

Package ‘foresightr’

October 13, 2026

Title Forecasts Chosen by What Would Have Worked

Version 0.1.1

Description Time series forecasting that replays the past before trusting a model: a rolling-origin backtest, Tashman (2000) [doi:10.1016/S0169-2070\(00\)00065-0](https://doi.org/10.1016/S0169-2070(00)00065-0), refits every candidate at each origin, the choice is made by out-of-sample error, and prediction intervals are quantiles of the errors actually observed, for each horizon and for totals of the next periods. Models include seasonal ARIMA by exact maximum likelihood with automatic orders, Hyndman and Khandakar (2008) [doi:10.18637/jss.v027.i03](https://doi.org/10.18637/jss.v027.i03); exponential smoothing in state space form, Hyndman, Koehler, Snyder and Grose (2002) [doi:10.1016/S0169-2070\(01\)00110-8](https://doi.org/10.1016/S0169-2070(01)00110-8); the Theta method, Assimakopoulos and Nikolopoulos (2000) [doi:10.1016/S0169-2070\(00\)00066-2](https://doi.org/10.1016/S0169-2070(00)00066-2); a trend with changepoints and events in the manner of Taylor and Letham (2018) [doi:10.1080/00031305.2017.1380080](https://doi.org/10.1080/00031305.2017.1380080); TBATS, De Livera, Hyndman and Snyder (2011) [doi:10.1198/jasa.2011.tm09771](https://doi.org/10.1198/jasa.2011.tm09771); seasonal-trend decompositions by LOESS for one or several seasonal periods, Bandara, Hyndman and Bergmeir (2021) [doi:10.48550/arXiv.2107.13462](https://doi.org/10.48550/arXiv.2107.13462); the method of Croston (1972) [doi:10.1057/jors.1972.50](https://doi.org/10.1057/jors.1972.50) for intermittent demand; and ensembles. The computations are done by the 'Rust' crate 'foresight', bundled with the package.

License MIT + file LICENSE

URL <https://github.com/StrategicProjects/foresightr>,
<https://strategicprojects.github.io/foresightr/>

BugReports <https://github.com/StrategicProjects/foresightr/issues>

Encoding UTF-8

Depends R (>= 4.2)

Imports ggplot2 (>= 3.5.0), stats, tibble, utils

Suggests testthat (>= 3.0.0), jsonlite, knitr, rmarkdown

Config/testthat/edition 3

SystemRequirements Cargo (Rust's package manager), rustc >= 1.81

VignetteBuilder knitr

Config/rextendr/version 0.5.0

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author André Leite [aut, cre] (ORCID: <<https://orcid.org/0000-0002-4718-9766>>),
 Marcos Wasiliew [aut] (ORCID: <<https://orcid.org/0009-0004-4694-3159>>),
 Hugo Vasconcelos [aut] (ORCID: <<https://orcid.org/0000-0001-6249-0920>>),
 Carlos Amorim [aut] (ORCID: <<https://orcid.org/0000-0001-6315-8305>>),
 Diogo Bezerra [aut] (ORCID: <<https://orcid.org/0000-0002-1216-8674>>),
 Júlia Nascimento Barreto [aut] (ORCID:
 <<https://orcid.org/0009-0004-2851-7770>>),
 The authors of the dependency Rust crates [ctb] (see inst/AUTHORS file
 for details)

Maintainer André Leite <leite@castlab.org>

Repository CRAN

Date/Publication 2026-10-13 13:30:08 UTC

Contents

autoplot.foresight_backtest	3
backtest	4
candidates_default	6
decompose_stl	6
fill_gaps	8
fit_model	8
foresight_threads	10
fourier_terms	10
kpss_statistic	11
mape	12
model_arima	13
model_auto_arima	14
model_box_cox	15
model_croston	15
model_decomposed	16
model_ensemble	17
model_ets	18
model_holt_winters	19
model_log_linear	19
model_mean	20
model_name	21
model_prophet	22
model_tbats	23
model_theta	24
theme_foresight	25
total_forecast	25
with_name	26

autoplot.foresight_backtest

Charts of a backtest or a decomposition

Description

autoplot() draws, with ggplot2:

Usage

```
## S3 method for class 'foresight_backtest'
autoplot(
  object,
  type = c("forecast", "accuracy", "ranking"),
  candidate = object$best,
  history = 48,
  top = 4,
  ...
)

## S3 method for class 'foresight_backtest'
plot(
  x,
  type = c("forecast", "accuracy", "ranking"),
  candidate = x$best,
  history = 48,
  top = 4,
  ...
)

## S3 method for class 'foresight_decomposition'
autoplot(object, ...)

## S3 method for class 'foresight_decomposition'
plot(x, ...)
```

Arguments

object, x	A result of backtest() or of decompose_stl() / decompose_mstl() .
type	What to draw.
candidate	A candidate name, for type = "forecast"; by default the chosen one.
history	How many of the last observations to show.
top	How many of the best candidates, for type = "accuracy": at most 4, so that they can be told apart.
...	Not used.

