,
  K0 = 20L,
  max_iter = 20L,
  tol = 1e-04,
  seed = 2026L,
  verbose = FALSE,
  parallel = FALSE,
  num.cores = 4L
)
```

Arguments

`Xt`	A list of finite `T x p_m x q_m` arrays as in `est_MMEFM()`, or an `mmefm_fit`. Arrays are reduced to direct additive residuals `check_Y`; a fit supplies its stored `check_Y`.
`rank`	NULL to select ranks for array input or use a fit's ranks. Alternatively, supply a complete eight-field rank list or `mmefm_rank` as in `est_MMEFM()`. Supplied ranks override fit ranks. Rank values are validated, and any rank-object diagnostics are retained.

B	Positive whole-number bootstrap size. The default is 199. Values such as 3 are suitable only for illustrative examples, not reliable calibration.
alpha	Nominal calibration level strictly between zero and one, used to determine the bootstrap cutoff.
reference_group	Integer index from 1 through M identifying the group held fixed during bootstrap shifts.
K0	Positive whole number of random half-panel column directions per group for initializing the iterative projection procedure (Algorithm 1).
max_iter	Positive whole-number limit on Algorithm 1 update iterations.
tol	Finite positive tolerance for the sum of loading-space discrepancies in Algorithm 1.
seed	Nonnegative whole number within R's integer range controlling projection directions and bootstrap shifts. The caller's RNG state is restored on return or error.
verbose	One nonmissing logical indicating whether to print Algorithm 1 progress.
parallel	One nonmissing logical. FALSE runs the bootstrap serially. TRUE uses PSOCK workers on Windows, macOS, and Linux when B and num.cores both exceed one.
num.cores	Positive whole-number maximum worker count. At most min(B, num.cores) workers are used; one worker or B = 1 gives serial execution. Stay within the available resource allocation.

Details

Detection concerns global common matrix-factor interactions, not global main effects. Each group's direct residual is divided by its RMS scale without further centering. Initial global loading spaces are re-estimated from these standardized residuals, including for fit input.

The group statistic $S_{G,m}$ measures cross-group dependence relative to within-group variation. Its numerator sums the largest squared cross-group moments of projected global-core coordinates over other groups. Its denominator is the square of the largest within-group projected-coordinate mean square, using total global plus local common loading dimensions. These moments are not additionally demeaned over time.

The existence test uses the second-largest group statistic because a global common component must involve at least two groups. The bootstrap holds the reference group fixed and independently circularly shifts each other group's time series by a uniformly drawn offset from 0 through T-1. Loading spaces and statistics are re-estimated at fixed ranks for each replication. This retains within-group temporal dependence while disrupting synchronous cross-group alignment.

One Type-8 empirical (1 - alpha) quantile of bootstrap second-largest statistics supplies the common cutoff. Existence requires the observed second-largest statistic to strictly exceed it. Final screening retains groups strictly above the same cutoff only if existence is detected and at least two groups qualify; otherwise the selected set is empty.

Whole-sample circular shifts assume a single stationary segment. They are not justified across arbitrary structural breaks or piecewise nonstationary series. The model's allowance for nonstationary main effects does not remove this bootstrap restriction.

Invalid residual scales, denominators, and numerical failures error without repair or discarded bootstrap draws. Bootstrap errors identify the replication. Projection non-convergence retains final iterates and warns for the observed statistic or, in aggregate, for bootstrap replications. The caller's RNG state is restored on success or error; automatic rank selection uses a separate scoped seed.

Parallel execution requires the current installed MMEFM namespace. Serial and parallel calls use matching replication-specific randomness under the same seed; workers are cleaned up on completion or error. Process startup and memory overhead can make small bootstraps slower. Avoid nested worker pools when replications are already parallelized externally.

Value

An `mmefm_global_detection` object with `call`, group `statistic`, observed `second_largest`, `cutoff`, and logical `exists`. `initial_selected` marks groups exceeding the cutoff; `selected` applies the existence and at-least-two-groups requirements. Group vectors preserve optional names.

`scale` contains group residual RMS values, and `rank` contains the validated `mmefm_rank`. `bootstrap` contains `B`, `alpha`, `reference_group`, and the length-`B` vector `second_largest` used for calibration. Loading estimates and residual arrays are not returned.

