Package 'bayesRecon'

April 16, 2026

Type Package

Date 2026-04-16

Title Probabilistic Reconciliation via Conditioning

Version 1.0.1

Maintainer Dario Azzimonti <dario.azzimonti@gmail.com>

Description Provides methods for probabilistic reconciliation of hierarchical forecasts of time series. The available methods include analytical Gaussian reconciliation (Corani et al., 2021) <doi:10.1007/978-3-030-67664-3_13>, MCMC reconciliation of count time series (Corani et al., 2024) <doi:10.1016/j.ijforecast.2023.04.003>, Bottom-Up Importance Sampling (Zambon et al., 2024) <doi:10.1007/s11222-023-10343-y>, methods for the reconciliation of mixed hierarchies (Mix-Cond and TD-cond) (Zambon et al., 2024) <https://proceedings.mlr.press/v244/zambon24a.html>, analytical reconciliation with Bayesian treatment of the covariance matrix (Carrara et al., 2025) <doi:10.48550/arXiv.2506.19554>.

License LGPL (>= 3)

Depends R (>= 4.1.0)

Imports stats, utils, lpSolve (>= 5.6.18), nloptr

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Suggests knitr, rmarkdown, forecast, glarma, scoringRules, ggplot2, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

URL https://github.com/IDSIA/bayesRecon, https://idsia.github.io/bayesRecon/

BugReports https://github.com/IDSIA/bayesRecon/issues

NeedsCompilation no

Author Dario Azzimonti [aut, cre] (ORCID:
<https://orcid.org/0000-0001-5080-3061>),
Lorenzo Zambon [aut] (ORCID: <https://orcid.org/0000-0002-8939-993X>),
Stefano Damato [aut] (ORCID: <https://orcid.org/0009-0001-0225-3736>),
Nicolò Rubattu [aut] (ORCID: <https://orcid.org/0000-0002-2703-1005>),
Giorgio Corani [aut] (ORCID: <https://orcid.org/0000-0002-1541-8384>)

Repository CRAN

Date/Publication 2026-04-16 10:32:05 UTC

Contents

`carparts_example` *Example of a time series from carparts*

Description

A monthly time series from the `carparts` dataset, 51 observations, Jan 1998 - Mar 2002.

Usage

```
carparts_example
```

Format

Univariate time series of class ts.

Source

Godahewa, R., Bergmeir, C., Webb, G., Hyndman, R.J., & Montero-Manso, P. (2020). Car Parts Dataset (without Missing Values) (Version 2) doi:10.5281/zenodo.4656021

References

Hyndman, R.J., Koehler, A.B., Ord, J.K., and Snyder, R.D., (2008). Forecasting with exponential smoothing: the state space approach. Springer Science & Business Media.

Godahewa, R., Bergmeir, C., Webb, G., Hyndman, R., & Montero-Manso, P. (2020). Car Parts Dataset (without Missing Values) (Version 2) doi:10.5281/zenodo.4656021

`extr_mkt_events` *Extreme market events dataset*

Description

Count time series of extreme market events in five economic sectors. The data refer to the trading days between 2004/12/31 and 2018/12/19 (3508 trading days in total).

Usage

```
extr_mkt_events
```

Format

A multivariate time series of class ts.

Details

The counts are computed by considering 29 companies included in the Euro Stoxx 50 index and observing if the value of the CDS spread on a given day exceeds the 90-th percentile of its distribution in the last trading year. The companies are divided in the following sectors: Financial (FIN), Information and Communication Technology (ICT), Manufacturing (MFG), Energy (ENG), and Trade (TRD).

There are 6 time series:

- 5 bottom time series, corresponding to the daily counts for each sector
- 1 upper time series, which is the sum of all the bottom (ALL)

Source

Zambon, L., Agosto, A., Giudici, P., Corani, G. (2024). *Properties of the reconciled distributions for Gaussian and count forecasts*. International Journal of Forecasting (in press). doi:10.1016/j.ijforecast.2023.12.004.

References

Zambon, L., Agosto, A., Giudici, P., Corani, G. (2024). *Properties of the reconciled distributions for Gaussian and count forecasts*. International Journal of Forecasting (in press). doi:10.1016/j.ijforecast.2023.12.004.

Agosto, A. (2022). *Multivariate Score-Driven Models for Count Time Series to Assess Financial Contagion*. doi:10.2139/ssrn.4119895

`extr_mkt_events_basefc`

Base forecasts for the extreme market events dataset

Description

Base forecasts for the `extr_mkt_events` dataset, computed using the model by Agosto, A. (2022). *Multivariate Score-Driven Models for Count Time Series to Assess Financial Contagion*. doi:10.2139/ssrn.4119895.

Usage

```
extr_mkt_events_basefc
```

Format

A list `extr_mkt_events_basefc` containing

`extr_mkt_events_basefc$mu` data frame of the base forecast means, for each day

`extr_mkt_events_basefc$size` data frame of the static base forecast size parameters

Details

The predictive distribution for the bottom time series is a multivariate negative binomial with a static vector of dispersion parameters and a time-varying vector of location parameters following a score-driven dynamics. The base forecasts for the upper time series are computed using a univariate version of this model. They are in-sample forecasts: for each training instant, they are computed for time t+1 by conditioning on the counts observed up to time t.

Source

Agosto, A. (2022). *Multivariate Score-Driven Models for Count Time Series to Assess Financial Contagion*. doi:10.2139/ssrn.4119895

References

Agosto, A. (2022). *Multivariate Score-Driven Models for Count Time Series to Assess Financial Contagion*. doi:10.2139/ssrn.4119895

Zambon, L., Agosto, A., Giudici, P., Corani, G. (2024). *Properties of the reconciled distributions for Gaussian and count forecasts*. International Journal of Forecasting (in press). doi:10.1016/j.ijforecast.2023.12.004.

`get_reconc_matrices` *Build hierarchy matrices*

Description

Creates the aggregation and summing matrices for a temporal hierarchy of time series from a user-selected list of aggregation levels.

Usage

```
get_reconc_matrices(agg_levels, h)
```

Arguments

`agg_levels`	user-selected list of aggregation levels.
`h`	number of steps ahead for the bottom level forecasts.

Value

A list containing the named elements:

- `A` the aggregation matrix;
- `S` the summing matrix.

See Also

`temporal_aggregation()`

Examples

```
library(bayesRecon)

# Create monthly hierarchy
agg_levels <- c(1, 2, 3, 4, 6, 12)
h <- 12
rec_mat <- get_reconc_matrices(agg_levels, h)
S <- rec_mat$S
A <- rec_mat$A
```

infantMortality *Infant Mortality grouped time series dataset*

Description

A yearly grouped time series dataset, from 1901 to 2003, of infant mortality counts (deaths) in Australia; disaggregated by state (see below), and sex (male and female).