Details

- `type = "forecast"`: the last observations and the forecast of a candidate with its empirical intervals;
- `type = "accuracy"`: the error of the best candidates by horizon;
- `type = "ranking"`: every candidate by its score, the chosen one highlighted;

and, for a decomposition, the series, the trend, each seasonal pattern and the remainder. `plot()` draws the same chart. Both return a ggplot object, which can be themed and annotated further.

Value

A ggplot object.

Examples

```
bt <- backtest(AirPassengers, origins = 24)
autoplot(bt)
autoplot(bt, "accuracy")
autoplot(decompose_stl(log(AirPassengers), seasonal_window = 13))
```

backtest

Choose a model by what would have worked

Description

Replays the past with each candidate: at each of the last origins periods the model is fitted on the data before it and forecasts up to horizon periods ahead. The candidates are ranked by the average error (`metric`) over the horizons, the simple average of the best combine is added as one more candidate, and every candidate forecasts from the whole series with intervals taken from the errors it made: quantiles of the relative error at each horizon, and of the relative error of totals for the forecast of the next `k` periods together.

Usage

```
backtest(
  y,
  candidates = candidates_default(),
  period = NULL,
  origins = 36,
  horizon = 12,
  min_train = 48,
  window = NULL,
  combine = 2,
  levels = c(0.8, 0.95),
  metric = c("mape", "mae", "rmse", "mase"),
  parallel = TRUE
)
```

Arguments

<code>y</code>	A <code>ts</code> or a numeric vector.
<code>candidates</code>	A list of models with distinct names (see with_name()); by default candidates_default() . A candidate that cannot forecast at every origin and from the whole series is left out, with a warning.
<code>period</code>	The seasonal period when <code>y</code> is a plain vector.
<code>origins</code>	Forecast origins, the last periods of the series.
<code>horizon</code>	Longest horizon forecast and evaluated.
<code>min_train</code>	Observations to train at the first origin; shorter series get fewer origins.
<code>window</code>	Train on the last window observations only, at every origin and for the final forecast (default: everything before the origin).
<code>combine</code>	Also evaluate the average of the best combine models (fewer than 2, or more than the candidates left, disables it).
<code>levels</code>	Coverage of the intervals, each above 0 and below 1.
<code>metric</code>	"mape", "mae", "rmse" or "mase".
<code>parallel</code>	Fit the origins on several threads: all cores, or as many as foresight_threads() allows. The result is the same either way.

Value

An object of class `foresight_backtest`, a list whose tables are tibbles:

- `ranking`: one row per candidate that went through the backtest, with its score, whether it was chosen, the models involved and a description; dropped names those left out;
- `best`: the name of the chosen candidate;
- `forecast`: the forecast of the chosen candidate with its intervals, and the time of each period for a `ts` (a date for monthly and quarterly data);
- `candidates`: for each candidate, its accuracy by horizon (pairs evaluated, MAPE, bias, MAE, RMSE, MASE), the relative error bands, the forecast, the cumulative forecast of totals, the trajectories of the backtest (origins by horizons) and the fitted params; an interval is NA at a horizon where every forecast of the backtest was zero;
- `origins`, `first_origin` (position of the first period forecast), `horizon`, `metric` and `levels`.

Examples

```
bt <- backtest(AirPassengers, origins = 24)
bt
bt$forecast
total_forecast(bt, 6)
```

candidates_default	<i>Ready sets of candidates</i>
--------------------	---------------------------------

Description

`candidates_default()` has the benchmarks and every model that needs no external data and fits in a moment: naive, drift, seasonal naive (plain and with growth), Theta, Holt-Winters, log-linear regression, the airline ARIMA and Prophet (each on the original and the log scale): 11 models. `candidates_thorough()` adds two ensembles of those (inverse error and stacked), ETS chosen automatically (alone, after STL, and after STL on the log scale) and ARIMA with automatic orders (original and log): 18 models, seconds rather than milliseconds in a backtest.

Usage

```
candidates_default()

candidates_thorough()
```

Value

A list of models.

Examples

```
vapply(candidates_default(), model_name, "")
```

decompose_stl	<i>Decompose a series by STL or MSTL</i>
---------------	--

Description

`decompose_stl()` splits a series into trend, seasonal pattern and remainder by LOESS (Cleveland, Cleveland, McRae & Terpenning, 1990); its defaults are those of `stats::stl()` and, without robust, so are its numbers (with `robust = TRUE` the two agree up to the eleventh robustness round and drift apart in the third digit after that). `decompose_mstl()` applies STL in turn to each of several seasonal periods (Bandara, Hyndman & Bergmeir, 2021).