See Also

`est_MMEFM()`, `select_MMEFM_rank()`, `gen_MMEFM()`

Examples

```
rank <- list(r1 = 1L, l1 = 1L, r2 = rep(1L, 3), l2 = rep(1L, 3),
             kr = 1L, kc = 1L, kr_m = rep(1L, 3), kc_m = rep(1L, 3))
simulation <- gen_MMEFM(TT = 40L, p = c(a = 6L, b = 7L, c = 8L), q = c(a = 7L, b = 8L, c = 6L),
                        rank = rank, burn = 20L, filter_length = 40L, seed = 2026L)
# B = 3 illustrates the call; it does not provide reliable calibration.
detection <- detect_MMEFM_global(simulation$Xt, rank = simulation$rank, B = 3L,
                                 K0 = 8L, max_iter = 50L, parallel = FALSE, seed = 2026L)
detection
detection$selected
```

`est_MMEFM` *Estimate a Multilevel Main Effects Matrix Factor Model*

Description

Estimate global and local grand means, row and column main effects, and common matrix-factor components in grouped matrix-valued time series.

Usage

```
est_MMEFM(
  Xt,
  rank = NULL,
  alignment_method = c("VanLoan", "Procrustes"),
```

```
  refit_method = c("VanLoan", "ALS"),
  K0 = 20L,
  max_iter = 20L,
  tol = 1e-04,
  max_iter_procrustes = 20L,
  tol_procrustes = 1e-04,
  max_iter_als = 20L,
  tol_als = 1e-04,
  lambda = 0,
  seed = 2026L,
  verbose = FALSE
)
```

Arguments

`Xt`	A list of at least two finite numeric arrays with dimensions `T x p_m x q_m`, in time, row, and column order. All groups share `T >= 1`; each spatial dimension is at least two. Group names are optional, but must be nonempty and unique when supplied.
`rank`	NULL for automatic selection, a complete named list containing exactly `r1`, `l1`, `r2`, `l2`, `kr`, `kc`, `kr_m`, and `kc_m`, or an `mmefm_rank` object. Global ranks are positive scalar whole numbers; local ranks are positive whole-number vectors in group order. Named local vectors are matched to named `Xt` groups. Each rank is smaller than its spatial dimension. Under IC1, row sums `r1 + r2[m]` and `kr + kr_m[m]` are at most `p_m - 1`; column sums `l1 + l2[m]` and `kc + kc_m[m]` are at most `q_m - 1`. Equality is allowed.
`alignment_method`	Method for aligning group-specific global common-factor coordinates: `"VanLoan"` (default) or `"Procrustes"`.
`refit_method`	Method for refitting global common loadings: `"VanLoan"` (default) or `"ALS"`. This choice is independent of `alignment_method`.
`K0`	Positive whole number of random half-panel column directions per group for initializing the iterative projection procedure (Algorithm 1).
`max_iter`	Positive whole-number limit on Algorithm 1 update iterations.
`tol`	Finite positive tolerance for the sum of loading-space discrepancies in Algorithm 1.
`max_iter_procrustes`	Positive whole-number limit on Procrustes alignment iterations.
`tol_procrustes`	Finite positive tolerance for relative change in the Procrustes alignment score.
`max_iter_als`	Positive whole-number limit on alternating least-squares (ALS) refitting iterations.
`tol_als`	Finite positive tolerance for relative change in the squared-residual objective during ALS refitting.
`lambda`	Finite nonnegative ridge parameter for main-effect Gram systems only. Zero, the default, gives the unregularized manuscript estimator.

seed	Nonnegative whole number within R's integer range controlling random projection directions. Results are reproducible under the same seed and RNG kind. The caller's RNG state is restored on return or error.
verbose	One nonmissing logical indicating whether to print Algorithm 1 progress.

Details

Direct moment estimators first recover the combined grand mean and centered row and column main effects in each group. Cross-group moments and projections separate global and local main-effect structures. Shared main-effect scores are estimated by pooling temporal information, and their global loadings are refitted.

The direct additive estimates are subtracted to form `check_Y`. Iterative projection estimates initial global common loading spaces from these residuals. Projections then recover local common loadings and factors. After removing local common components, group-specific global common-factor coordinates are aligned and pooled into one shared factor series. Global common loadings are refitted to that series.

