Package: bayesRecon (via r-universe)

August 29, 2024

Type Package

Date 2024-08-28

Title Probabilistic Reconciliation via Conditioning

Version 0.3.1

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. The 40th Conference on Uncertainty in Artificial Intelligence, accepted).

License LGPL (>= 3)

Depends R (>= 4.1.0)

Imports stats, utils, lpSolve (>= 5.6.18)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

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

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

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

Date/Publication 2024-08-28 20:10:02 UTC

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

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

RemoteRef HEAD

RemoteSha e9d90fa7a9b5d86ea8f6fa864517c72efcab5371

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.

`PMF.get_mean` *Get the mean of the distribution from a PMF object*

Description

Returns the mean from the PMF specified by `pmf`.

Usage

```
PMF.get_mean(pmf)
```

Arguments

`pmf`	the PMF object.

Value

A numerical value for mean of the distribution.

See Also

`PMF.get_var()`, `PMF.get_quantile()`, `PMF.sample()`, `PMF.summary()`

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(0,10)),MARGIN=1,FUN=function(x) dbinom(x,size=n,prob=p))
```

```
# The true mean corresponds to n*p
true_mean <- n*p
mean_from_PMF <- PMF.get_mean(pmf=pmf_binomial)
cat("True mean:", true_mean, "\nMean from PMF:", mean_from_PMF)
```

`PMF.get_quantile` *Get quantile from a PMF object*

Description

Returns the p quantile from the PMF specified by `pmf`.

Usage

```
PMF.get_quantile(pmf, p)
```

Arguments

`pmf`	the PMF object.
`p`	the probability of the required quantile.

Value

A numeric value for the quantile.

See Also

`PMF.get_mean()`, `PMF.get_var()`, `PMF.sample()`, `PMF.summary()`

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(0,10)),MARGIN=1,FUN=function(x) dbinom(x,size=n,prob=p))

# The true median is ceiling(n*p)
quant_50 <- PMF.get_quantile(pmf=pmf_binomial,p=0.5)
cat("True median:", ceiling(n*p), "\nMedian from PMF:", quant_50)
```

PMF.get_var — *Get the variance of the distribution from a PMF object*

Description

Returns the variance from the PMF specified by `pmf`.

Usage

```
PMF.get_var(pmf)
```

Arguments

`pmf`	the PMF object.

Value

A numerical value for variance.

See Also

`PMF.get_mean()`, `PMF.get_quantile()`, `PMF.sample()`, `PMF.summary()`

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(0,10)),MARGIN=1,FUN=function(x) dbinom(x,size=n,prob=p))

# The true variance corresponds to n*p*(1-p)
true_var <- n*p*(1-p)
var_from_PMF <- PMF.get_var(pmf=pmf_binomial)
cat("True variance:", true_var, "\nVariance from PMF:", var_from_PMF)
```

PMF.sample — *Sample from the distribution given as a PMF object*

Description

Samples (with replacement) from the probability distribution specified by `pmf`.

Usage

```
PMF.sample(pmf, N_samples)
```

Arguments

pmf	the PMF object.
N_samples	number of samples.

Value

Samples drawn from the distribution specified by pmf.

See Also

PMF.get_mean(), PMF.get_var(), PMF.get_quantile(), PMF.summary()

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=function(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)

# Plot the histogram computed with the samples and the true value of the PMF
hist(samples,breaks=seq(0,n),freq=FALSE)
points(seq(0,n)-0.5,pmf_binomial,pch=16)
```

PMF.summary *Returns summary of a PMF object*

Description

Returns the summary (min, max, IQR, median, mean) of the PMF specified by pmf.

