Package 'FoRecoML'

May 7, 2026

Type Package

Title Forecast Reconciliation with Machine Learning

Version 1.0.0

Description Nonlinear forecast reconciliation with machine learning in cross-sectional (Spiliotis et al. 2021 <doi:10.1016/j.asoc.2021.107756>), temporal, and cross-temporal (Rombouts et al. 2024 <doi:10.1016/j.ijforecast.2024.05.008>) frameworks.

Depends R (>= 3.4), Matrix, FoReco

Imports stats, cli, methods, randomForest, lightgbm, xgboost, mlr3, mlr3tuning, mlr3learners, paradox

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.3

URL https://github.com/danigiro/FoRecoML, https://danigiro.github.io/FoRecoML/

BugReports https://github.com/danigiro/FoRecoML/issues

Suggests testthat (>= 3.0.0), ranger

Config/testthat/edition 3

NeedsCompilation no

Author Daniele Girolimetto [aut, cre] (ORCID: <https://orcid.org/0000-0001-9387-1232>), Yangzhuoran Fin Yang [aut] (ORCID: <https://orcid.org/0000-0002-1232-8017>), Jeroen Rombouts [aut] (ORCID: <https://orcid.org/0000-0003-2255-4875>), Ines Wilms [aut] (ORCID: <https://orcid.org/0000-0003-3269-4601>)

Maintainer Daniele Girolimetto <daniele.girolimetto@unipd.it>

Repository CRAN

Date/Publication 2026-04-21 09:12:09 UTC

Contents

`FoRecoML-package` *FoRecoML: Forecast Reconciliation with Machine Learning*

Description

Nonlinear forecast reconciliation with machine learning in cross-sectional (Spiliotis et al. 2021 doi:10.1016/j.asoc.2021.107756), temporal, and cross-temporal (Rombouts et al. 2024 doi:10.1016/j.ijforecast.2024.05.008) frameworks.

Author(s)

Maintainer: Daniele Girolimetto <daniele.girolimetto@unipd.it> (ORCID)

Authors:

- Yangzhuoran Fin Yang <yangyangzhuoran@gmail.com> (ORCID)
- Jeroen Rombouts <rombouts@essec.edu> (ORCID)
- Ines Wilms <i.wilms@maastrichtuniversity.nl> (ORCID)

See Also

Useful links:

- https://github.com/danigiro/FoRecoML
- https://danigiro.github.io/FoRecoML/
- Report bugs at https://github.com/danigiro/FoRecoML/issues

csrml *Cross-sectional Reconciliation with Machine Learning*

Description

This function performs machine-learning–based cross-sectional forecast reconciliation for linearly constrained (e.g., hierarchical/grouped) multiple time series (Spiliotis et al., 2021). Reconciled forecasts are obtained by training non-linear predictive models (e.g., random forests, gradient boosting) that learn mappings from base forecasts across all series to bottom-level series values. Coherent forecasts for the entire hierarchy are then derived by aggregating the reconciled bottom-level forecasts through the summing constraints. While the approach is designed for hierarchical and grouped structures, in the case of general linearly constrained time series it can be applied within the broader reconciliation framework described by Girolimetto and Di Fonzo (2024).

Usage

```
# Reconciled forecasts
csrml(base, hat, obs, agg_mat, features = "all", approach = "randomForest",
      params = NULL, tuning = NULL, sntz = FALSE, round = FALSE, fit = NULL)

# Pre-trained reconciled ML models
csrml_fit(hat, obs, agg_mat, features = "all", approach = "randomForest",
          params = NULL, tuning = NULL)
```