Van Loan alignment approximates the between-group coordinate mapping by a Kronecker product. Van Loan refitting similarly approximates the unconstrained fitted loading product, with the aligned global core held fixed. Optional Procrustes alignment alternates orthogonal row and column transformations. Optional ALS refitting alternates row and column loading updates and updates the shared global core after each sweep.

The direct main-effect estimates are not the final low-rank main-effect reconstruction. `fitted.mmefm_fit()` uses global and local factor scores and loadings, including refitted global main loadings. `check_Y` is an intermediate residual, not the final model residual; use `residuals.mmefm_fit()` for the latter.

With `rank = NULL`, `select_MMEFM_rank()` is followed by a fresh known-rank fit using the same seed. Supplying that selected rank explicitly gives the same statistical estimates. Both stochastic stages preserve the caller's RNG state, including on error.

Global main-effect ranks must not exceed T. Van Loan alignment or refitting additionally requires `kr * kc <= T`. These bounds do not guarantee nonsingular Gram systems. Singular unregularized systems error without numerical fallback or rank repair. Iteration-limit warnings retain the final iterates; inspect `fit$convergence`.

Value

An object of class `mmefm_fit` with `call`, `dimensions`, `rank`, `main_effect`, `common_component`, `check_Y`, and `convergence`. `rank` is an `mmefm_rank`; observed arrays are not stored separately.

`main_effect` contains `direct` estimates (`c_hat`, `a_hat`, `b_hat`), plus `global` and `local` grand means, loadings, and factor scores. Initial global loadings `A1_hat` and `B1_hat` are retained alongside refitted `A1_tilde` and `B1_tilde`. Local loadings are `A2_hat` and `B2_hat`; both levels store `alpha_hat` and `beta_hat`.

`common_component` contains `global`, `local`, and method-specific `alignment` estimates. Global `Q_hat` and `J_hat` are initial loading estimates; `Q_tilde` and `J_tilde` are refitted loadings. `G_hat` is the aligned, pooled core. `G_tilde` equals `G_hat` for Van Loan refitting and is updated during ALS. Local estimates are `R_hat`, `C_hat`, and `F_hat`.

`check_Y` contains the direct additive residual arrays. `convergence` records iteration counts and convergence indicators for final `projection`, `procrustes`, and `als` stages; inactive methods are

NULL. summary.mmefm_fit() reports dimensions, ranks, and method choices, not convergence diagnostics.

See Also

gen_MMEFM(), select_MMEFM_rank(), detect_MMEFM_global(), fitted.mmefm_fit(), residuals.mmefm_fit()

Examples

```
rank <- list(r1 = 1L, l1 = 1L, r2 = rep(1L, 3), l2 = rep(1L, 3),
             kr = 1L, kc = 1L, kr_m = rep(1L, 3), kc_m = rep(1L, 3))
simulation <- gen_MMEFM(TT = 40L, p = c(a = 6L, b = 7L, c = 8L), q = c(a = 7L, b = 8L, c = 6L),
                        rank = rank, burn = 20L, filter_length = 40L, seed = 2026L)
fit <- est_MMEFM(simulation$Xt, rank = simulation$rank, K0 = 8L, max_iter = 50L, seed = 2026L)
summary(fit)
fit$convergence
lapply(fitted(fit), dim)
lapply(fitted(fit, component = "global_common"), dim)
vapply(residuals(fit), function(x) sqrt(mean(x^2)), numeric(1L))
```

---

fitted.mmefm_fit *Reconstruct fitted MMEFM components*

Description

Recover the estimated signal or one of its global and local components from an MMEFM fit.

Usage

```
## S3 method for class 'mmefm_fit'
fitted(
  object,
  component = c("all", "main", "global_main", "local_main", "global_common",
    "local_common"),
  ...
)
```

Arguments

object	An mmefm_fit returned by est_MMEFM().
component	Component to reconstruct. "all" (default) sums all four structured components; "main" sums global and local main effects. "global_main" and "local_main" include their respective grand means, row main effects, and column main effects. "global_common" and "local_common" give the corresponding matrix-factor interactions.
...	Further arguments, currently unused.