Usage

```
PMF.summary(pmf, Ltoll = .TOLL, Rtoll = .RTOLL)
```

Arguments

pmf	the PMF object.
Ltoll	used for computing the min of the PMF: the min is the smallest value with probability greater than Ltoll (default: 1e-15)
Rtoll	used for computing the max of the PMF: the max is the largest value with probability greater than Rtoll (default: 1e-9)

Value

A summary data.frame

See Also

PMF.get_mean(), PMF.get_var(), PMF.get_quantile(), PMF.sample()

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(0,10)),MARGIN=1,FUN=function(x) dbinom(x,size=n,prob=p))

# Print the summary of this distribution
PMF.summary(pmf=pmf_binomial)
```

---

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

Arguments

A	aggregation matrix (n_upper x n_bottom).
base_forecasts	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.

`seed` Seed for reproducibility.

Details

The parameter `base_forecast` 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_forecast[[i]]` is a vector containing samples from the base forecast distribution.

If `in_type[[i]]='params'`, then `base_forecast[[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_reconciled_samples`: a matrix (n_bottom x `num_samples`) containing the reconciled samples for the bottom time series;
- `upper_reconciled_samples`: a matrix (n_upper x `num_samples`) containing the reconciled samples for the upper time series;
- `reconciled_samples`: a matrix (n x `num_samples`) containing the reconciled samples for all 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_forecasts = list()
for (i in 1:length(mus)) {
base_forecasts[[i]] = list(mean = mus[[i]], sd = sigmas[[i]])
}


#Sample from the reconciled forecast distribution using the BUIS algorithm
buis <- reconc_BUIS(A, base_forecasts, in_type="params",
```

```
                  distr="gaussian", num_samples=100000, seed=42)

samples_buis <- buis$reconciled_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_forecasts.mu = mus,
                                base_forecasts.Sigma = Sigma)

#Compare the reconciled means obtained analytically and via BUIS
print(c(S %*% analytic_rec$bottom_reconciled_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_forecasts <- list()
for (i in 1:length(lambdas)) {
 base_forecasts[[i]] = list(lambda = lambdas[i])
}

#Sample from the reconciled forecast distribution using the BUIS algorithm
buis <- reconc_BUIS(A, base_forecasts, in_type="params",
                          distr="poisson", num_samples=100000, seed=42)
samples_buis <- buis$reconciled_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_forecasts.mu, base_forecasts.Sigma)
```

Arguments

`A`	aggregation matrix (n_upper x n_bottom).
`base_forecasts.mu`	a vector containing the means of the base forecasts.
`base_forecasts.Sigma`	a matrix containing the covariance matrix of the base forecasts.

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.

The function returns only the reconciled parameters of the bottom variables. The reconciled upper parameters and the reconciled samples for the entire hierarchy can be obtained from the reconciled bottom parameters. See the example section.

Value

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

- `bottom_reconciled_mean`: reconciled mean for the bottom forecasts;
- `bottom_reconciled_covariance`: reconciled covariance for the bottom 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_BUIS()`

Examples

```
library(bayesRecon)

# 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
mus <- c(muY,mu1,mu2)
```

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

Sigma <- diag(sigmas^2)  # need to transform into covariance matrix
analytic_rec <- reconc_gaussian(A, base_forecasts.mu = mus,
                                base_forecasts.Sigma = Sigma)

bottom_mu_reconc <- analytic_rec$bottom_reconciled_mean
bottom_Sigma_reconc <- analytic_rec$bottom_reconciled_covariance

# Obtain reconciled mu and Sigma for the upper variable
upper_mu_reconc <- A %*% bottom_mu_reconc
upper_Sigma_reconc <- A %*% bottom_Sigma_reconc %*% t(A)

# Obtain reconciled mu and Sigma for the entire hierarchy
S <- rbind(A, diag(2))  # first, get summing matrix S
Y_mu_reconc <- S %*% bottom_mu_reconc
Y_Sigma_reconc <- S %*% bottom_Sigma_reconc %*% t(S)  # note that this is a singular matrix

# Obtain reconciled samples for the entire hierarchy:
# i.e., sample from the reconciled bottoms and multiply by S
chol_decomp = chol(bottom_Sigma_reconc) # Compute the Cholesky Decomposition
Z = matrix(stats::rnorm(n = 2000), nrow = 2) # Sample from standard normal
B = t(chol_decomp) %*% Z + matrix(rep(bottom_mu_reconc, 1000), nrow=2) # Apply the transformation

U = S %*% B
Y_reconc = rbind(U, B)
```

`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_forecasts,
  distr,
```

```
  num_samples = 10000,
  tuning_int = 100,
  init_scale = 1,
  burn_in = 1000,
  seed = NULL
)
```

Arguments

A	aggregation matrix (n_upper x n_bottom).
base_forecasts	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.
seed	seed for reproducibility.