Usage

```
infantMortality
```

Format

List of time series of class ts.

Details

States: New South Wales (NSW), Victoria (VIC), Queensland (QLD), South Australia (SA), Western Australia (WA), Northern Territory (NT), Australian Capital Territory (ACT), and Tasmania (TAS).

Source

hts package CRAN

References

Hyndman, R.J., Ahmed, R.A., Athanasopoulos, G., Shang, H.L. (2011). Optimal combination forecasts for hierarchical time series. Computational Statistics and Data Analysis, 55(9), 2579-2589.

M3_example *Example of a time series from the M3 forecasting competition*

Description

A monthly time series, from the M3 forecasting competition ("N1485").

Usage

```
M3_example
```

Format

List of time series of class ts.

Source

https://forecasters.org/resources/time-series-data/m3-competition/

M5_CA1_basefc *Example of hierarchical forecasts for a store from the M5 competition*

Description

This dataset contains forecasts for the hierarchy of time series related to the store CA_1 from the M5 competition.

Usage

```
M5_CA1_basefc
```

Format

A list containing:

- upper: a list of 11 elements each representing an aggregation level. Each element contains: mu, sigma the mean and standard deviation of the Gaussian forecast, actual the actual value, residuals the residuals of the model used to estimate forecasts covariance.
- lower: a list of 3049 elements each representing a forecast for each item. Each element contains pmf the probability mass function of the item level forecast, actual the actual value.
- A: the aggregation matrix for A.
- S: the S matrix for the hierarchy.
- Q_u: scaling factors for computing MASE on the upper forecasts.
- Q_b: scaling factors for computing MASE on the bottom forecasts.

Details

The store CA_1 contains 3049 item level time series and 11 aggregate time series:

- Store level aggregation (CA_1)
- Category level aggregations (HOBBIES, HOUSEHOLD, FOODS)
- Department level aggregations (HOBBIES_1, HOBBIES_2, HOUSEHOLD_1, HOUSEHOLD_2, FOODS_1, FOODS_2, FOODS_3)

Forecasts are generated with the function forecast and the model adam from the package smooth.

- The models for the bottom time series are selected with multiplicative Gamma error term (MNN);
- The models for the upper time series (AXZ) is selected with Gaussian additive error term, seasonality selected based on information criterion.

The raw data was downloaded with the package m5.

Source

Makridakis, Spyros & Spiliotis, Evangelos & Assimakopoulos, Vassilis. (2020). *The M5 Accuracy competition: Results, findings and conclusions.* International Journal of Forecasting 38(4) 1346-1364. doi:10.1016/j.ijforecast.2021.10.009

References

Joachimiak K (2022). *m5: 'M5 Forecasting' Challenges Data*. R package version 0.1.1, https://CRAN.R-project.org/package=m5.

Makridakis, Spyros & Spiliotis, Evangelos & Assimakopoulos, Vassilis. (2020). *The M5 Accuracy competition: Results, findings and conclusions.* International Journal of Forecasting 38(4) 1346-1364. doi:10.1016/j.ijforecast.2021.10.009

Svetunkov I (2023). *smooth: Forecasting Using State Space Models*. R package version 4.0.0, https://CRAN.R-project.org/package=smooth.

`Mixed_reconciliation` *Probabilistic forecast reconciliation of mixed hierarchies*

Description

`reconc_MixCond()` uses importance sampling to draw samples from the reconciled forecast distribution, obtained via conditioning, in the case of a mixed hierarchy.

`reconc_TDcond()` uses a top-down conditioning algorithm: first, upper base forecasts are reconciled via conditioning using only the hierarchical constraints between the upper; then, the bottom distributions are updated via a probabilistic top-down procedure.

Usage

```
reconc_MixCond(
  A,
  base_fc_bottom,
  base_fc_upper,
  bottom_in_type = "pmf",
  distr = NULL,
  num_samples = 20000,
  return_type = "pmf",
  return_upper = TRUE,
  suppress_warnings = FALSE,
  seed = NULL
)

reconc_TDcond(
  A,
  base_fc_bottom,
  base_fc_upper,
  bottom_in_type = "pmf",
```

```
  distr = NULL,
  num_samples = 20000,
  return_type = "pmf",
  return_upper = TRUE,
  suppress_warnings = TRUE,
  seed = NULL
)
```

Arguments

A	Aggregation matrix (n_upper x n_bottom).
base_fc_bottom	A list containing the bottom base forecasts, see details.
base_fc_upper	A list containing the upper base forecasts, see details.
bottom_in_type	A string with three possible values: • 'pmf' if the bottom base forecasts are in the form of pmf, see details; • 'samples' if the bottom base forecasts are in the form of samples; • 'params' if the bottom base forecasts are in the form of estimated parameters.
distr	A string describing the type of bottom base forecasts ('poisson' or 'nbinom'). This is only used if bottom_in_type='params'.
num_samples	Number of samples drawn from the reconciled distribution. This is ignored if bottom_in_type='samples'; in this case, the number of reconciled samples is equal to the number of samples of the base forecasts.
return_type	The return type of the reconciled distributions. A string with three possible values: • 'pmf' returns a list containing the reconciled marginal pmf objects; • 'samples' returns a list containing the reconciled multivariate samples; • 'all' returns a list with both pmf objects and samples.
return_upper	Logical, whether to return the reconciled parameters for the upper variables (default is TRUE).
suppress_warnings	Logical. If TRUE, no warnings about samples are triggered; if FALSE, warnings are generated. Default is FALSE for reconc_MixCond and TRUE for reconc_TDcond. See the respective sections above.
seed	Seed for reproducibility.

Details

The base bottom forecasts base_fc_bottom must be a list of length n_bottom, where each element is either

- a PMF object (see details below), if bottom_in_type='pmf';
- a vector of samples, if bottom_in_type='samples';
- a list of parameters, if bottom_in_type='params':

  - lambda for the Poisson base forecast if `distr='poisson'`, see Poisson;
  - size and prob (or mu) for the negative binomial base forecast if `distr='nbinom'`, see NegBinomial.

The base upper forecasts `base_fc_upper` must be a list containing the parameters of the multivariate Gaussian distribution of the upper forecasts. The list must contain only the named elements `mean` (vector of length n_upper) and `cov` (n_upper x n_upper matrix).

The order of the upper and bottom base forecasts must match the order of (respectively) the rows and the columns of A.

A PMF object is a numerical vector containing the probability mass function of a discrete distribution. Each element corresponds to the probability of the integers from 0 to the last value of the support. See also PMF for functions that handle PMF objects.

Warnings and errors.

In `reconc_MixCond`, warnings are triggered from the importance sampling step if:

- weights are all zeros, then the upper forecast is ignored during reconciliation;
- the effective sample size is < 200;
- the effective sample size is $< 1\%$ of the sample size.