Arguments

base	A ($h \times n$) numeric matrix or multivariate time series (`mts` class) containing the base forecasts to be reconciled; h is the forecast horizon, and n is the total number of time series ($n = n_a + n_b$).
hat	A ($N \times n$) numeric matrix containing the base forecasts to train the ML approach; N is the training length.
obs	A ($N \times n_b$) numeric matrix containing (observed) values to train the ML approach; n_b is the total number of bottom variables.
agg_mat	A ($n_a \times n_b$) numeric matrix representing the cross-sectional aggregation matrix. It maps the n_b bottom-level (free) variables into the n_a upper (constrained) variables.
features	Character string specifying which features are used for model training. Options include "`bts`" (only bottom-level series as features), `str` (features based on the structural matrix), "`str-bts`" (`bts` + `str` features), and "`all`" (all series as features, *default*).
approach	Character string specifying the machine learning method used for reconciliation. Options are: • "`randomForest`" (*default*): Random Forest algorithm (see the **randomForest** package). • "`xgboost`": Extreme Gradient Boosting (see the **xgboost** package).

- "lightgbm": Light Gradient Boosting Machine (see the **lightgbm** package).
- "mlr3": Any regression learner available in the **mlr3** package. The learner must be specified via params, e.g. params = list(.key = "regr.ranger").

params	Optional list of additional parameters passed to the chosen ML approach These may include algorithm-specific hyperparameters for **randomForest**, **xgboost**, **lightgbm**, or learner options for **mlr3**. When approach = "mlr3", the list must include .key to select the learner (e.g. .key = "regr.ranger", *default*).
tuning	Optional list specifying tuning options when using the mlr3tuning::mlr3tuning framework (e.g., terminators, search spaces). The argument format follows mlr3tuning::auto_tuner, except that the learner is set through params.
sntz	Logical. If TRUE, enforces non-negativity on reconciled forecasts using the heuristic "set-negative-to-zero" (Di Fonzo and Girolimetto, 2023). *Default* is FALSE.
round	Logical. If TRUE, reconciled forecasts are rounded to integer values and coherence is ensured via a bottom-up adjustment. *Default* is FALSE.
fit	A pre-trained ML reconciliation model (see, extract_reconciled_ml). If supplied, training data (hat, obs) are not required.

Value

- csrml returns a cross-sectional reconciled forecast matrix with the same dimensions, along with attributes containing the fitted model and reconciliation settings (see, FoReco::recoinfo and extract_reconciled_ml).
- csrml_fit returns a fitted object that can be reused for reconciliation on new base forecasts.

References

Di Fonzo, T. and Girolimetto, D. (2023), Spatio-temporal reconciliation of solar forecasts, *Solar Energy*, 251, 13–29. doi:10.1016/j.solener.2023.01.003

Girolimetto, D. and Di Fonzo, T. (2023), Point and probabilistic forecast reconciliation for general linearly constrained multiple time series, *Statistical Methods & Applications*, 33, 581-607. doi:10.1007/s10260023007386.

Spiliotis, E., Abolghasemi, M., Hyndman, R. J., Petropoulos, F., and Assimakopoulos, V. (2021). Hierarchical forecast reconciliation with machine learning. *Applied Soft Computing*, 112, 107756. doi:10.1016/j.asoc.2021.107756