Usage

```
decompose_stl(
  y,
  period = NULL,
  seasonal_window = NULL,
  trend_window = NULL,
  low_pass_window = NULL,
  degrees = NULL,
```

```

    robust = FALSE,
    inner = NULL,
    outer = NULL
  )

  decompose_mstl(
    y,
    periods = NULL,
    windows = NULL,
    iterations = NULL,
    robust = FALSE
  )

```

Arguments

<code>y</code>	A ts or a numeric vector, longer than two full cycles.
<code>period</code>	The seasonal period, when <code>y</code> is a plain vector.
<code>seasonal_window</code>	The LOESS window over the cycles, an odd number, usually 7 or more (an even number is taken as the next odd one): the smaller, the faster the pattern may change. NULL keeps the same pattern in every cycle (<code>s.window = "periodic"</code>).
<code>trend_window, low_pass_window</code>	LOESS windows of the trend and of the low-pass filter.
<code>degrees</code>	LOESS degrees (0 or 1) of the seasonal, trend and low-pass smoothers; default <code>c(0, 1, 1)</code> .
<code>robust</code>	Down-weight outliers.
<code>inner, outer</code>	Passes of the inner loop and robustness rounds.
<code>periods</code>	Seasonal periods.
<code>windows</code>	Seasonal windows, one per period in increasing order of period (default 11, 15, 19, ...).
<code>iterations</code>	Rounds over the periods (default 2).

Value

An object of class `foresight_decomposition`, a list with `trend`, `seasonal` (a matrix, one column per period), `remainder`, `seasonally_adjusted`, `periods`, `trend_strength` and `seasonal_strength` (from 0 to 1; Wang, Smith & Hyndman, 2006).

Examples

```

d <- decompose_stl(log(AirPassengers), seasonal_window = 13)
d$seasonal_strength
plot(d)

```

fill_gaps

Gaps and outliers

Description

fill_gaps() fills missing values, following the seasonal pattern when there is one. find_outliers() finds the observations far from what the trend and the season suggest, and what they should rather be. clean_series() does both.

Usage

```
fill_gaps(y, period = NULL)
```

```
find_outliers(y, period = NULL)
```

```
clean_series(y, period = NULL)
```

Arguments

y A ts or a numeric vector, possibly with NA.
period The seasonal period, when y is a plain vector.

Value

fill_gaps() and clean_series(): the series, a ts when y is one. find_outliers(): a tibble with the index (from 1), the value and its replacement, plus the time for a ts (a date for monthly and quarterly data).

Examples

```
y <- log(AirPassengers)
y[c(30, 100)] <- y[c(30, 100)] + c(0.8, -0.7)
y[60] <- NA
find_outliers(y)
clean_series(y)[c(30, 60, 100)]
```

fit_model

Fit a model and forecast

Description

fit_model() estimates a model on a series and keeps what was estimated; predict() forecasts from it. forecast_model() does both in one go.

Usage

```
fit_model(model, y, period = NULL)

## S3 method for class 'foresight_fit'
predict(object, h = 12, ...)

forecast_model(model, y, h = 12, period = NULL)
```

Arguments

model	A model, e.g. <code>model_theta()</code> .
y	A ts or a numeric vector.
period	The seasonal period when y is a plain vector (1 for none); a ts brings its own.
object	A fit from <code>fit_model()</code> .
h	Periods to forecast.
...	Not used.

Details

The estimate lives in memory: a fit restored with `readRDS()` cannot forecast and has to be fitted again.

Value

`fit_model()`: an object of class `foresight_fit`, a list with the model name and description, the estimated params (named numeric), `log_likelihood`, `aic`, `aicc`, `bic` and `residuals` when the model has them (NA otherwise; the residuals of ARIMA are those of the differenced series, so there are fewer of them than observations), and details: orders and coefficients for ARIMA, the code and smoothing for ETS, changepoints (positions from 1) and event effects for Prophet, the structure for TBATS and its `minus_two_log_likelihood` (up to a constant; the lower the better, as its AIC). `forecast_model()`: the forecasts, a ts when y is one. Both fail when a forecast is not finite, as when regressors do not reach the horizon.

Examples

```
fit <- fit_model(model_log(model_airline()), AirPassengers)
predict(fit, h = 12)
forecast_model(model_theta(), AirPassengers, h = 3)
```

foresight_threads	<i>Threads used by backtests and ensembles</i>
-------------------	--

Description

Backtests and ensembles fit on several threads: by default, all the cores of the machine. `foresight_threads(n)` limits them to `n`, for the whole session; `foresight_threads(0)` removes the limit. The results do not depend on the number of threads.

Usage

```
foresight_threads(n = NULL)
```

Arguments

<code>n</code>	The most threads to use, or 0 for all cores. NULL changes nothing.
----------------	--

Details

When the package is loaded the limit comes from the option `foresightr.threads` or, failing that, from the environment variable `OMP_THREAD_LIMIT`; under R CMD check it is 2.

Value

The number of threads in use after the call; invisibly when it was set.

Examples

```
foresight_threads()
```

fourier_terms	<i>External variables for a regression with ARIMA errors</i>
---------------	--

Description

`fourier_terms()` gives the sine and cosine pairs of a seasonal period up to order harmonics (those beyond half the period, which would repeat the earlier ones, are left out); `seasonal_dummies()` one dummy per season but the first. Both have rows `rows`, which must cover the series and the horizon to be forecast. Combine them, or add your own columns, with `cbind()`.