Details

The parameter `base_forecast` 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_reconciled_samples`: a matrix (n_bottom x `num_samples`) containing reconciled samples for the bottom time series;
- `upper_reconciled_samples`: a matrix (n_upper x `num_samples`) containing reconciled samples for the upper time series;
- `reconciled_samples`: a matrix (n x `num_samples`) containing the reconciled samples for all time series.

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_forecasts = list()
for (i in 1:length(lambdas)) {
 base_forecasts[[i]] = list(lambda = lambdas[i])
}

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

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

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

---

reconc_MixCond *Probabilistic forecast reconciliation of mixed hierarchies via conditioning*

Description

Uses importance sampling to draw samples from the reconciled forecast distribution, obtained via conditioning, in the case of a mixed hierarchy.

Usage

```
reconc_MixCond(
  A,
```

```
  fc_bottom,
  fc_upper,
  bottom_in_type = "pmf",
  distr = NULL,
  num_samples = 20000,
  return_type = "pmf",
  suppress_warnings = FALSE,
  seed = NULL
)
```

Arguments

A	Aggregation matrix (n_upper x n_bottom).
fc_bottom	A list containing the bottom base forecasts, see details.
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.
suppress_warnings	Logical. If TRUE, no warnings about samples are triggered. If FALSE, warnings are generated. Default is FALSE. See Details.
seed	Seed for reproducibility.

Details

The base bottom forecasts `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 `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 `mu` (vector of length n_upper) and `Sigma` (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.get_mean, PMF.get_var, PMF.sample, PMF.get_quantile, PMF.summary for functions that handle PMF objects.

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.

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_reconciled`: a list containing the pmf, the samples (matrix n_bottom x `num_samples`) or both, depending on the value of `return_type`;
- `upper_reconciled`: a list containing the pmf, the samples (matrix n_upper x `num_samples`) or both, depending on the value of `return_type`.

References

Zambon, L., Azzimonti, D., Rubattu, N., Corani, G. (2024). *Probabilistic reconciliation of mixed-type hierarchical time series*. The 40th Conference on Uncertainty in Artificial Intelligence, accepted.

See Also

`reconc_TDcond()`, `reconc_BUIS()`

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
fc_bottom <- list()
```

```
fc_bottom[[1]] <- apply(matrix(seq(0,n_tot)),MARGIN=1,FUN=function(x) dpois(x,lambda=lambda))
fc_bottom[[2]] <- apply(matrix(seq(0,n_tot)),MARGIN=1,FUN=function(x) dpois(x,lambda=lambda))

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

# We can reconcile with reconc_MixCond
res.mixCond <- reconc_MixCond(A, fc_bottom, fc_upper)

# Note that the bottom distributions are slightly shifted to the right
PMF.summary(res.mixCond$bottom_reconciled$pmf[[1]])
PMF.summary(fc_bottom[[1]])

PMF.summary(res.mixCond$bottom_reconciled$pmf[[2]])
PMF.summary(fc_bottom[[2]])

# The upper distribution is slightly shifted to the left
PMF.summary(res.mixCond$upper_reconciled$pmf[[1]])
PMF.get_var(res.mixCond$upper_reconciled$pmf[[1]])
```

`reconc_TDcond` *Probabilistic forecast reconciliation of mixed hierarchies via top-down conditioning*

Description

Uses the top-down conditioning algorithm to draw samples from the reconciled forecast distribution. Reconciliation is performed in two steps: first, the upper base forecasts are reconciled via conditioning, using only the hierarchical constraints between the upper variables; then, the bottom distributions are updated via a probabilistic top-down procedure.

Usage