These warnings are an indication that the base forecasts might have issues. Please check the base forecasts in case of warnings.

In `reconc_TDcond`, if some of the reconciled upper samples lie outside the support of the bottom-up distribution, those samples are discarded; the remaining ones are resampled with replacement, so that the number of output samples is equal to `num_samples`. In this case, a warning is issued if `suppress_warnings=FALSE` (default is `TRUE`). If the fraction of discarded samples is above 50%, the function returns an error.

Value

A list containing the reconciled forecasts. The list has the following named elements:

- `bottom_rec_pmf`: a list of PMF objects for each bottom series (only if `return_type` is `'pmf'` or `'all'`);
- `bottom_rec_samples`: a matrix (n_bottom x `num_samples`) of reconciled bottom samples (only if `return_type` is `'samples'` or `'all'`);
- `upper_rec_pmf`: a list of PMF objects for each upper series (only if `return_type` is `'pmf'` or `'all'`, and `return_upper = TRUE`);
- `upper_rec_samples`: a matrix (n_upper x `num_samples`) of reconciled upper samples (only if `return_type` is `'samples'` or `'all'`, and `return_upper = TRUE`).

References

Zambon, L., Azzimonti, D., Rubattu, N., Corani, G. (2024). *Probabilistic reconciliation of mixed-type hierarchical time series*. Proceedings of the Fortieth Conference on Uncertainty in Artificial Intelligence, PMLR 244:4078-4095. https://proceedings.mlr.press/v244/zambon24a.html.

See Also

reconc_BUIS(), reconc_gaussian(), PMF

Examples

```
library(bayesRecon)

# Consider a simple hierarchy with two bottom and one upper
A <- matrix(c(1, 1), nrow = 1)
# The bottom forecasts are Poisson with lambda=15
lambda <- 15
n_tot <- 60
base_fc_bottom <- list()
base_fc_bottom[[1]] <- apply(matrix(seq(0, n_tot)), MARGIN = 1,
                             FUN = \(x) dpois(x, lambda = lambda))
base_fc_bottom[[2]] <- apply(matrix(seq(0, n_tot)), MARGIN = 1,
                             FUN = \(x) dpois(x, lambda = lambda))

# The upper forecast is a Normal with mean 40 and std 5
base_fc_upper <- list(mean = 40, cov = matrix(5^2))

# Reconcile with reconc_MixCond
res.mixCond <- reconc_MixCond(A, base_fc_bottom, base_fc_upper)

# Note that the bottom distributions are slightly shifted to the right
PMF_summary(res.mixCond$bottom_rec_pmf[[1]])
PMF_summary(base_fc_bottom[[1]])

PMF_summary(res.mixCond$bottom_rec_pmf[[2]])
PMF_summary(base_fc_bottom[[2]])

# The upper distribution is slightly shifted to the left
PMF_summary(res.mixCond$upper_rec_pmf[[1]])
PMF_get_var(res.mixCond$upper_rec_pmf[[1]])


library(bayesRecon)

# Consider a simple hierarchy with two bottom and one upper
A <- matrix(c(1, 1), nrow = 1)
# The bottom forecasts are Poisson with lambda=15
lambda <- 15
n_tot <- 60
base_fc_bottom <- list()
base_fc_bottom[[1]] <- apply(matrix(seq(0, n_tot)), MARGIN = 1,
                             FUN = \(x) dpois(x, lambda = lambda))
base_fc_bottom[[2]] <- apply(matrix(seq(0, n_tot)), MARGIN = 1,
                             FUN = \(x) dpois(x, lambda = lambda))

# The upper forecast is a Normal with mean 40 and std 5
base_fc_upper <- list(mean = 40, cov = matrix(c(5^2)))
```

```
# Reconcile with reconc_TDcond
res.TDcond <- reconc_TDcond(A, base_fc_bottom, base_fc_upper)

# Note that the bottom distributions are shifted to the right
PMF_summary(res.TDcond$bottom_rec_pmf[[1]])
PMF_summary(base_fc_bottom[[1]])

PMF_summary(res.TDcond$bottom_rec_pmf[[2]])
PMF_summary(base_fc_bottom[[2]])

# The upper distribution remains similar
PMF_summary(res.TDcond$upper_rec_pmf[[1]])
PMF_get_var(res.TDcond$upper_rec_pmf[[1]])

## Example 2: reconciliation with unbalanced hierarchy
# We consider the example in Fig. 9 of Zambon et al. (2024).

# The hierarchy has 5 bottoms and 3 uppers
A <- matrix(c(
  1, 1, 1, 1, 1,
  1, 1, 0, 0, 0,
  0, 0, 1, 1, 0
), nrow = 3, byrow = TRUE)
# Note that the 5th bottom only appears in the highest level, this is an unbalanced hierarchy.
n_upper <- nrow(A)
n_bottom <- ncol(A)

# The bottom forecasts are Poisson with lambda=15
lambda <- 15
n_tot <- 60
base_fc_bottom <- list()
for (i in seq(n_bottom)) {
  base_fc_bottom[[i]] <- apply(matrix(seq(0, n_tot)), MARGIN = 1,
                               FUN = \(x) dpois(x, lambda = lambda))
}

# The upper forecasts are a multivariate Gaussian
mean <- c(75, 30, 30)
cov <- matrix(c(
  5^2, 5, 5,
  5, 10, 0,
  5, 0, 10
), nrow = 3, byrow = TRUE)

base_fc_upper <- list(mean = mean, cov = cov)
## Not run:
# If we reconcile with reconc_TDcond it won't work (unbalanced hierarchy)
res.TDcond <- reconc_TDcond(A, base_fc_bottom, base_fc_upper)

## End(Not run)

# We can balance the hierarchy by duplicating the node b5:
# i) consider the time series observations for b5 as the upper u4,
```

```
# ii) fit the multivariate ts model for u1, u2, u3, u4.

# In this example we simply assume that the forecast for u1-u4 is
# Gaussian with the mean and variance of u4 given by the parameters in b5.
mean_b5 <- lambda
var_b5 <- lambda
mean <- c(75, 30, 30, mean_b5)
cov <- matrix(c(
  5^2, 5, 5, 5,
  5, 10, 0, 0,
  5, 0, 10, 0,
  5, 0, 0, var_b5
), nrow = 4, byrow = TRUE)
base_fc_upper <- list(mean = mean, cov = cov)

# We also need to update the aggregation matrix
A <- matrix(c(
  1, 1, 1, 1, 1,
  1, 1, 0, 0, 0,
  0, 0, 1, 1, 0,
  0, 0, 0, 0, 1
), nrow = 4, byrow = TRUE)

# We can now reconcile with TDcond
res.TDcond <- reconc_TDcond(A, base_fc_bottom, base_fc_upper)

# Note that the reconciled distribution of b5 and u4 are identical,
# keep this in mind when using the results of your reconciliation!
max(abs(res.TDcond$bottom_rec_pmf[[5]] - res.TDcond$upper_rec_pmf[[4]]))
```

`multi_log_score_optimization` *Optimize degrees of freedom (nu) via LOO Cross-Validation*

Description

Optimize degrees of freedom (nu) via LOO Cross-Validation

Usage

```
multi_log_score_optimization(res, prior_mean, trim = 0.1)
```

Arguments

`res`	Matrix of residuals (n_obs x n_var).
`prior_mean`	The prior mean covariance matrix (n_var x n_var).
`trim`	Fraction of observations to trim (0 to 1). Default is 0.1 (10%).