Details

Global main effects use `A1_tilde` and `B1_tilde` with the global `alpha_hat` and `beta_hat` scores. Local main effects use `A2_hat` and `B2_hat` with group-local scores. These final factor representations differ from the direct moment estimates in `object$main_effect$direct`.

Global common components use `Q_tilde`, `G_tilde`, and `J_tilde`. Local common components use `R_hat`, `F_hat`, and `C_hat`. `check_Y` is an intermediate residual and is not an additional fitted component.

Value

A list of `T x p_m x q_m` arrays preserving the data's group names and dimensions.

See Also

`est_MMEFM()` for an executable example, `residuals.mmefm_fit()`, `summary.mmefm_fit()`

---

gen_MMEFM *Simulate a Multilevel Main Effects Matrix Factor Model*

Description

Generate grouped matrix-valued time series with global and local grand means, row and column main effects, common matrix-factor interactions, and error. Return the data and generating quantities for comparison with fitted components.

Usage

```
gen_MMEFM(
  TT,
  p,
  q,
  rank,
  strength = NULL,
  error_rank = NULL,
  memory = NULL,
  phi = 0.6,
  error_phi = 0.2,
  error_loading_zero_prob = 0.95,
  innovation = c("Gaussian", "Student-t"),
  df = 6,
  burn = 200L,
  filter_length = NULL,
  active_global = NULL,
  seed = 2026L,
  store_components = FALSE
)
```

Arguments

TT	Positive whole-number time dimension within R's integer range.
p, q	Whole-number row and column dimension vectors of equal length M >= 2, with entries at least two. Both are unnamed or have unique nonempty matching group names; q is matched to p order.
rank	Complete eight-field rank list or mmefm_rank; see est_MMEFM(). Nominal model ranks are positive. IC1 centering requires r1 + r2[m] and kr + kr_m[m] to be at most p_m - 1, and l1 + l2[m] and kc + kc_m[m] at most q_m - 1. Equality is allowed. The generator additionally requires each global/local main-score rank sum to be at most TT to construct temporal orthogonality. Supplied rank diagnostics are discarded.
strength	NULL or a named list with any subset of main_global, main_local, common_global, and common_local. Omitted or NULL fields independently default to strength 1. All nested lists and vectors must be unnamed. Values are finite in (0.5, 1]; 1 gives strong loadings. See Details for the positional structure.
error_rank	NULL or a named list with any subset of ker, kec, ker_m, and kec_m. The first two are scalar shared error row/column ranks; the last two are length-M local error row/column rank vectors. Omitted fields default to 2. Values are nonnegative whole numbers not exceeding the corresponding spatial dimensions. Named local vectors match named p/q groups; otherwise matching is positional. Zero in either direction gives a zero factor-error component while retaining zero-rank tensor dimensions.
memory	NULL for two zero-memory segments for all latent components, or exactly a list with mu_global, mu_local, alpha, beta, G, and F. A basic specification is a numeric vector of finite d < 0.5, or list(d = ..., seg_lens = ...) with positive whole segment lengths summing to TT. Numeric specifications split TT equally, with the remainder in the last segment. Means and G use basic specifications. F uses a basic specification or a group list of them. Alpha/beta use a basic specification or list(global = <basic>, local = <basic or group list>). Local mean innovations share one specification because group demeaning mixes them. No lower bound on d is imposed.
phi, error_phi	Stable AR coefficient vectors for latent and error processes, respectively, in lag order. A scalar specifies AR(1) with absolute value below one. For AR(p), the roots of 1 - phi[1]*z - ... - phi[p]*z^p (and the analogous error polynomial) must lie strictly outside the unit circle. Individual coefficients may exceed one in magnitude.
error_loading_zero_prob	Probability in [0, 1) of setting each Gaussian error-loading entry to zero independently.
innovation	Gaussian (default) or raw Student-t temporal innovations. Spatial loadings and error-scale entries always use Gaussian draws.
df	Finite positive Student-t degrees of freedom. Raw draws are not variance-standardized. Simulation accepts any positive df, which need not meet the manuscript's moment assumptions. The value is retained in metadata for either innovation family.

`burn` Nonnegative whole-number burn-in within R's integer range.