Examples

```
# agg_mat: simple aggregation matrix, A = B + C
agg_mat <- t(c(1,1))
dimnames(agg_mat) <- list("A", c("B", "C"))

# N_hat: dimension for the most aggregated training set
N_hat <- 100

# ts_mean: mean for the Normal draws used to simulate data
```

```
ts_mean <- c(20, 10, 10)

# hat: a training (base forecasts) feautures matrix
hat <- matrix(rnorm(length(ts_mean)*N_hat, mean = ts_mean),
              N_hat, byrow = TRUE)
colnames(hat) <- unlist(dimnames(agg_mat))

# obs: (observed) values for bottom-level series (B, C)
obs <- matrix(rnorm(length(ts_mean[-1])*N_hat, mean = ts_mean[-1]),
              N_hat, byrow = TRUE)
colnames(obs) <- colnames(agg_mat)

# h: base forecast horizon
h <- 2

# base: base forecasts matrix
base <- matrix(rnorm(length(ts_mean)*h, mean = ts_mean),
               h, byrow = TRUE)
colnames(base) <- unlist(dimnames(agg_mat))

##############################################################################
# Different ML approaches
##############################################################################
# XGBoost Reconciliation (xgboost pkg)
reco <- csrml(base = base, hat = hat, obs = obs, agg_mat = agg_mat,
              approach = "xgboost", features = "all")

# XGBoost Reconciliation with Tweedie loss function (xgboost pkg)
reco <- csrml(base = base, hat = hat, obs = obs, agg_mat = agg_mat,
              approach = "xgboost", features = "all",
              params =  list(
                eta = 0.3, colsample_bytree = 1, min_child_weight = 1,
                max_depth = 6, gamma = 0, subsample = 1,
                objective = "reg:tweedie", # Tweedie regression objective
                tweedie_variance_power = 1.5 # Tweedie power parameter
              ))

# LightGBM Reconciliation (lightgbm pkg)
reco <- csrml(base = base, hat = hat, obs = obs, agg_mat = agg_mat,
              approach = "lightgbm", features = "all")

# Random Forest Reconciliation (randomForest pkg)
reco <- csrml(base = base, hat = hat, obs = obs, agg_mat = agg_mat,
              approach = "randomForest", features = "all")

# Using the mlr3 pkg:
# With 'params = list(.key = mlr_learners)' we can specify different
# mlr_learners implemented in mlr3 such as "regr.ranger" for Random Forest,
# "regr.xgboost" for XGBoost, and others.
reco <- csrml(base = base, hat = hat, obs = obs, agg_mat = agg_mat,
              approach = "mlr3", features = "all",
              # choose mlr3 learner (here Random Forest via ranger)
              params = list(.key = "regr.ranger"))
```

```
# With mlr3 we can also tune our parameters: e.g. explore mtry in [1,2].
# We can reduce excessive logging by calling:
# if(requireNamespace("lgr", quietly = TRUE)){
#   lgr::get_logger("mlr3")$set_threshold("warn")
#   lgr::get_logger("bbotk")$set_threshold("warn")
# }
reco <- csrml(base = base, hat = hat, obs = obs, agg_mat = agg_mat,
              approach = "mlr3", features = "all",
              params = list(
                .key = "regr.ranger",
                # number of features tried at each split
                mtry = paradox::to_tune(paradox::p_int(1, 2))
              ),
              tuning = list(
                # stop after 10 evaluations
                terminator = mlr3tuning::trm("evals", n_evals = 20)
              ))

##############################################################################
# Usage with pre-trained models
##############################################################################
# Pre-trained machine learning models (e.g., omit the base param)
mdl <- csrml_fit(hat = hat, obs = obs, agg_mat = agg_mat,
                 approach = "xgboost", features = "all")

# Pre-trained machine learning models with base param
reco <- csrml(base = base, hat = hat, obs = obs, agg_mat = agg_mat,
              approach = "xgboost", features = "all")
mdl2 <- extract_reconciled_ml(reco)

# New base forecasts matrix
base_new <- matrix(rnorm(length(ts_mean)*h, mean = ts_mean), h, byrow = TRUE)
reco_new <- csrml(base = base_new, fit = mdl, agg_mat = agg_mat)
```

`ctrml` *Cross-temporal Reconciliation with Machine Learning*

Description

This function performs machine-learning–based cross-temporal forecast reconciliation for linearly constrained multiple time series (Rombouts et al., 2024). Reconciled forecasts are obtained by fitting non-linear models that map base forecasts across both temporal and cross-sectional dimensions to bottom-level high-frequency series. Fully coherent forecasts across all temporal and cross-sectional linear combinations are then derived by cross-temporal bottom-up. While the approach is designed for hierarchical and grouped structures, in the case of general linearly constrained time series it can be applied within the broader reconciliation framework described by Girolimetto and Di Fonzo (2024).