Details

TODO: check these details

Leave-One-Out (LOO) Cross-Validation: This function estimates the optimal degrees of freedom ν by maximizing the out-of-sample predictive performance. This is achieved by computing the log-density of each held-out observation $\mathbf{r}_i$ given the remaining data $\mathbf{R}_{-i}$. The total objective function is the sum of these predictive log-densities:

$$\mathcal{L}(\nu) = \sum_{i=1}^{T} \log f(\mathbf{r}_i | \mathbf{R}_{-i}, \nu)$$

The Log-Density Function: For each LOO step, the residuals are assumed to follow a multivariate t-distribution. The density is expressed directly as a function of the posterior sum-of-squares matrix Ψ, where Ψ scales implicitly with ν:

$$f(\mathbf{r}_i | \Psi, \nu) = \frac{\Gamma(\frac{\nu+T}{2})}{\Gamma(\frac{\nu+T-p}{2})\pi^{p/2}} |\Psi|^{-1/2} \left(1 + \mathbf{r}_i^\top \Psi^{-1} \mathbf{r}_i\right)^{-\frac{\nu+T}{2}}$$

In the code, Ψ is constructed as:

$$\Psi = (\nu - p - 1)\bar{\Sigma}_{prior} + \mathbf{R}^\top \mathbf{R}$$

By using this formulation, the standard scaling factors $1/\nu$ and $\nu^{-p/2}$ are absorbed into the matrix inverse and determinant, respectively.

Efficient Computation via Sherman-Morrison: Rather than recomputing Ψ_{-i} and its inverse T times, the function uses the full-sample matrix Ψ and adjusts it using the leverage $h_i = \mathbf{r}_i^\top \Psi^{-1} \mathbf{r}_i$.

Through the Matrix Determinant Lemma and the Sherman-Morrison formula, the internal term $(1 + \mathbf{r}_i^\top \Psi_{-i}^{-1} \mathbf{r}_i)$ simplifies to $(1 - h_i)^{-1}$. The final log-density contribution used in the code is:

$$\log f_i \propto \text{const} - \frac{1}{2} \log |\Psi| + \frac{\nu + T - 1}{2} \log(1 - h_i)$$

Value

A list containing the optimization results:

- `optimal_nu`: The optimal degrees of freedom found.
- `min_neg_log_score`: The minimum negative log score achieved.
- `convergence`: The convergence status of the optimizer.
- `time_elapsed`: The time taken for the optimization.

PMF *PMF operations and utilities*

Description

A set of functions for working with Probability Mass Functions (PMFs) in bayesRecon. A PMF is represented as a normalized numeric vector where element `v[j+1]` represents the probability of value `j` (support starts from 0).

These functions provide utilities for:

- Drawing samples from PMFs
- Computing summary statistics (mean, variance, quantiles)
- Summarizing PMF distributions

Usage

```
PMF_sample(pmf, N_samples)
PMF_get_mean(pmf)
PMF_get_var(pmf)
PMF_get_quantile(pmf, p)
PMF_summary(pmf, Ltoll = .TOLL, Rtoll = .RTOLL)

PMF_get_mean(pmf)

PMF_get_var(pmf)

PMF_get_quantile(pmf, p)

PMF_summary(pmf, Ltoll = .TOLL, Rtoll = .RTOLL)
```

Arguments

`pmf`	A PMF object (numeric vector where element j+1 is the probability of value j).
`N_samples`	Number of samples to draw from the PMF.
`p`	Probability level for quantile computation. Either a number or a vector with all elements between 0 and 1.
`Ltoll`	Lower tolerance for computing the minimum of the PMF (default: 1e-15).
`Rtoll`	Upper tolerance for computing the maximum of the PMF (default: 1e-9).

Functions

`PMF_sample(pmf, N_samples)` Draws random samples from the probability distribution specified by the PMF. Uses sampling with replacement from the discrete support values, weighted by their probabilities. This is useful for generating synthetic data or for Monte Carlo simulations.

`PMF_get_mean(pmf)` Computes the expected value (mean) of the distribution represented by the PMF. The mean is calculated as the sum of each value in the support weighted by its probability: $\sum_x x \cdot P(X = x)$.

`PMF_get_var(pmf)` Computes the variance of the distribution represented by the PMF. The variance measures the spread of the distribution and is calculated as $E[X^2] - (E[X])^2$, where $E[X]$ is the mean.

`PMF_get_quantile(pmf, p)` Computes the quantile of the distribution at probability level `p`. Returns the smallest value `x` such that the cumulative probability up to `x` is greater than or equal to `p`. For example, `p=0.5` gives the median. The parameter `p` can be a single number or a vector of probabilities.

`PMF_summary(pmf, Ltoll, Rtoll)` Provides a comprehensive summary of the distribution including minimum, maximum, quartiles, median, and mean. The minimum and maximum are determined based on probability thresholds to handle the potentially infinite tails of discrete distributions.

Examples

```
library(bayesRecon)

# Let's build the pmf of a Binomial distribution with parameters n and p
n <- 10
p <- 0.6
pmf_binomial <- apply(matrix(seq(0, n)), MARGIN = 1, FUN = \(x) dbinom(x, size = n, prob = p))

# Draw samples from the PMF object
set.seed(1)
samples <- PMF_sample(pmf = pmf_binomial, N_samples = 1e4)

# Compute statistics
PMF_get_mean(pmf_binomial) # Mean: should be close to n*p = 6
PMF_get_var(pmf_binomial) # Variance: should be close to n*p*(1-p) = 2.4
PMF_get_quantile(pmf_binomial, 0.5) # Median
PMF_summary(pmf_binomial) # Full summary
```

`reconc_BUIS` *BUIS for Probabilistic Reconciliation of forecasts via conditioning*

Description

Uses the Bottom-Up Importance Sampling algorithm to draw samples from the reconciled forecast distribution, obtained via conditioning.