`filter_length` NULL for `min(2000, max(200, 5*TT))`, or a nonnegative whole fractional/AR impulse-response truncation length within R's integer range. See `gen_piecewise_arfima()` for filtering and warmup conventions.

`active_global` NULL for all groups, a nonmissing logical length-M vector in group order, or unique whole group indices, including empty indices. Zero or at least two active groups are allowed; a singleton errors. Inactive global common loadings Q/J are zeroed after their ordinary draws. Nominal positive ranks and G are unchanged.

`seed` Nonnegative whole-number seed within R's integer range. The caller's RNG state is preserved on success and error; RNG kind is unchanged.

`store_components`
One nonmissing logical. TRUE stores seven full-array signal/error component lists in addition to truth loadings and factors. FALSE, the default, omits these arrays.

Details

The model separates shared temporal variation from group-local variation, and additive main effects from row-column interactions. `est_MMEFM()` fits the same component structure. Generating loadings and fitted loadings need not agree entrywise because factor coordinates are not unique.

`strength$main_global` is a length-two vector for A1 and B1, shared across groups. `strength$main_local` is a length-M list of length-two vectors for A2 and B2. Main strengths are uniform within each loading matrix, as in ME1. Changing global main strengths does not change the defaults for local main strengths.

`strength$common_global` and `strength$common_local` are length-M lists in p order. Each group entry is exactly `list(row_strengths, column_strengths)`. Global entries control Q/J with lengths kr/kc; local entries control R/C with lengths `kr_m[m]`/`kc_m[m]`. Each side accepts one scalar, broadcast to its rank, or one value per loading column. This permits heterogeneous strengths as in L1. Groups are positional after q is matched to p; nested group or side names are not accepted.

The error ranks correspond to E1's $k_{e,r}$, $k_{e,c}$, $k_{e,r}^{(m)}$, and $k_{e,c}^{(m)}$. NULL and an empty list give rank 2 on every side. Partial lists retain defaults for omitted fields. These nuisance ranks do not have the model's IC1-centered combined-rank restrictions.

A loading column in spatial dimension n with strength ξ is generated from Gaussian entries scaled by $n^{-(1-\xi)/2}$ and then centered under IC1. Its squared norm is of order n^{ξ}. Global and local spatial loadings are not artificially orthogonalized. Local grand means sum to zero across groups at each time. Local row and column main-score series are projected off the corresponding global temporal span, enforcing sample IC2 orthogonality up to roundoff. Singular Gram systems error without ridge or fallback.

Memory specifications apply segment-specific fractional filters to stable AR processes. Zero memory gives direct AR recursion. See `gen_piecewise_arfima()` for the retained-sample warmup and truncation conventions. Piecewise memory can change temporal dependence; it does not imply whole-sample stationarity.

The error contains shared and local factor processes plus cell-specific error. Error loadings are independent sparse Gaussian matrices across groups. This finite-sample construction deliberately relaxes the manuscript's E1 cross-group error-loading orthogonality. Error AR processes are not rescaled to unit variance, and cell-specific scale entries are absolute Gaussian draws. Generated data

do not automatically satisfy every asymptotic E1/E2 regularity condition. Nonfinite observations error without repair.

Value

A list with `Xt`, `dimensions`, `rank`, `main_effect`, `common_component`, `error`, `memory`, `innovation`, and `components`.

`Xt` contains `T x p_m x q_m` arrays. `dimensions` contains M, T, p, q, and optional group names. `rank` is an `mmefm_rank` with NULL diagnostics.

`main_effect` contains direct truth `c`, `a`, `b`, global truth `mu`, `A1`, `alpha`, `B1`, `beta`, and local truth `mu`, `A2`, `alpha`, `B2`, `beta`. `common_component` contains global `Q`, `J`, `G`, `active`, and local `R`, `C`, `F`. `error` contains global and local `Q`/`J`/`G` or `R`/`C`/`F` quantities and group `sigma` matrices. Scores and means are time-by-direction matrices; cores are time-by-row-rank-by-column-rank arrays. Group lists preserve optional names.

`memory` contains canonical d/segment-length specifications, with alpha/beta split into global and group-local specifications and F expanded by group. `innovation` contains `family` and `df`. If stored, `components` contains `global_main`, `local_main`, `global_common`, `local_common`, `error_global`, `error_local`, and `error_idiosyncratic`. Their sum recovers `Xt`; otherwise `components` is NULL.