Usage

```
# Reconciled forecasts
ctrml(base, hat, obs, agg_mat, agg_order, tew = "sum", features = "all",
      approach = "randomForest", params = NULL, tuning = NULL,
      sntz = FALSE, round = FALSE, fit = NULL)

# Pre-trained reconciled ML models
ctrml_fit(hat, obs, agg_mat, agg_order, tew = "sum", features = "all",
          approach = "randomForest", params = NULL, tuning = NULL)
```

Arguments

base	A $(n \times h(k^* + m))$ numeric matrix containing the base forecasts to be reconciled; n is the total number of variables, m is the maximum aggregation order, and k^* is the sum of a chosen subset of the $p-1$ factors of m (excluding m itself), and h is the forecast horizon for the lowest frequency time series. The row identifies a time series, and the forecasts in each row are ordered from the lowest frequency (most temporally aggregated) to the highest frequency.
hat	A $(n \times N(k^* + m))$ numeric matrix containing the base forecasts ordered from lowest to highest frequency; N is the training length for the lowest frequency time series. The row identifies a time series, and the forecasts in each row are ordered from the lowest frequency (most temporally aggregated) to the highest frequency. These forecasts are used to train the ML approach.
obs	A $(n_b \times Nm)$ numeric matrix containing (observed) values for the highest frequency series ($k = 1$); n_b is the total number of high-frequency bottom variables. These values are used to train the ML approach.
agg_mat	A $(n_a \times n_b)$ numeric matrix representing the cross-sectional aggregation matrix. It maps the n_b bottom-level (free) variables into the n_a upper (constrained) variables.
agg_order	Highest available sampling frequency per seasonal cycle (max. order of temporal aggregation, m), or a vector representing a subset of p factors of m.
tew	A string specifying the type of temporal aggregation. Options include: "sum" (simple summation, *default*), "avg" (average), "first" (first value of the period), and "last" (last value of the period).
features	Character string specifying which features are used for model training. Options include "all" (see Rombouts et al. 2025), and "compact" (see Rombouts et al. 2025, *default*).
approach	Character string specifying the machine learning method used for reconciliation. Options are: • "randomForest" (*default*): Random Forest algorithm (see the **randomForest** package). • "xgboost": Extreme Gradient Boosting (see the **xgboost** package). • "lightgbm": Light Gradient Boosting Machine (see the **lightgbm** package). • "mlr3": Any regression learner available in the **mlr3** package. The learner must be specified via params, e.g. params = list(.key = "regr.ranger").

`params`	Optional list of additional parameters passed to the chosen ML approach These may include algorithm-specific hyperparameters for **randomForest**, **xgboost**, **lightgbm**, or learner options for **mlr3**. When `approach = "mlr3"`, the list must include `.key` to select the learner (e.g. `.key = "regr.ranger"`, *default*).
`tuning`	Optional list specifying tuning options when using the mlr3tuning::mlr3tuning framework (e.g., terminators, search spaces). The argument format follows mlr3tuning::auto_tuner, except that the learner is set through `params`.
`sntz`	Logical. If `TRUE`, enforces non-negativity on reconciled forecasts using the heuristic "set-negative-to-zero" (Di Fonzo and Girolimetto, 2023). *Default* is `FALSE`.
`round`	Logical. If `TRUE`, reconciled forecasts are rounded to integer values and coherence is ensured via a bottom-up adjustment. *Default* is `FALSE`.
`fit`	A pre-trained ML reconciliation model (see, extract_reconciled_ml). If supplied, training data (`hat`, `obs`) are not required.

Value

- ctrml returns a cross-temporal reconciled forecast matrix with the same dimensions, along with attributes containing the fitted model and reconciliation settings (see, FoReco::recoinfo and extract_reconciled_ml).
- ctrml_fit returns a fitted object that can be reused for reconciliation on new base forecasts.

References

Di Fonzo, T. and Girolimetto, D. (2023), Spatio-temporal reconciliation of solar forecasts, *Solar Energy*, 251, 13–29. doi:10.1016/j.solener.2023.01.003

Girolimetto, D. and Di Fonzo, T. (2023), Point and probabilistic forecast reconciliation for general linearly constrained multiple time series, *Statistical Methods & Applications*, 33, 581-607. doi:10.1007/s10260023007386.