Usage

```
reconc_BUIS(
  A,
  base_fc,
```

```
  in_type,
  distr,
  num_samples = 20000,
  suppress_warnings = FALSE,
  return_upper = TRUE,
  seed = NULL
)
```

Arguments

A	aggregation matrix (n_upper x n_bottom).
base_fc	A list containing the base_forecasts, see details.
in_type	A string or a list of length n_upper + n_bottom. If it is a list the i-th element is a string with two possible values: • 'samples' if the i-th base forecasts are in the form of samples; • 'params' if the i-th base forecasts are in the form of estimated parameters. If it in_type is a string it is assumed that all base forecasts are of the same type.
distr	A string or a list of length n_upper + n_bottom describing the type of base forecasts. If it is a list the i-th element is a string with two possible values: • 'continuous' or 'discrete' if in_type[[i]]='samples'; • 'gaussian', 'poisson' or 'nbinom' if in_type[[i]]='params'. If distr is a string it is assumed that all distributions are of the same type.
num_samples	Number of samples drawn from the reconciled distribution. This is ignored if bottom_in_type='samples'; in this case, the number of reconciled samples is equal to the number of samples of the base forecasts.
suppress_warnings	Logical. If TRUE, no warnings about effective sample size are triggered. If FALSE, warnings are generated. Default is FALSE. See Details.
return_upper	Logical, whether to return the reconciled parameters for the upper variables (default is TRUE).
seed	Seed for reproducibility.

Details

The parameter base_fc is a list containing n = n_upper + n_bottom elements. The first n_upper elements of the list are the upper base forecasts, in the order given by the rows of A. The elements from n_upper+1 until the end of the list are the bottom base forecasts, in the order given by the columns of A.

The i-th element depends on the values of in_type[[i]] and distr[[i]].

If in_type[[i]]='samples', then base_fc[[i]] is a vector containing samples from the base forecast distribution.

If in_type[[i]]='params', then base_fc[[i]] is a list containing the estimated:

- mean and sd for the Gaussian base forecast if distr[[i]]='gaussian', see Normal;
- lambda for the Poisson base forecast if distr[[i]]='poisson', see Poisson;

- size and prob (or mu) for the negative binomial base forecast if `distr[[i]]`='nbinom', see NegBinomial.

See the description of the parameters `in_type` and `distr` for more details.

Warnings are triggered from the Importance Sampling step if:

- weights are all zeros, then the upper is ignored during reconciliation;
- the effective sample size is < 200;
- the effective sample size is < 1% of the sample size (`num_samples` if `in_type` is 'params' or the size of the base forecast if if `in_type` is 'samples').

Note that warnings are an indication that the base forecasts might have issues. Please check the base forecasts in case of warnings.

Value

A list containing the reconciled forecasts. The list has the following named elements:

- `bottom_rec_samples`: a matrix (n_bottom x `num_samples`) containing the reconciled samples for the bottom time series;
- `upper_rec_samples`: (only if `return_upper = TRUE`) a matrix (n_upper x `num_samples`) containing the reconciled samples for the upper time series.

References

Zambon, L., Azzimonti, D. & Corani, G. (2024). *Efficient probabilistic reconciliation of forecasts for real-valued and count time series*. Statistics and Computing 34 (1), 21. doi:10.1007/s11222-02310343y.

See Also

`reconc_gaussian()`

Examples

```
library(bayesRecon)

# Create a minimal hierarchy with 2 bottom and 1 upper variable
rec_mat <- get_reconc_matrices(agg_levels = c(1, 2), h = 2)
A <- rec_mat$A
S <- rec_mat$S


# 1) Gaussian base forecasts

# Set the parameters of the Gaussian base forecast distributions
mu1 <- 2
mu2 <- 4
muY <- 9
mus <- c(muY, mu1, mu2)
```

```
sigma1 <- 2
sigma2 <- 2
sigmaY <- 3
sigmas <- c(sigmaY, sigma1, sigma2)

base_fc <- list()
for (i in 1:length(mus)) {
  base_fc[[i]] <- list(mean = mus[[i]], sd = sigmas[[i]])
}


# Sample from the reconciled forecast distribution using the BUIS algorithm
buis <- reconc_BUIS(A, base_fc,
  in_type = "params",
  distr = "gaussian", num_samples = 100000, seed = 42
)

samples_buis <- rbind(buis$upper_rec_samples, buis$bottom_rec_samples)

# In the Gaussian case, the reconciled distribution is still Gaussian and can be
# computed in closed form
Sigma <- diag(sigmas^2) # transform into covariance matrix
analytic_rec <- reconc_gaussian(A,
  base_fc_mean = mus,
  base_fc_cov = Sigma
)

# Compare the reconciled means obtained analytically and via BUIS
print(c(S %*% analytic_rec$bottom_rec_mean))
print(rowMeans(samples_buis))


# 2) Poisson base forecasts

# Set the parameters of the Poisson base forecast distributions
lambda1 <- 2
lambda2 <- 4
lambdaY <- 9
lambdas <- c(lambdaY, lambda1, lambda2)

base_fc <- list()
for (i in 1:length(lambdas)) {
  base_fc[[i]] <- list(lambda = lambdas[i])
}

# Sample from the reconciled forecast distribution using the BUIS algorithm
buis <- reconc_BUIS(A, base_fc,
  in_type = "params",
  distr = "poisson", num_samples = 100000, seed = 42
)
samples_buis <- rbind(buis$upper_rec_samples, buis$bottom_rec_samples)

# Print the reconciled means
```

```
print(rowMeans(samples_buis))
```

`reconc_gaussian` *Analytical reconciliation of Gaussian base forecasts*

Description

Closed form computation of the reconciled forecasts in case of Gaussian base forecasts.

Usage

```
reconc_gaussian(
  A,
  base_fc_mean,
  base_fc_cov = NULL,
  residuals = NULL,
  return_upper = FALSE
)
```

Arguments

`A`	aggregation matrix (n_upper x n_bottom).
`base_fc_mean`	a vector containing the means of the base forecasts.
`base_fc_cov`	a matrix containing the covariance matrix of the base forecasts.
`residuals`	a matrix with the residuals of the base forecasts, with n_upper + n_bottom columns. The covariance matrix of the base forecasts is computed from the residuals using the Schäfer Strimmer shrinkage estimator. If base_fc_cov is provided, residuals are ignored.
`return_upper`	logical, whether to return the reconciled parameters for the upper variables (default is FALSE).

Details

In the vector of the means of the base forecasts the order must be: first the upper, then the bottom; the order within the uppers is given by the rows of A, the order within the bottoms by the columns of A. The order of the rows of the covariance matrix of the base forecasts is the same.

Unless `return_upper = TRUE`, the function returns only the reconciled parameters of the bottom variables.

Value

A list containing the reconciled forecasts. The list has the following named elements:

- `bottom_rec_mean`: reconciled mean for the bottom forecasts;
- `bottom_rec_cov`: reconciled covariance for the bottom forecasts;

- upper_rec_mean: (only if return_upper = TRUE) reconciled mean for the upper forecasts;
- upper_rec_cov: (only if return_upper = TRUE) reconciled covariance for the upper forecasts.