See Also

`gen_piecewise_arfima()`, `est_MMEFM()`, `select_MMEFM_rank()`, `detect_MMEFM_global()`

Examples

```
rank <- list(r1 = 1L, l1 = 1L, r2 = c(1L, 1L), l2 = c(1L, 1L),
             kr = 1L, kc = 1L, kr_m = c(1L, 1L), kc_m = c(1L, 1L))
simulation <- gen_MMEFM(TT = 8L, p = c(a = 4L, b = 5L), q = c(a = 5L, b = 4L), rank = rank,
                        strength = list(main_global = c(0.8, 1),
                          common_global = list(list(0.7, 1), list(0.9, 0.8))),
                error_rank = list(ker = 1L, kec = 2L, ker_m = c(0L, 2L), kec_m = c(1L, 0L)),
                    burn = 4L, filter_length = 8L, seed = 2026L, store_components = TRUE)
lapply(simulation$Xt, dim)
reconstructed <- Reduce(`+`, lapply(simulation$components, function(x) x[[1L]]))
max(abs(reconstructed - simulation$Xt[[1L]]))
```

`gen_piecewise_arfima` *Generate piecewise ARFIMA time series*

Description

Simulate independent ARFIMA(p, d, 0) series with a stable AR short-memory component and segment-specific fractional memory. The memory specification may be shared across series or differ by series.

Usage

```
gen_piecewise_arfima(
  TT,
  n = 1L,
  d_vec = 0,
  phi = 0.5,
  innovation = c("Gaussian", "Student-t"),
  df = 6,
  burn = 200L,
  filter_length = NULL,
  seed = 2026L,
  return_filters = FALSE
)
```

Arguments

TT	Positive whole-number sample length within R's integer range.
n	Positive whole-number number of series within R's integer range. The series dimension is retained when n = 1.
d_vec	A shared numeric vector of finite memory parameters below 0.5, a shared `list(d = ..., seg_lens = ...)`, or a list of length n specifying each series. Numeric vectors split TT approximately equally, with the remainder in the final segment. Explicit segment lengths are positive whole numbers summing to TT. No lower bound on d is imposed.
phi	Nonempty finite AR coefficient vector in lag order: `X[t] = phi[1]*X[t-1] + ... + phi[p]*X[t-p] + innovation[t]`. All roots of `1 - phi[1]*z - ... - phi[p]*z^p` must lie strictly outside the unit circle. A scalar specifies AR(1) with absolute value below one; stable higher-order coefficients may individually exceed one in magnitude.
innovation	Gaussian (default) or raw, unstandardized Student-t innovations.
df	Finite positive Student-t degrees of freedom, used for Student-t draws only. Small df may give infinite variance or violate the manuscript's moment assumptions.
burn	Nonnegative whole-number burn-in within R's integer range.
filter_length	NULL for `min(2000, max(200, 5*TT))`, or a nonnegative whole-number filter truncation length within R's integer range.
seed	Nonnegative whole-number seed within R's integer range. The caller's RNG state and kind are preserved on success and error; repeated calls agree under the same seed and RNG kind.
return_filters	One nonmissing logical. TRUE returns deterministic filter information alongside the series.

Details

Within a segment of memory d, the fractional filter represents $(1 - B)^{-d}$, where B is the lag operator. Zero memory gives an AR process; positive memory below 0.5 introduces long-memory dependence. Changing d between segments need not give a stationary series over the whole sample.

Nonzero memory uses a truncated AR impulse response followed by segment-specific fractional filters. The output segment determines its fractional parameter; the innovation stream is not restarted at a boundary. Entirely zero-memory series use direct AR recursion from zero pre-sample values. Scalar AR(1) uses the corresponding geometric filter or recursion. The retained sample follows `burn + 2*filter_length` warmup observations, and innovations are generated in series-column order.

AR stability concerns only the short-memory component. The bounds on d and df do not by themselves establish every theoretical assumption. Nonfinite generated series error without numerical repair.

Value

With `return_filters = FALSE`, a finite numeric `T x n` matrix. With TRUE, a list containing `series` (the same matrix), supplied `phi`, `filter_length`, `short_filter`, and `series_specs`.