Usage

```
fourier_terms(period, order, rows)
```

```
seasonal_dummies(period, rows)
```

Arguments

period	The seasonal period (need not be a whole number for Fourier terms).
order	Harmonics.
rows	Rows: the length of the series plus the horizon.

Value

A tibble.

Examples

```
x <- fourier_terms(12, 3, length(AirPassengers) + 12)
fit <- fit_model(model_arima(c(1, 1, 1), constant = TRUE, regressors = x), log(AirPassengers))
predict(fit, 12)
```

kpss_statistic	<i>Tests and transformations</i>
----------------	----------------------------------

Description

- `kpss_statistic()`: the KPSS statistic for level stationarity (Kwiatkowski et al., 1992); above 0.463 a difference is called for at 5%.
- `n_differences()`: differences needed by repeated KPSS tests.
- `n_seasonal_differences()`: seasonal differences needed, by the strength of seasonality.
- `seasonal_strength()`: from 0 to 1 (Wang, Smith & Hyndman, 2006).
- `autocorrelations()`: at lags 1 to `max_lag`.
- `box_cox()`, `inv_box_cox()`: the transformation and its inverse; `guerrero_lambda()` chooses `lambda` by Guerrero's method (1993).

Usage

```
kpss_statistic(y)

n_differences(y, max = 2)

n_seasonal_differences(y, period = NULL)

seasonal_strength(y, period = NULL)

autocorrelations(y, max_lag)

box_cox(x, lambda)

inv_box_cox(x, lambda)

guerrero_lambda(y, period = NULL)
```

Arguments

y, x	A ts or a numeric vector.
max	Most differences.
period	The seasonal period, when y is a plain vector.
max_lag	Largest lag.
lambda	The Box-Cox parameter; 0 is the log.

Value

A number, or the transformed values.

Examples

```
kpss_statistic(log(AirPassengers))
n_differences(log(AirPassengers))
guerrero_lambda(AirPassengers)
```

mape	<i>Accuracy of forecasts</i>
------	------------------------------

Description

mape() is the mean absolute percentage error and pct_bias() the mean of (forecast - actual) / actual, both in percent; mae() and rmse() are the mean absolute and root mean squared errors; mase() scales the mean absolute error by the in-sample error of the seasonal naive forecast on train (Hyndman & Koehler, 2006).

Usage

```
mape(actual, forecast)

pct_bias(actual, forecast)

mae(actual, forecast)

rmse(actual, forecast)

mase(actual, forecast, train, period = 1)
```

Arguments

actual, forecast	Numeric vectors of the same length.
train	The history the forecasts were made from, for mase().
period	Its seasonal period.

Value

A number (NA when it cannot be computed).

Examples

```
mape(c(100, 200), c(110, 180))
```

model_arima	<i>Seasonal ARIMA</i>
-------------	-----------------------

Description

ARIMA(p, d, q)(P, D, Q) estimated by exact Gaussian maximum likelihood (the innovations algorithm), from three starting points. `model_airline()` is ARIMA(0,1,1)(0,1,1), a good default for seasonal series, usually on the log scale (`model_log()`).

Usage

```
model_arima(
  order = c(0, 1, 1),
  seasonal = c(0, 0, 0),
  constant = NULL,
  regressors = NULL
)

model_airline()
```

Arguments

order	(p, d, q).
seasonal	(P, D, Q), ignored for series without seasonality.
constant	Estimate a mean (series not differenced) or a drift (one difference). By default only the mean is estimated.
regressors	External variables, a data frame, matrix or named list of numeric columns with one row per period from the first observation on; to forecast, the rows must also cover the horizon. The model becomes a regression with ARIMA errors. See fourier_terms() .

Value

A model specification, to use with [fit_model\(\)](#), [forecast_model\(\)](#) or [backtest\(\)](#).

See Also

Other models: [model_auto_arima\(\)](#), [model_box_cox\(\)](#), [model_croston\(\)](#), [model_decomposed\(\)](#), [model_ensemble\(\)](#), [model_ets\(\)](#), [model_holt_winters\(\)](#), [model_log_linear\(\)](#), [model_mean\(\)](#), [model_prophet\(\)](#), [model_tbats\(\)](#), [model_theta\(\)](#)

Examples

```
fit <- fit_model(model_airline(), log(AirPassengers))
fit$details$seasonal_ma
```

model_auto_arima	<i>ARIMA with automatic orders</i>
------------------	------------------------------------

Description

Differences chosen by the KPSS test and by the strength of seasonality, orders by a stepwise search on the information criterion (Hyndman & Khandakar, 2008). In a backtest the choice is made again at every origin, so the whole procedure is judged, not one lucky specification.

Usage

```
model_auto_arima(
  criterion = c("aicc", "aic", "bic"),
  d = NULL,
  seasonal_d = NULL,
  max_order = c(5, 5, 2, 2),
  regressors = NULL
)
```

Arguments

criterion	"aicc", "aic" or "bic".
d, seasonal_d	Fix the number of differences instead of testing.
max_order	Largest p, q, P and Q.
regressors	External variables, a data frame, matrix or named list of numeric columns with one row per period from the first observation on; to forecast, the rows must also cover the horizon. The model becomes a regression with ARIMA errors. See fourier_terms() .