Rombouts, J., Ternes, M., and Wilms, I. (2025). Cross-temporal forecast reconciliation at digital platforms with machine learning. *International Journal of Forecasting*, 41(1), 321-344. doi:10.1016/j.ijforecast.2024.05.008

Examples

```
# m: quarterly temporal aggregation order
m <- 4
te_set <- tetools(m)$set

# agg_mat: simple aggregation matrix, A = B + C
agg_mat <- t(c(1,1))
dimnames(agg_mat) <- list("A", c("B", "C"))

# te_fh: minimum forecast horizon per temporal aggregate
te_fh <- m/te_set

# N_hat: dimension for the lowest-frequency (k = m) training set
N_hat <- 16
```

```
# bts_mean: mean for the Normal draws used to simulate data
bts_mean <- 5

# hat: a training (base forecasts) feautures matrix
hat <- rbind(
  rnorm(sum(te_fh)*N_hat, rep(2*te_set*bts_mean, N_hat*te_fh)),  # Series A
  rnorm(sum(te_fh)*N_hat, rep(te_set*bts_mean,  N_hat*te_fh)),   # Series B
  rnorm(sum(te_fh)*N_hat, rep(te_set*bts_mean,  N_hat*te_fh))    # Series C
)
rownames(hat) <- c("A", "B", "C")

# obs: (observed) values for the highest-frequency bottom-level series
# (B and C with k = 1)
obs <- rbind(
  rnorm(m*N_hat, bts_mean),  # Observed for series B
  rnorm(m*N_hat, bts_mean)   # Observed for series C
)
rownames(obs) <- c("B", "C")


# h: base forecast horizon at the lowest-frequency series (k = m)
h <- 2

# base: base forecasts matrix
base <- rbind(
  rnorm(sum(te_fh)*h, rep(2*te_set*bts_mean, h*te_fh)),  # Base for A
  rnorm(sum(te_fh)*h, rep(te_set*bts_mean,  h*te_fh)),   # Base for B
  rnorm(sum(te_fh)*h, rep(te_set*bts_mean,  h*te_fh))    # Base for C
)
rownames(base) <- c("A", "B", "C")

##############################################################################
# Different ML approaches
##############################################################################
# XGBoost Reconciliation (xgboost pkg)
reco <- ctrml(base = base, hat = hat, obs = obs, agg_order = m,
              agg_mat = agg_mat, approach = "xgboost")

# XGBoost Reconciliation with Tweedie loss function (xgboost pkg)
reco <- ctrml(base = base, hat = hat, obs = obs, agg_order = m,
              agg_mat = agg_mat, approach = "xgboost",
              params =  list(
                eta = 0.3, colsample_bytree = 1, min_child_weight = 1,
                max_depth = 6, gamma = 0, subsample = 1,
                objective = "reg:tweedie", # Tweedie regression objective
                tweedie_variance_power = 1.5 # Tweedie power parameter
              ))

# LightGBM Reconciliation (lightgbm pkg)
reco <- ctrml(base = base, hat = hat, obs = obs, agg_order = m,
              agg_mat = agg_mat, approach = "lightgbm")
```

```
# Random Forest Reconciliation (randomForest pkg)
reco <- ctrml(base = base, hat = hat, obs = obs, agg_order = m,
              agg_mat = agg_mat, approach = "randomForest")

# Using the mlr3 pkg:
# With 'params = list(.key = mlr_learners)' we can specify different
# mlr_learners implemented in mlr3 such as "regr.ranger" for Random Forest,
# "regr.xgboost" for XGBoost, and others.
reco <- ctrml(base = base, hat = hat, obs = obs, agg_order = m,
              agg_mat = agg_mat, approach = "mlr3",
              # choose mlr3 learner (here Random Forest via ranger)
              params = list(.key = "regr.ranger"))

# With mlr3 we can also tune our parameters: e.g. explore mtry in [1,4].
# We can reduce excessive logging by calling:
# if(requireNamespace("lgr", quietly = TRUE)){
#   lgr::get_logger("mlr3")$set_threshold("warn")
#   lgr::get_logger("bbotk")$set_threshold("warn")
# }
reco <- ctrml(base = base, hat = hat, obs = obs, agg_order = m,
              agg_mat = agg_mat, approach = "mlr3",
              params = list(
                .key = "regr.ranger",
                # number of features tried at each split
                mtry = paradox::to_tune(paradox::p_int(1, 4))
              ),
              tuning = list(
                # stop after 10 evaluations
                terminator = mlr3tuning::trm("evals", n_evals = 10)
              ))

##############################################################################
# Usage with pre-trained models
##############################################################################
# Pre-trained machine learning models (e.g., omit the base param)
mdl <- ctrml_fit(hat = hat, obs = obs, agg_order = m, agg_mat = agg_mat,
                 approach = "xgboost")

# Pre-trained machine learning models with base param
reco <- ctrml(base = base, hat = hat, obs = obs, agg_order = m,
              agg_mat = agg_mat, approach = "xgboost")
mdl2 <- extract_reconciled_ml(reco)

# New base forecasts matrix
base_new <- rbind(
  rnorm(sum(te_fh)*h, rep(2*te_set*bts_mean, h*te_fh)),  # Base for A
  rnorm(sum(te_fh)*h, rep(te_set*bts_mean,  h*te_fh)),   # Base for B
  rnorm(sum(te_fh)*h, rep(te_set*bts_mean,  h*te_fh))    # Base for C
)
reco_new <- ctrml(base = base_new, fit = mdl, agg_order = m,
                  agg_mat = agg_mat)
```