References

Corani, G., Azzimonti, D., Augusto, J.P.S.C., Zaffalon, M. (2021). *Probabilistic Reconciliation of Hierarchical Forecast via Bayes' Rule*. ECML PKDD 2020. Lecture Notes in Computer Science, vol 12459. doi:10.1007/9783030676643_13.

Zambon, L., Agosto, A., Giudici, P., Corani, G. (2024). *Properties of the reconciled distributions for Gaussian and count forecasts*. International Journal of Forecasting (in press). doi:10.1016/j.ijforecast.2023.12.004.

See Also

reconc_t(), reconc_BUIS()

Examples

```
library(bayesRecon)

#' # ---- Example 1: base forecasts are given ----

# Create a minimal hierarchy with 2 bottom and 1 upper variable
A <- get_reconc_matrices(agg_levels = c(1, 2), h = 2)$A

# Set the parameters of the Gaussian base forecast distributions
mu1 <- 2
mu2 <- 4
muY <- 9
base_fc_mean <- c(muY, mu1, mu2)  # vector of means

sigma1 <- 2
sigma2 <- 2
sigmaY <- 3
sigmas <- c(sigmaY, sigma1, sigma2)
base_fc_cov <- diag(sigmas^2)  # covariance matrix

analytic_rec <- reconc_gaussian(A,
  base_fc_mean = base_fc_mean,
  base_fc_cov = base_fc_cov
)

bottom_mean_rec <- analytic_rec$bottom_rec_mean
bottom_cov_rec <- analytic_rec$bottom_rec_cov

# To obtain reconciled samples for the entire hierarchy, sample from the reconciled
# bottom distribution and then aggregate using A.

# Sample from the reconciled bottom-level Gaussian distribution
# First, compute the Cholesky decomposition of the reconciled covariance matrix:
```

```
chol_decomp <- chol(bottom_cov_rec)
# Then, sample from the standard normal distribution and apply the transformation:
Z <- matrix(stats::rnorm(n = 2000), nrow = 2)
B <- t(chol_decomp) %*% Z + matrix(rep(bottom_mean_rec, 1000), nrow = 2)

# Aggregate bottom samples to get upper samples, then stack
U <- A %*% B
Y_reconc <- rbind(U, B)

cat("Dimensions of reconciled samples (upper + bottom):", dim(Y_reconc), "\n")


# ---- Example 2: using residuals from fitted ETS models ----

if (requireNamespace("forecast", quietly = TRUE)) {

  # Simulate 2 bottom series from AR(1) processes
  set.seed(1234)
  n_obs <- 200
  y1 <- arima.sim(model = list(ar = 0.8), n = n_obs)
  y2 <- arima.sim(model = list(ar = 0.5), n = n_obs)

  # Upper series is the sum of the two bottom series
  y_upper <- y1 + y2

  # Aggregation matrix A:
  A <- matrix(c(1, 1), nrow = 1)

  # Fit additive ETS models
  fit1 <- forecast::ets(y1, additive.only = TRUE)
  fit2 <- forecast::ets(y2, additive.only = TRUE)
  fit_upper <- forecast::ets(y_upper, additive.only = TRUE)

  # Point forecasts (h = 1):
  fc1 <- forecast::forecast(fit1, h = 1)$mean
  fc2 <- forecast::forecast(fit2, h = 1)$mean
  fc_upper <- forecast::forecast(fit_upper, h = 1)$mean
  base_fc_mean <- c(fc_upper, fc1, fc2)

  # Residuals matrix (T x n, columns in same order as base_fc_mean)
  res <- cbind(residuals(fit_upper),
               residuals(fit1),
               residuals(fit2))

  # Reconcile (covariance estimated internally via Schafer-Strimmer)
 result <- reconc_gaussian(A, base_fc_mean = base_fc_mean, residuals = res, return_upper = TRUE)

  bottom_mean <- result$bottom_rec_mean
  bottom_cov <- result$bottom_rec_cov
  upper_mean <- result$upper_rec_mean
  upper_cov <- result$upper_rec_cov

  # Print reconciled means
```

```
  cat("Reconciled bottom means:", round(bottom_mean, 3), "\n")
  cat("Reconciled upper mean:", round(upper_mean, 3), "\n")

  # Print 95% predictions intervals
  cat("Reconciled bottom 95% prediction intervals:\n")
  for (i in 1:length(bottom_mean)) {
    lower <- bottom_mean[i] - 1.96 * sqrt(bottom_cov[i, i])
    upper <- bottom_mean[i] + 1.96 * sqrt(bottom_cov[i, i])
    cat(paste0("Bottom ", i, ": [", round(lower, 3), ", ", round(upper, 3), "]\n"))
  }
  cat("Reconciled upper 95% prediction interval:\n")
  lower <- upper_mean - 1.96 * sqrt(upper_cov[1, 1])
  upper <- upper_mean + 1.96 * sqrt(upper_cov[1, 1])
  cat(paste0("Upper: [", round(lower, 3), ", ", round(upper, 3), "]\n"))

}
```

---

reconc_MCMC *MCMC for Probabilistic Reconciliation of forecasts via conditioning*

Description

Uses Markov Chain Monte Carlo algorithm to draw samples from the reconciled forecast distribution, which is obtained via conditioning.

This is a bare-bones implementation of the Metropolis-Hastings algorithm, we suggest the usage of tools to check the convergence. The function only works with Poisson or Negative Binomial base forecasts.

The function reconc_BUIS() is generally faster on most hierarchies.

Usage

```
reconc_MCMC(
  A,
  base_fc,
  distr,
  num_samples = 10000,
  tuning_int = 100,
  init_scale = 1,
  burn_in = 1000,
  return_upper = TRUE,
  seed = NULL
)
```

Arguments

A	aggregation matrix (n_upper x n_bottom).
base_fc	list of the parameters of the base forecast distributions, see details.
distr	a string describing the type of predictive distribution.
num_samples	number of samples to draw using MCMC.
tuning_int	number of iterations between scale updates of the proposal.
init_scale	initial scale of the proposal.
burn_in	number of initial samples to be discarded.
return_upper	Logical. If TRUE (default), also returns the reconciled samples for the upper time series. Default is TRUE.
seed	seed for reproducibility.

Details

The parameter base_fc is a list containing n = n_upper + n_bottom elements. Each element is a list containing the estimated:

- mean and sd for the Gaussian base forecast, see Normal, if distr='gaussian';
- lambda for the Poisson base forecast, see Poisson, if distr='poisson';
- size and prob (or mu) for the negative binomial base forecast, see NegBinomial, if distr='nbinom'.