Value

A model specification, to use with [fit_model\(\)](#), [forecast_model\(\)](#) or [backtest\(\)](#).

See Also

Other models: [model_arima\(\)](#), [model_box_cox\(\)](#), [model_croston\(\)](#), [model_decomposed\(\)](#), [model_ensemble\(\)](#), [model_ets\(\)](#), [model_holt_winters\(\)](#), [model_log_linear\(\)](#), [model_mean\(\)](#), [model_prophet\(\)](#), [model_tbats\(\)](#), [model_theta\(\)](#)

Examples

```
fit <- fit_model(model_auto_arima(), log(AirPassengers))
fit$details$order
fit$details$seasonal_order
```

model_box_cox	<i>A model on the log or Box-Cox scale</i>
---------------	--

Description

Runs model on the transformed series and brings the forecasts back to the original scale. The series must be positive.

Usage

```
model_box_cox(model, lambda = 0)

model_log(model)
```

Arguments

model	The model.
lambda	The Box-Cox parameter (0 is the log), or "guerrero" to choose it at each fit by Guerrero's method.

Value

A model specification, to use with [fit_model\(\)](#), [forecast_model\(\)](#) or [backtest\(\)](#).

See Also

Other models: [model_arima\(\)](#), [model_auto_arima\(\)](#), [model_croston\(\)](#), [model_decomposed\(\)](#), [model_ensemble\(\)](#), [model_ets\(\)](#), [model_holt_winters\(\)](#), [model_log_linear\(\)](#), [model_mean\(\)](#), [model_prophet\(\)](#), [model_tbats\(\)](#), [model_theta\(\)](#)

Examples

```
forecast_model(model_log(model_airline()), AirPassengers, h = 3)
```

model_croston	<i>Intermittent demand</i>
---------------	----------------------------

Description

Croston's method (1972) and its variants for series where most periods have no demand: "sba" (Syntetos & Boylan, 2005) removes Croston's upward bias; "tsb" (Teunter, Syntetos & Babai, 2011) smooths the probability of demand, so the rate falls while nothing is sold. The forecast is the same for every horizon.

Usage

```
model_croston(
  variant = c("croston", "sba", "tsb"),
  alpha = NULL,
  beta = NULL,
  optimised = FALSE
)
```

Arguments

variant	"croston", "sba" or "tsb".
alpha	Smoothing of the demand sizes and intervals (default 0.1).
beta	Smoothing of the probability of demand, for TSB (default: alpha).
optimised	Choose alpha by the squared error of the rate.

Value

A model specification, to use with `fit_model()`, `forecast_model()` or `backtest()`.

See Also

Other models: `model_arima()`, `model_auto_arima()`, `model_box_cox()`, `model_decomposed()`, `model_ensemble()`, `model_ets()`, `model_holt_winters()`, `model_log_linear()`, `model_mean()`, `model_prophet()`, `model_tbats()`, `model_theta()`

Examples

```
demand <- c(0, 0, 3, 0, 0, 0, 2, 0, 0, 4, 0, 0, 0, 0, 3, 0, 2, 0, 0, 0)
forecast_model(model_croston("sba"), demand, h = 3)
```

model_decomposed	<i>Forecast the seasonally adjusted series</i>
------------------	--

Description

Decomposes the series by STL (or MSTL, for several periods), forecasts the seasonally adjusted series with model and adds back the seasonal pattern of the last cycle.

Usage

```
model_decomposed(model, periods = NULL, robust = FALSE)
```

Arguments

model	The model for the seasonally adjusted series.
periods	Seasonal periods to take out; by default the period of the series.
robust	Down-weight outliers in the decomposition.

Value

A model specification, to use with `fit_model()`, `forecast_model()` or `backtest()`.

See Also

Other models: `model_arima()`, `model_auto_arima()`, `model_box_cox()`, `model_croston()`, `model_ensemble()`, `model_ets()`, `model_holt_winters()`, `model_log_linear()`, `model_mean()`, `model_prophet()`, `model_tbats()`, `model_theta()`

Examples

```
forecast_model(model_log(model_decomposed(model_drift()))), AirPassengers, h = 3)
```

model_ensemble	<i>Several models combined</i>
----------------	--------------------------------

Description

The weights are learnt from the series itself: the last origins periods are forecast by every member from the data before them, and the errors decide how much each member counts. Then the members are fitted on the whole series and their forecasts combined. As a candidate in a `backtest()`, an ensemble learns its weights again at every origin.