`extract_reconciled_ml` *Extract the reconciled model from a reconciliation result*

Description

Extract the fitted reconciled model(s) from a reconciliation function's output (e.g., csrml, terml and ctrml). The model can be reused for forecast reconciliation in the reconciliation functions.

Usage

```
extract_reconciled_ml(reco)
```

Arguments

`reco`	An object returned by a reconciliation function (e.g., the result of csrml, terml and ctrml).

Value

A named list with reconciliation information:

`sel_mat`	Features used (e.g., the selected feature matrix or indices).
`fit`	List of reconciled models.
`approach`	The learning approach used (e.g., `"xgboost"`, `"lightgbm"`, `"randomForest"`, `"mlr3"`).

Examples

```
# agg_mat: simple aggregation matrix, A = B + C
agg_mat <- t(c(1,1))
dimnames(agg_mat) <- list("A", c("B", "C"))

# N_hat: dimension for the most aggregated training set
N_hat <- 100

# ts_mean: mean for the Normal draws used to simulate data
ts_mean <- c(20, 10, 10)

# hat: a training (base forecasts) feautures matrix
hat <- matrix(
  rnorm(length(ts_mean)*N_hat, mean = ts_mean),
  N_hat, byrow = TRUE)
colnames(hat) <- unlist(dimnames(agg_mat))

# obs: (observed) values for bottom-level series (B, C)
obs <- matrix(
  rnorm(length(ts_mean[-1])*N_hat, mean = ts_mean[-1]),
  N_hat, byrow = TRUE)
colnames(obs) <- colnames(agg_mat)
```

```
# h: base forecast horizon
h <- 2

# base: base forecasts matrix
base <- matrix(
  rnorm(length(ts_mean)*h, mean = ts_mean),
  h, byrow = TRUE)
colnames(base) <- unlist(dimnames(agg_mat))
# `reco` is the result of a reconciliation call:
reco <- csrml(base = base, hat = hat, obs = obs, agg_mat = agg_mat)

mdl <- extract_reconciled_ml(reco)
mdl
```

---

terml *Temporal Reconciliation with Machine Learning*

Description

This function performs machine-learning–based temporal forecast reconciliation for linearly constrained multiple time series based on the cross-temporal approach proposed by Rombouts et al. (2024). Reconciled forecasts are obtained by fitting non-linear models that map base forecasts across both temporal dimensions to high-frequency series. Fully coherent forecasts are then derived by temporal bottom-up.