The first n_upper elements of the list are the upper base forecasts, in the order given by the rows of A. The elements from n_upper+1 until the end of the list are the bottom base forecasts, in the order given by the columns of A.

Value

A list containing the reconciled forecasts. The list has the following named elements:

- bottom_rec_samples: a matrix (n_bottom x num_samples) containing reconciled samples for the bottom time series;
- upper_rec_samples: a matrix (n_upper x num_samples) containing reconciled samples for the upper time series (only if return_upper = TRUE).

References

Corani, G., Azzimonti, D., Rubattu, N. (2024). *Probabilistic reconciliation of count time series*. International Journal of Forecasting 40 (2), 457-469. doi:10.1016/j.ijforecast.2023.04.003.

See Also

reconc_BUIS()

Examples

```
library(bayesRecon)

# Create a minimal hierarchy with 2 bottom and 1 upper variable
rec_mat <- get_reconc_matrices(agg_levels = c(1, 2), h = 2)
A <- rec_mat$A

# Set the parameters of the Poisson base forecast distributions
lambda1 <- 2
lambda2 <- 4
lambdaY <- 9
lambdas <- c(lambdaY, lambda1, lambda2)

base_fc <- list()
for (i in 1:length(lambdas)) {
  base_fc[[i]] <- list(lambda = lambdas[i])
}

# Sample from the reconciled forecast distribution using MCMC
mcmc <- reconc_MCMC(A, base_fc,
  distr = "poisson",
  num_samples = 30000, seed = 42
)
samples_mcmc <- rbind(mcmc$upper_rec_samples, mcmc$bottom_rec_samples)

# Compare the reconciled means with those obtained via BUIS
buis <- reconc_BUIS(A, base_fc,
  in_type = "params",
  distr = "poisson", num_samples = 100000, seed = 42
)
samples_buis <- rbind(buis$upper_rec_samples, buis$bottom_rec_samples)

print(rowMeans(samples_mcmc))
print(rowMeans(samples_buis))
```

`reconc_t` *t-Rec: reconciliation via conditioning with uncertain covariance via multivariate t-distribution*

Description

Reconciles base forecasts in a hierarchy by conditioning on the hierarchical constraints, specified by the aggregation matrix A. The base forecasts are assumed to be jointly Gaussian, conditionally on the covariance matrix of the forecast errors. A Bayesian approach is adopted to account for the uncertainty of the covariance matrix. An Inverse-Wishart prior is specified on the covariance matrix, leading to a multivariate t-distribution for the base forecasts. The reconciliation via conditioning is in closed-form, yielding a multivariate t reconciled distribution.

Usage

```
reconc_t(
  A,
  base_fc_mean,
  y_train = NULL,
  residuals = NULL,
  ...,
  return_upper = FALSE,
  return_parameters = FALSE
)
```

Arguments

A	Matrix (n_upp x n_bott) defining the hierarchy (u = Ab).
base_fc_mean	Vector of base forecasts (length n = n_upp + n_bott).
y_train	mts (or matrix) of historical training data (T x n) used for setting prior parameters.
residuals	Matrix (T x n) of base forecast residuals.
...	Additional arguments for advanced usage: see details.
return_upper	Logical; if TRUE, also returns parameters for the upper-level reconciled distribution.
return_parameters	Logical; if TRUE, also returns prior and posterior parameters.

Details

Standard usage.

The standard workflow for this function is to provide the in-sample `residuals` and the historical training data `y_train`. The parameters of the Inverse-Wishart prior distribution of the covariance matrix are set as follows:

- Prior scale matrix (Psi): set as the covariance of the residuals of naive (or seasonal naive, a criterion is used to choose between the two) forecasts computed on `y_train`.
- Prior degrees of freedom (nu): estimated via Bayesian Leave-One-Out Cross-Validation (LOOCV) to maximize out-of-sample performance.

The posterior distribution of the covariance matrix is still Inverse-Wishart. The parameters of the posterior are computed in closed-form using the sample covariance of the provided `residuals`.

Advanced Options.

Users can bypass the automated estimation by specifying:

1. `prior`: a list with entries 'nu' and 'Psi'. This skips the LOOCV step for ν_0 and the covariance estimation from `y_train`. It requires `residuals` to compute the posterior.
2. `posterior`: a list with entries 'nu' and 'Psi'. This skips all internal estimation and updating logic.

Moreover, users can specify:

- `freq`: positive integer, used as frequency of data for the seasonal naive forecast in the specification of the prior scale matrix. By default, if `y_train` is a multivariate time series, the frequency of the data is used; otherwise, it is set to 1 (no seasonality).
- `criterion`: either 'RSS' (default) or 'seas-test', specifying which criterior is used to choose between the naive and seasonal naive forecasts for the specification of the prior scale matrix. 'RSS' computes the residual sum of squares for both methods and chooses the one with lower RSS, while 'seas-test' uses a statistical test for seasonality (currently implemented using the number of seasonal differences suggested by the `forecast` package, which must be installed).

Reconciled distribution.

The reconciled distribution is a multivariate t-distribution, specified by a vector of means, a scale matrix, and a number of degrees of freedom. These parameters are computed in closed-form. By default, only the parameters of the reconciled distribution for the bottom-level series are returned. See examples.

Value

A list containing:

- `bottom_rec_mean`: reconciled bottom-level mean.
- `bottom_rec_scale_matrix`: reconciled bottom-level scale matrix.
- `bottom_rec_df`: reconciled degrees of freedom.

If `return_upper` is TRUE, also returns:

- `upper_rec_mean`: reconciled upper-level mean.
- `upper_rec_scale_matrix`: reconciled upper-level scale matrix.
- `upper_rec_df`: reconciled upper-level degrees of freedom.

If `return_parameters` is TRUE, also returns:

- `prior_nu`: prior degrees of freedom.
- `prior_Psi`: prior scale matrix.
- `posterior_nu`: posterior degrees of freedom.
- `posterior_Psi`: posterior scale matrix.