Usage

```
model_ensemble(
  members,
  weighting = c("inverse_error", "equal", "median", "stacked"),
  origins = NULL,
  horizon = NULL,
  top = NULL
)
```

Arguments

members	A list of models.
weighting	"inverse_error" (in inverse proportion to the mean squared error), "equal", "median" or "stacked" (the weights, none negative and adding up to one, with the smallest squared error).
origins	Periods forecast to learn the weights (default 12).
horizon	How far ahead those forecasts go (default: one seasonal cycle, at most 12).
top	Keep only the members with the smallest error.

Value

A model specification, to use with `fit_model()`, `forecast_model()` or `backtest()`.

See Also

Other models: `model_arima()`, `model_auto_arima()`, `model_box_cox()`, `model_croston()`, `model_decomposed()`, `model_ets()`, `model_holt_winters()`, `model_log_linear()`, `model_mean()`, `model_prophet()`, `model_tbats()`, `model_theta()`

Examples

```
fit <- fit_model(model_ensemble(candidates_default(), top = 3), AirPassengers)
fit$params
```

model_ets

Exponential smoothing

Description

A member of the exponential smoothing family in state space form (Hyndman et al., 2008), by its code: error (A, M), trend (N, A, Ad) and season (N, A, M), such as "MAM" or "AAdN". `model_auto_ets()` chooses the member with the best criterion; multiplicative parts only for positive series.

Usage

```
model_ets(code)

model_auto_ets(criterion = c("aicc", "aic", "bic"))
```

Arguments

code	The model, e.g. "MAM".
criterion	"aicc", "aic" or "bic".

Value

A model specification, to use with `fit_model()`, `forecast_model()` or `backtest()`.

See Also

Other models: `model_arima()`, `model_auto_arima()`, `model_box_cox()`, `model_croston()`, `model_decomposed()`, `model_ensemble()`, `model_holt_winters()`, `model_log_linear()`, `model_mean()`, `model_prophet()`, `model_tbats()`, `model_theta()`

Examples

```
fit <- fit_model(model_auto_ets(), AirPassengers)
fit$details$code
```

model_holt_winters	<i>Holt-Winters</i>
--------------------	---------------------

Description

Holt-Winters with multiplicative seasonality, the smoothing parameters chosen by the squared error of the one-step forecasts. The series must be positive and cover three cycles.

Usage

```
model_holt_winters()
```

Value

A model specification, to use with `fit_model()`, `forecast_model()` or `backtest()`.

See Also

Other models: `model_arima()`, `model_auto_arima()`, `model_box_cox()`, `model_croston()`, `model_decomposed()`, `model_ensemble()`, `model_ets()`, `model_log_linear()`, `model_mean()`, `model_prophet()`, `model_tbats()`, `model_theta()`

Examples

```
fit_model(model_holt_winters(), AirPassengers)$params
```

model_log_linear	<i>Log-linear regression</i>
------------------	------------------------------

Description

Regression of the log of the series on a linear trend and seasonal dummies, brought back with Duan's smearing correction.

Usage

```
model_log_linear(window = NULL, deflator = NULL)
```

Arguments

window	Fit on the last window observations only.
deflator	A price index to deflate by before fitting, one value per period from the first observation on. The future of the index is not read: the forecasts are inflated back at its growth over the last cycle.

Value

A model specification, to use with `fit_model()`, `forecast_model()` or `backtest()`.

See Also

Other models: `model_arima()`, `model_auto_arima()`, `model_box_cox()`, `model_croston()`, `model_decomposed()`, `model_ensemble()`, `model_ets()`, `model_holt_winters()`, `model_mean()`, `model_prophet()`, `model_tbats()`, `model_theta()`

Examples

```
forecast_model(model_log_linear(window = 60), AirPassengers, h = 3)
```

model_mean	<i>Benchmark models</i>
------------	-------------------------

Description

The yardsticks every other model must beat.

Usage

```
model_mean()
```

```
model_naive()
```

```
model_drift()
```

```
model_seasonal_naive(growth = FALSE)
```

Arguments

`growth` Scale by the growth of the last cycle.

Details

- `model_mean()`: the mean of the history.
- `model_naive()`: the last value (random walk).
- `model_drift()`: the last value plus the average change (random walk with drift).
- `model_seasonal_naive()`: the same season of the last cycle; with `growth = TRUE`, scaled by the growth of the last cycle over the one before.

Value

A model specification, to use with `fit_model()`, `forecast_model()` or `backtest()`.

See Also

Other models: `model_arima()`, `model_auto_arima()`, `model_box_cox()`, `model_croston()`, `model_decomposed()`, `model_ensemble()`, `model_ets()`, `model_holt_winters()`, `model_log_linear()`, `model_prophet()`, `model_tbats()`, `model_theta()`

Examples

```
forecast_model(model_seasonal_naive(), AirPassengers, h = 3)
```

model_name	<i>Name and description of a model</i>
------------	--

Description

Name and description of a model

Usage

```
model_name(model)
```

```
model_description(model)
```

Arguments

model	A model.
-------	----------

Value

`model_name()`: the short name used in reports, such as "log_arima_011_011"; `model_description()`: one line about the method.