`short_filter` contains AR impulse-response coefficients at lags 0 through `filter_length`. Each of the `n` `series_specs` contains `d`, `seg_lens`, inclusive `seg_start`/`seg_end` indices, and one `fractional_filters` coefficient vector per segment at those same lags. For zero memory, `short_filter` is an inspection representation; simulation uses recursion. Returning metadata consumes no additional random draws.

See Also

`gen_MMEFM()` for grouped matrix-valued simulation

Examples

```
series <- gen_piecewise_arfima(
  TT = 12L, n = 2L, d_vec = c(0, 0.2), phi = c(0.4, -0.1),
  burn = 4L, filter_length = 8L, seed = 2026L
)
dim(series)
```

---

print.mmefm_global_detection

Print global common-factor detection results

Description

Display the second-largest statistic, bootstrap cutoff, calibration level, existence decision, and selected groups.

Usage

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

Arguments

x	An mmefm_global_detection returned by detect_MMEFM_global().
...	Further arguments, currently unused.

Value

The input object invisibly.

See Also

detect_MMEFM_global()

---

residuals.mmefm_fit *Compute residuals of an MMEFM fit*

Description

Subtract the complete final fitted signal from the observed group arrays.

Usage

```
## S3 method for class 'mmefm_fit'
residuals(object, ...)
```

Arguments

object	An mmefm_fit returned by est_MMEFM().
...	Further arguments, currently unused.

Details

The final fitted signal contains low-rank global and local main effects and common components. It differs from the direct additive estimate removed when constructing object$check_Y. Hence check_Y and the final model residual are distinct.

The observations are recovered from the retained direct moments and check_Y; a separate observed-data copy is not needed.

Value

A list of T x p_m x q_m residual arrays preserving group names and dimensions.

See Also

fitted.mmefm_fit(), est_MMEFM() for an executable example

`select_MMEFM_rank` *Select global and local MMEFM ranks*

Description

Estimate the loading ranks for global and local row main effects, column main effects, and common components. Selection uses the manuscript's unperturbed consecutive eigenvalue-ratio rule.

Usage

```
select_MMEFM_rank(
  Xt,
  K0 = 20L,
  max_iter = 20L,
  tol = 1e-04,
  seed = 2026L,
  verbose = FALSE
)
```

Arguments

`Xt`	A list of at least two finite numeric arrays with dimensions `T x p_m x q_m`, in time, row, and column order. All groups share `T >= 1`; each spatial dimension is at least two. Group names are optional, but must be nonempty and unique when supplied.
`K0`	Positive whole number of random half-panel column directions per group for initializing the iterative projection procedure (Algorithm 1).
`max_iter`	Positive whole-number limit on Algorithm 1 update iterations.
`tol`	Finite positive tolerance for the sum of loading-space discrepancies in Algorithm 1.
`seed`	Nonnegative whole number within R's integer range controlling random projection directions. Results are reproducible under the same seed and RNG kind. The caller's RNG state is restored on return or error.
`verbose`	One nonmissing logical indicating whether to print Algorithm 1 progress.

Details

The eight ranks correspond to global/local row main effects (`r1`, `r2`), column main effects (`l1`, `l2`), row common loadings (`kr`, `kr_m`), and column common loadings (`kc`, `kc_m`). Ranks are positive in this version.

For a local rank, the selected index minimizes λ_{j+1}/λ_j for the group's relevant moment matrix. For a global rank, eigenvalues are first pooled by taking their maximum across groups separately at each index. Consecutive ratios of that pooled spectrum are then minimized. The search uses every available ratio: through the smallest group dimension minus one for global ranks, and the group's dimension minus one for local ranks. A zero denominator gives an infinite ratio; ties select the first minimizing index.

Global main-effect ranks use cross-group moments. Local main-effect ranks use moments projected off the estimated global spaces. Global common ranks are updated within Algorithm 1; local common ranks use residual moments after projection off the global common spaces. Projection convergence requires unchanged global common ranks and a loading-space discrepancy below `tol`. If `max_iter` is reached, the final iterate is used with a warning.