Usage

```
# Reconciled forecasts
terml(base, hat, obs, agg_order, tew = "sum", features = "all",
      approach = "randomForest", params = NULL, tuning = NULL,
      sntz = FALSE, round = FALSE, fit = NULL)

# Pre-trained reconciled ML models
terml_fit(hat, obs, agg_order, tew = "sum", features = "all",
          approach = "randomForest", params = NULL, tuning = NULL)
```

Arguments

base	A $(h(k^*+m)\times 1)$ numeric vector containing the base forecasts to be reconciled, ordered from lowest to highest frequency; m is the maximum aggregation order, k^* is the sum of a chosen subset of the $p-1$ factors of m (excluding m itself) and h is the forecast horizon for the lowest frequency time series.
hat	A $(N(k^*+m)\times 1)$ numeric vector containing the base forecasts ordered from lowest to highest frequency; N is the training length for the lowest frequency time series. These forecasts are used to train the ML approach.

`obs` A ($Nm \times 1$) numeric vector containing (observed) values for the highest frequency series ($k = 1$). These values are used to train the ML approach.

`agg_order` Highest available sampling frequency per seasonal cycle (max. order of temporal aggregation, m), or a vector representing a subset of p factors of m.

`tew` A string specifying the type of temporal aggregation. Options include: `"sum"` (simple summation, *default*), `"avg"` (average), `"first"` (first value of the period), and `"last"` (last value of the period).

`features` Character string specifying which features are used for model training. Options include `"all"` (see Rombouts et al. 2025, *default*) and `"low-high"` (only the lowest- and highest-frequency base forecasts as features).

`approach` Character string specifying the machine learning method used for reconciliation. Options are:

- `"randomForest"` (*default*): Random Forest algorithm (see the **randomForest** package).
- `"xgboost"`: Extreme Gradient Boosting (see the **xgboost** package).
- `"lightgbm"`: Light Gradient Boosting Machine (see the **lightgbm** package).
- `"mlr3"`: Any regression learner available in the **mlr3** package. The learner must be specified via `params`, e.g. `params = list(.key = "regr.ranger")`.

`params` Optional list of additional parameters passed to the chosen ML approach These may include algorithm-specific hyperparameters for **randomForest**, **xgboost**, **lightgbm**, or learner options for **mlr3**. When `approach = "mlr3"`, the list must include `.key` to select the learner (e.g. `.key = "regr.ranger"`, *default*).

`tuning` Optional list specifying tuning options when using the mlr3tuning::mlr3tuning framework (e.g., terminators, search spaces). The argument format follows mlr3tuning::auto_tuner, except that the learner is set through `params`.

`sntz` Logical. If `TRUE`, enforces non-negativity on reconciled forecasts using the heuristic "set-negative-to-zero" (Di Fonzo and Girolimetto, 2023). *Default* is `FALSE`.

`round` Logical. If `TRUE`, reconciled forecasts are rounded to integer values and coherence is ensured via a bottom-up adjustment. *Default* is `FALSE`.

`fit` A pre-trained ML reconciliation model (see, extract_reconciled_ml). If supplied, training data (`hat`, `obs`) are not required.

Value

- terml returns a temporal reconciled forecast vector with the same dimensions, along with attributes containing the fitted model and reconciliation settings (see, FoReco::recoinfo and extract_reconciled_ml).

- terml_fit returns a fitted object that can be reused for reconciliation on new base forecasts.

References

Di Fonzo, T. and Girolimetto, D. (2023), Spatio-temporal reconciliation of solar forecasts, *Solar Energy*, 251, 13–29. doi:10.1016/j.solener.2023.01.003

Girolimetto, D. and Di Fonzo, T. (2023), Point and probabilistic forecast reconciliation for general linearly constrained multiple time series, *Statistical Methods & Applications*, 33, 581-607. doi:10.1007/s10260023007386.