Examples

```
model_name(model_log(model_airline()))
```

model_prophet	<i>Prophet-style trend with changepoints and events</i>
---------------	---

Description

The model of Taylor & Letham (2018) fitted without Stan: a piecewise linear trend whose changes of slope are shrunk by a Laplace prior (those that do not matter come out as exactly zero), Fourier seasonality, and optional events and lasting steps.

Usage

```
model_prophet(
  changepoints = NULL,
  changepoint_range = NULL,
  changepoint_prior_scale = NULL,
  seasonality_prior_scale = NULL,
  fourier_order = NULL,
  event_prior_scale = NULL,
  events = NULL,
  steps = NULL
)
```

Arguments

changepoints	Potential changepoints (default 25).
changepoint_range	Share of the history where they may fall (default 0.8).
changepoint_prior_scale, seasonality_prior_scale, event_prior_scale	Scales of the priors (defaults 0.05, 10 and 10).
fourier_order	Harmonics of the seasonal pattern (default: 10 for yearly patterns, at most half the period).
events	Named list: for each event, the positions where it happens, counted from 1 at the first observation, future ones included. Seasonal terms are fitted from two full cycles on.
steps	Named list: for each lasting change of level, the position from which it applies.

Value

A model specification, to use with [fit_model\(\)](#), [forecast_model\(\)](#) or [backtest\(\)](#).

See Also

Other models: [model_arima\(\)](#), [model_auto_arima\(\)](#), [model_box_cox\(\)](#), [model_croston\(\)](#), [model_decomposed\(\)](#), [model_ensemble\(\)](#), [model_ets\(\)](#), [model_holt_winters\(\)](#), [model_log_linear\(\)](#), [model_mean\(\)](#), [model_tbats\(\)](#), [model_theta\(\)](#)

Examples

```
# ten years of monthly sales with a campaign every other November (+20)
# and a lasting change of level from the 81st month on (-15)
t <- 1:120
sales <- 200 + 0.8 * t + 5 * sin(2 * pi * t / 12)
campaigns <- c(11, 35, 59, 83, 107)
sales[campaigns] <- sales[campaigns] + 20
sales[t >= 81] <- sales[t >= 81] - 15
# the campaign planned for month 131 enters the forecast
model <- model_prophet(changepoints = 0,
                       events = list(campaign = c(campaigns, 131)),
                       steps = list(new_law = 81))
fit <- fit_model(model, ts(sales, frequency = 12))
round(fit$details$effects, 1)
```

model_tbats	<i>TBATS</i>
-------------	--------------

Description

Trigonometric seasonality, Box-Cox transformation, ARMA errors, trend and seasonal components (De Livera, Hyndman & Snyder, 2011). The seasonal periods need not be whole numbers and there may be several. Whatever is left as NULL is chosen by AIC. It is the slowest model of the package: seconds for ten years of monthly data.

Usage

```
model_tbats(
  periods = NULL,
  harmonics = NULL,
  box_cox = NULL,
  trend = NULL,
  damped = NULL,
  arma_errors = NULL,
  arma_orders = NULL
)
```

Arguments

periods	Seasonal periods; by default the period of the series.
harmonics	Harmonics of each period, in increasing order of period.
box_cox, trend, damped, arma_errors	TRUE, FALSE, or NULL to choose.
arma_orders	(p, q) of the ARMA errors, instead of choosing.

Value

A model specification, to use with `fit_model()`, `forecast_model()` or `backtest()`.

See Also

Other models: `model_arma()`, `model_auto_arma()`, `model_box_cox()`, `model_croston()`, `model_decomposed()`, `model_ensemble()`, `model_ets()`, `model_holt_winters()`, `model_log_linear()`, `model_mean()`, `model_prophet()`, `model_theta()`

Examples

```
# a structure given in full fits at once; whatever is left out is chosen
model <- model_tbats(harmonics = 3, box_cox = FALSE, trend = TRUE, damped = FALSE,
                    arma_errors = FALSE)
fit <- fit_model(model, log(AirPassengers))
exp(predict(fit, 3))
```

model_theta

Theta method

Description

The Theta method (Assimakopoulos & Nikolopoulos, 2000), equivalent to simple exponential smoothing with drift, applied to the series seasonally adjusted by classical multiplicative decomposition when it is seasonal.

Usage

```
model_theta()
```

Value

A model specification, to use with `fit_model()`, `forecast_model()` or `backtest()`.

See Also

Other models: `model_arma()`, `model_auto_arma()`, `model_box_cox()`, `model_croston()`, `model_decomposed()`, `model_ensemble()`, `model_ets()`, `model_holt_winters()`, `model_log_linear()`, `model_mean()`, `model_prophet()`, `model_tbats()`

Examples

```
forecast_model(model_theta(), AirPassengers, h = 3)
```

theme_foresight	<i>The theme of the package's charts</i>
-----------------	--

Description

A quiet theme on top of `ggplot2::theme_minimal()`: horizontal grid lines only, muted axes, the title aligned with the plot.

Usage

```
theme_foresight(base_size = 11, base_family = "")
```

Arguments

base_size	Base font size.
base_family	Base font family.