```
reconc_TDcond(
  A,
  fc_bottom,
  fc_upper,
  bottom_in_type = "pmf",
  distr = NULL,
  num_samples = 20000,
  return_type = "pmf",
  suppress_warnings = FALSE,
  seed = NULL
)
```

Arguments

A	aggregation matrix (n_upper x n_bottom).
fc_bottom	A list containing the bottom base forecasts, see details.
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.
suppress_warnings	Logical. If TRUE, no warnings about samples are triggered. If FALSE, warnings are generated. Default is FALSE. See Details.
seed	Seed for reproducibility.

Details

The base bottom forecasts 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 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 mu (vector of length n_upper) and Sigma (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.get_mean, PMF.get_var, PMF.sample, PMF.get_quantile, PMF.summary for functions that handle PMF objects.

If some of the reconciled upper samples lie outside the support of the bottom-up distribution, those samples are discarded and a warning is triggered. The warning reports the percentage of samples kept.

Value

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

- `bottom_reconciled`: a list containing the pmf, the samples (matrix n_bottom x `num_samples`) or both, depending on the value of `return_type`;
- `upper_reconciled`: a list containing the pmf, the samples (matrix n_upper x `num_samples`) or both, depending on the value of `return_type`.

References

Zambon, L., Azzimonti, D., Rubattu, N., Corani, G. (2024). *Probabilistic reconciliation of mixed-type hierarchical time series*. The 40th Conference on Uncertainty in Artificial Intelligence, accepted.

See Also

`reconc_MixCond()`, `reconc_BUIS()`

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
fc_bottom <- list()
fc_bottom[[1]] <- apply(matrix(seq(0,n_tot)),MARGIN=1,FUN=function(x) dpois(x,lambda=lambda))
fc_bottom[[2]] <- apply(matrix(seq(0,n_tot)),MARGIN=1,FUN=function(x) dpois(x,lambda=lambda))

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

# We can reconcile with reconc_TDcond
res.TDcond <- reconc_TDcond(A, fc_bottom, fc_upper)

# Note that the bottom distributions are shifted to the right
PMF.summary(res.TDcond$bottom_reconciled$pmf[[1]])
PMF.summary(fc_bottom[[1]])

PMF.summary(res.TDcond$bottom_reconciled$pmf[[2]])
PMF.summary(fc_bottom[[2]])
```

```
# The upper distribution remains similar
PMF.summary(res.TDcond$upper_reconciled$pmf[[1]])
PMF.get_var(res.TDcond$upper_reconciled$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
fc_bottom <- list()
for(i in seq(n_bottom)){
 fc_bottom[[i]] <- apply(matrix(seq(0,n_tot)),MARGIN=1,FUN=function(x) dpois(x,lambda=lambda))
}

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

fc_upper<- list(mu=mu, Sigma=Sigma)
## Not run:
# If we reconcile with reconc_TDcond it won't work
res.TDcond <- reconc_TDcond(A, fc_bottom, fc_upper)

## End(Not run)

# We can balance the hierarchy with by duplicating the node b5
# In practice this means:
# 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
mu = c(75, 30, 30,mean_b5)
Sigma = matrix(c(5^2,5,5,5,
                 5, 10, 0, 0,
                 5, 0, 10, 0,
                 5, 0,  0, var_b5), nrow=4, byrow = TRUE)
fc_upper<- list(mu=mu, Sigma=Sigma)

# 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, fc_bottom, 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$bottom_reconciled$pmf[[5]]- res.TDcond$upper_reconciled$pmf[[4]]))
```

`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
trueSigma <- matrix(c(3,2,2,2), nrow=2)
chol_trueSigma <- chol(trueSigma)
trueMean <- c(0,0)

# Generate samples
set.seed(42)
x <- replicate(nSamples, trueMean) +
     t(chol_trueSigma)%*%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
```

---

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