Rombouts, J., Ternes, M., and Wilms, I. (2025). Cross-temporal forecast reconciliation at digital platforms with machine learning. *International Journal of Forecasting*, 41(1), 321-344. doi:10.1016/j.ijforecast.2024.05.008

Examples

```
# m: quarterly temporal aggregation order
m <- 4
te_set <- tetools(m)$set

# te_fh: minimum forecast horizon per temporal aggregate
te_fh <- m/te_set

# N_hat: dimension for the lowest frequency (k = m) training set
N_hat <- 16

# bts_mean: mean for the Normal draws used to simulate data
bts_mean <- 5

# hat: a training (base forecasts) feautures vector
hat <- rnorm(sum(te_fh)*N_hat, rep(te_set*bts_mean,  N_hat*te_fh))

# obs: (observed) values for the highest frequency series (k = 1)
obs <- rnorm(m*N_hat, bts_mean)

# h: base forecast horizon at the lowest-frequency series (k = m)
h <- 2

# base: base forecasts matrix
base <- rnorm(sum(te_fh)*h, rep(te_set*bts_mean,  h*te_fh))

##############################################################################
# Different ML approaches
##############################################################################
# XGBoost Reconciliation (xgboost pkg)
reco <- terml(base = base, hat = hat, obs = obs, agg_order = m,
              approach = "xgboost")

# XGBoost Reconciliation with Tweedie loss function (xgboost pkg)
reco <- terml(base = base, hat = hat, obs = obs, agg_order = m,
              approach = "xgboost",
              params =  list(
                eta = 0.3, colsample_bytree = 1, min_child_weight = 1,
                max_depth = 6, gamma = 0, subsample = 1,
                objective = "reg:tweedie", # Tweedie regression objective
                tweedie_variance_power = 1.5 # Tweedie power parameter
              ))

# LightGBM Reconciliation (lightgbm pkg)
```

```
reco <- terml(base = base, hat = hat, obs = obs, agg_order = m,
              approach = "lightgbm")

# Random Forest Reconciliation (randomForest pkg)
reco <- terml(base = base, hat = hat, obs = obs, agg_order = m,
              approach = "randomForest")

# Using the mlr3 pkg:
# With 'params = list(.key = mlr_learners)' we can specify different
# mlr_learners implemented in mlr3 such as "regr.ranger" for Random Forest,
# "regr.xgboost" for XGBoost, and others.
reco <- terml(base = base, hat = hat, obs = obs, agg_order = m,
              approach = "mlr3",
              # choose mlr3 learner (here Random Forest via ranger)
              params = list(.key = "regr.ranger"))


# With mlr3 we can also tune our parameters: e.g. explore mtry in [1,4].
# We can reduce excessive logging by calling:
# if(requireNamespace("lgr", quietly = TRUE)){
#   lgr::get_logger("mlr3")$set_threshold("warn")
#   lgr::get_logger("bbotk")$set_threshold("warn")
# }
reco <- terml(base = base, hat = hat, obs = obs, agg_order = m,
              approach = "mlr3",
              params = list(
                .key = "regr.ranger",
                # number of features tried at each split
                mtry = paradox::to_tune(paradox::p_int(1, 2))
              ),
              tuning = list(
                # stop after 10 evaluations
                terminator = mlr3tuning::trm("evals", n_evals = 10)
              ))

##############################################################################
# Usage with pre-trained models
##############################################################################
# Pre-trained machine learning models (e.g., omit the base param)
mdl <- terml_fit(hat = hat, obs = obs, agg_order = m,
                 approach = "lightgbm")

# Pre-trained machine learning models with base param
reco <- terml(base = base, hat = hat, obs = obs, agg_order = m,
              approach = "lightgbm")
mdl2 <- extract_reconciled_ml(reco)

# New base forecasts matrix
base_new <- rnorm(sum(te_fh)*h, rep(te_set*bts_mean,  h*te_fh))
reco_new <- terml(base = base_new, fit = mdl2, agg_order = m)
```

Index