Value

A ggplot2 theme.

Examples

```
library(ggplot2)
ggplot(data.frame(x = 1:10, y = cumsum(rnorm(10)))), aes(x, y)) +
  geom_line() +
  theme_foresight()
```

total_forecast	<i>Forecast of a total</i>
----------------	----------------------------

Description

The forecast of the sum of the next *k* periods, with intervals measured on totals in the backtest. Adding up the limits of each period would overstate the uncertainty of a total.

Usage

```
total_forecast(backtest, k, candidate = backtest$best)
```

Arguments

backtest	A result of <code>backtest()</code> .
k	Periods added up.
candidate	A candidate name; by default the chosen one.

Value

A one-row tibble: periods, mean and the bounds of each interval.

Examples

```
bt <- backtest(AirPassengers, list(model_theta(), model_seasonal_naive()), origins = 24)
total_forecast(bt, 6)
colSums(bt$forecast[1:6, c("mean", "lower_80", "upper_80")]) # wider, and wrong
```

with_name	<i>Rename a model</i>
-----------	-----------------------

Description

Sets the name (and description) under which a model appears in a backtest or an ensemble.

Usage

```
with_name(model, name, description = NULL)
```

Arguments

model	A model.
name	The new name.
description	The new description; by default the model's own.

Value

The model, renamed.

Examples

```
model <- with_name(model_seasonal_naive(), "same_month", "The same month of last year")
model
```

Index

* models

- model_arima, 13
- model_auto_arima, 14
- model_box_cox, 15
- model_croston, 15
- model_decomposed, 16
- model_ensemble, 17
- model_ets, 18
- model_holt_winters, 19
- model_log_linear, 19
- model_mean, 20
- model_prophet, 22
- model_tbats, 23
- model_theta, 24

autocorrelations (kpss_statistic), 11

autoplot.foresight_backtest, 3

autoplot.foresight_decomposition
(autoplot.foresight_backtest),
3

backtest, 4

backtest(), 3, 13–20, 22–25

box_cox (kpss_statistic), 11

candidates_default, 6

candidates_default(), 5

candidates_thorough
(candidates_default), 6

cbind(), 10

clean_series (fill_gaps), 8

decompose_mstl (decompose_stl), 6

decompose_mstl(), 3

decompose_stl, 6

decompose_stl(), 3

fill_gaps, 8

find_outliers (fill_gaps), 8

fit_model, 8

fit_model(), 13–20, 22–24

forecast_model (fit_model), 8

forecast_model(), 13–20, 22–24

foresight_threads, 10

foresight_threads(), 5

fourier_terms, 10

fourier_terms(), 13, 14

ggplot2::theme_minimal(), 25

guerrero_lambda (kpss_statistic), 11

inv_box_cox (kpss_statistic), 11

kpss_statistic, 11

mae (mape), 12

mape, 12

mase (mape), 12

model_airline (model_arima), 13

model_arima, 13

model_arima(), 14–22, 24

model_auto_arima, 14

model_auto_arima(), 13, 15–22, 24

model_auto_ets (model_ets), 18

model_box_cox, 15

model_box_cox(), 13, 14, 16–22, 24

model_croston, 15

model_croston(), 13–15, 17–22, 24

model_decomposed, 16

model_decomposed(), 13–16, 18–22, 24

model_description (model_name), 21

model_drift (model_mean), 20

model_ensemble, 17

model_ensemble(), 13–22, 24

model_ets, 18

model_ets(), 13–22, 24

model_holt_winters, 19

model_holt_winters(), 13–18, 20–22, 24

model_log (model_box_cox), 15

model_log(), 13

model_log_linear, 19

`model_log_linear()`, [13–19](#), [21](#), [22](#), [24](#)
`model_mean`, [20](#)
`model_mean()`, [13–20](#), [22](#), [24](#)
`model_naive(model_mean)`, [20](#)
`model_name`, [21](#)
`model_prophet`, [22](#)
`model_prophet()`, [13–21](#), [24](#)
`model_seasonal_naive(model_mean)`, [20](#)
`model_tbats`, [23](#)
`model_tbats()`, [13–22](#), [24](#)
`model_theta`, [24](#)
`model_theta()`, [9](#), [13–22](#), [24](#)

`n_differences(kpss_statistic)`, [11](#)
`n_seasonal_differences`
 (`kpss_statistic`), [11](#)

`pct_bias(mape)`, [12](#)
`plot.foresight_backtest`
 (`autoplot.foresight_backtest`),
 [3](#)
`plot.foresight_decomposition`
 (`autoplot.foresight_backtest`),
 [3](#)
`predict()`, [8](#)
`predict.foresight_fit(fit_model)`, [8](#)

`rmse(mape)`, [12](#)

`seasonal_dummies(fourier_terms)`, [10](#)
`seasonal_strength(kpss_statistic)`, [11](#)
`stats::stl()`, [6](#)

`theme_foresight`, [25](#)
`total_forecast`, [25](#)

`with_name`, [26](#)
`with_name()`, [5](#)