References

Carrara, C., Corani, G., Azzimonti, D., & Zambon, L. (2025). Modeling the uncertainty on the covariance matrix for probabilistic forecast reconciliation. arXiv preprint arXiv:2506.19554. https://arxiv.org/abs/2506.19554

See Also

`reconc_gaussian()`

Examples

```
library(bayesRecon)

if (requireNamespace("forecast", quietly = TRUE)) {

  set.seed(1234)
  n_obs <- 100

  # Simulate 2 bottom series from AR(1) processes
  y1 <- arima.sim(model = list(ar = 0.8), n = n_obs)
  y2 <- arima.sim(model = list(ar = 0.5), n = n_obs)

  y_upper <- y1 + y2   # upper series is the sum of the two bottoms
  A <- matrix(c(1, 1), nrow = 1)  # Aggregation matrix

  # Fit additive ETS models
  fit1 <- forecast::ets(y1, additive.only = TRUE)
  fit2 <- forecast::ets(y2, additive.only = TRUE)
  fit_upper <- forecast::ets(y_upper, additive.only = TRUE)

  # Point forecasts (h = 1)
  fc_upper <- as.numeric(forecast::forecast(fit_upper, h = 1)$mean)
  fc1 <- as.numeric(forecast::forecast(fit1, h = 1)$mean)
  fc2 <- as.numeric(forecast::forecast(fit2, h = 1)$mean)
  base_fc_mean <- c(fc_upper, fc1, fc2)

 # Residuals and training data (n_obs x n matrices, columns in same order as base_fc_mean)
  res <- cbind(residuals(fit_upper), residuals(fit1), residuals(fit2))
  y_train <- cbind(y_upper, y1, y2)

  # --- 1) Generate joint reconciled samples ---
  result <- reconc_t(A, base_fc_mean, y_train = y_train, residuals = res)

  # Sample from the reconciled bottom-level t-distribution
  n_samples <- 2000
  L_chol <- t(chol(result$bottom_rec_scale_matrix))
  z <- matrix(rt(ncol(A) * n_samples, df = result$bottom_rec_df), nrow = ncol(A))
  bottom_samples <- result$bottom_rec_mean + L_chol %*% z  # 2 x n_samples

  # Aggregate bottom samples to get upper samples
  upper_samples <- A %*% bottom_samples
  joint_samples <- rbind(upper_samples, bottom_samples)
  rownames(joint_samples) <- c("upper", "bottom_1", "bottom_2")

  cat("Reconciled means (from samples):\n")
  print(round(rowMeans(joint_samples), 3))

  cat("Reconciled standard deviations (from samples):\n")
  print(round(apply(joint_samples, 1, sd), 3))

  # --- 2) 95% prediction intervals via t-distribution quantiles ---
```

```
  result2 <- reconc_t(A, base_fc_mean, y_train = y_train,
                      residuals = res, return_upper = TRUE)

  alpha <- 0.05
  # Bottom series intervals
  for (i in seq_len(ncol(A))) {
    s_i <- sqrt(result2$bottom_rec_scale_matrix[i, i])
   lo  <- result2$bottom_rec_mean[i] + s_i * qt(alpha / 2,     df = result2$bottom_rec_df)
   hi  <- result2$bottom_rec_mean[i] + s_i * qt(1 - alpha / 2, df = result2$bottom_rec_df)
    cat(sprintf("Bottom %d: 95%% PI = [%.3f, %.3f]\n", i, lo, hi))
  }
  # Upper series interval
  s_u <- sqrt(result2$upper_rec_scale_matrix[1, 1])
  lo  <- result2$upper_rec_mean[1] + s_u * qt(alpha / 2,     df = result2$upper_rec_df)
  hi  <- result2$upper_rec_mean[1] + s_u * qt(1 - alpha / 2, df = result2$upper_rec_df)
  cat(sprintf("Upper:    95%% PI = [%.3f, %.3f]\n", lo, hi))
}
```

`schaferStrimmer_cov` *Schäfer Strimmer covariance shrinkage*

Description

Computes the Schäfer Strimmer shrinkage estimator for a covariance matrix from a matrix of samples.

Usage

```
schaferStrimmer_cov(x)
```

Arguments

`x`	matrix of samples with dimensions nxp (n samples, p dimensions).

Details

This function computes the shrinkage to a diagonal covariance with unequal variances. Note that here we use the estimators $S = XX^T/n$ and $T = diag(S)$ and we internally use the correlation matrix in place of the covariance to compute the optimal shrinkage factor.

Value

A list containing the shrinkage estimator and the optimal lambda. The list has the following named elements:

- `shrink_cov`: the shrinked covariance matrix (p x p);
- `lambda_star`: the optimal lambda for the shrinkage;

References

Schäfer, Juliane, and Korbinian Strimmer. (2005). *A Shrinkage Approach to Large-Scale Covariance Matrix Estimation and Implications for Functional Genomics.* Statistical Applications in Genetics and Molecular Biology 4: Article32. doi:10.2202/15446115.1175.

Examples

```
# Generate some multivariate normal samples
# Parameters
nSamples <- 200
pTrue <- 2

# True moments
true_cov <- matrix(c(3, 2, 2, 2), nrow = 2)
chol_true_cov <- chol(true_cov)
true_mean <- c(0, 0)

# Generate samples
set.seed(42)
x <- replicate(nSamples, true_mean) +
  t(chol_true_cov) %*% matrix(stats::rnorm(pTrue * nSamples),
    nrow = pTrue, ncol = nSamples
  )
x <- t(x)
res_shrinkage <- schaferStrimmer_cov(x)
res_shrinkage$lambda_star # should be 0.01287923
```

`swiss_tourism` *Swiss tourism: monthly Swiss overnight stay data*

Description

A dataset of 27 time series with monthly observations of the number of overnight stays in Switzerland, from Jan 2005 to Jan 2025.

Usage

```
swiss_tourism
```

Format

A list that contains:

- `ts`: a multivariate time series of class ts;
- `n_bottom`: integer with the number of bottom time series (26);
- `n_upper`: integer with the number of upper time series (1);
- `agg_mat`: the aggregation matrix for the hierarchy. (1x26)

Details

This is a hierarchy of time series that contains:

- 26 bottom-level series corresponding to the 26 Swiss cantons
- 1 top-level series corresponding to the Swiss total. Each time series has 241 observations.

Source

Federal Statistical Office of Switzerland (2025). *Hotel sector: arrivals and overnight stays of open establishments by year, month, canton and visitors' country of residence*. https://www.bfs.admin.ch/asset/en/px-x-1003020000_102

References

Carrara, C., Azzimonti, D., Corani, G., Zambon, L. (2025). *Modeling the uncertainty on the covariance matrix for probabilistic forecast reconciliation*. arXiv:2506.19554 doi:10.48550/arXiv.2506.19554.

---

temporal_aggregation *Temporal aggregation of a time series*

Description

Creates a list of aggregated time series from a time series of class ts.

Usage

```
temporal_aggregation(y, agg_levels = NULL)
```

Arguments

y	univariate time series of class ts.
agg_levels	user-selected list of aggregation levels.

Details

If `agg_levels=NULL` then `agg_levels` is automatically generated by taking all the factors of the time series frequency.

Value

A list of ts objects each containing the aggregates time series in the order defined by `agg_levels`.

See Also

`get_reconc_matrices()`

Examples

```
# Create a monthly count time series with 100 observations
y <- ts(data = stats::rpois(100, lambda = 2), frequency = 12)

# Create the aggregate time series according to agg_levels
y_agg <- temporal_aggregation(y, agg_levels = c(2, 3, 4, 6, 12))

# Show annual aggregate time series
print(y_agg$`f=1`)
```

Index