The ranks are selected independently and then checked jointly. Under IC1, `r1 + r2[m]` and `kr + kr_m[m]` must not exceed `p_m - 1`; `l1 + l2[m]` and `kc + kc_m[m]` must not exceed `q_m - 1`. Equality is allowed. Infeasible selections error without capping or modifying ranks. Small-sample selection is not guaranteed to recover the true ranks.

Integer controls must be within R's integer range. Moment matrices are symmetrized before eigendecomposition. Roundoff-size negative eigenvalues are set to zero; materially indefinite matrices error. No eigenvalue perturbation is added.

Value

An `mmefm_rank` object containing the eight ranks and `diagnostics`. Global ranks `r1`, `l1`, `kr`, and `kc` are scalar integers. Local ranks `r2`, `l2`, `kr_m`, and `kc_m` are integer vectors retaining group names and order.

`diagnostics` has the same eight names. Global entries contain group `eigenvalues`, indexwise `pooled_eigenvalues`, and `ratios`. Local entries are group lists containing `eigenvalues` and `ratios`. Eigenvectors and iteration traces are not returned. The rank object can be supplied directly to `est_MMEFM()`.

See Also

`est_MMEFM()`, `gen_MMEFM()`

Examples

```
# Exact additive and common signals with orthogonal temporal scores.
H <- matrix(1, 1, 1)
for (i in seq_len(4L)) {
  H <- rbind(cbind(H, H), cbind(H, -H))
}
u <- c(1, 1, -2)
v <- c(1, -1, 0)
Xt <- setNames(vector('list', 3L), c('a','b','c'))
for (m in seq_along(Xt)) {
  Xt[[m]] <- array(0, c(16L, 3L, 3L))
  for (t in seq_len(16L)) {
    row <- 3 * (m + 1) * H[t, 2L] * u + H[t, 2L + m] * v
    column <- 2 * (m + 2) * H[t, 6L] * u + H[t, 6L + m] * v
    Xt[[m]][t, , ] <- outer(row, rep(1, 3)) + outer(rep(1, 3), column) +
      5 * (m + 1) * (m + 2) * H[t, 10L] * outer(u, u) + H[t, 10L + m] * outer(v, v)
  }
}
selected <- select_MMEFM_rank(Xt, K0 = 8L, max_iter = 50L, seed = 2026L)
selected[c("r1", "l1", "kr", "kc")]
selected$diagnostics$r1$ratios
```

summary.mmefm_fit *Summarize and print an MMEFM fit*

Description

Report the fitted model's dimensions, global and local ranks, alignment method, and refitting method.

Usage

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

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

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

Arguments

object	An mmefm_fit returned by est_MMEFM().
...	Further arguments, currently unused.
x	An mmefm_fit or summary.mmefm_fit object, as appropriate.

Details

The summary omits rank-selection diagnostics and convergence information. Inspect object$rank$diagnostics and object$convergence for these, respectively. Printing a fit displays the same model summary.

Value

summary() returns an object of class summary.mmefm_fit containing call, dimensions, the eight statistical rank values, alignment_method, and refit_method. The print methods return their input invisibly.

See Also

est_MMEFM() for an executable example, fitted.mmefm_fit(), residuals.mmefm_fit()

Index

detect_MMEFM_global, 2
detect_MMEFM_global(), *7*, *11*, *14*

est_MMEFM, 4
est_MMEFM(), *2*, *4*, *7–11*, *14*, *16, 17*

fitted.mmefm_fit, 7
fitted.mmefm_fit(), *6, 7*, *14*, *17*

gen_MMEFM, 8
gen_MMEFM(), *4*, *7*, *13*, *16*
gen_piecewise_arfima, 11
gen_piecewise_arfima(), *10, 11*

print.mmefm_fit *(*summary.mmefm_fit*)*, 17
print.mmefm_global_detection, 13
print.summary.mmefm_fit
*(*summary.mmefm_fit*)*, 17

residuals.mmefm_fit, 14
residuals.mmefm_fit(), *6–8*, *17*

select_MMEFM_rank, 15
select_MMEFM_rank(), *4*, *6, 7*, *11*
summary.mmefm_fit, 17
summary.mmefm_fit(), *7, 8*