Package ‘fable.prophet’

October 11, 2026

Version 1.0.0

Title Prophet Modelling Interface for 'fable'

Description
Allows prophet models from the 'prophet' package to be used in a tidy workflow with the modelling interface of 'fabletools'. This extends 'prophet' to provide enhanced model specification and management, performance evaluation methods, and model combination tools.

Depends R (>= 4.1.0), fabletools (>= 0.5.0)

Imports rlang, tsibble, lubridate, prophet, dplyr, distributional

Suggests tsibbledata, future, testthat, ggplot2, covr, knitr, rmarkdown

ByteCompile true

License GPL-3

Encoding UTF-8

Language en-GB

URL https://pkg.mitchelloharawild.com/fable.prophet/, https://github.com/mitchelloharawild/fable.prophet

BugReports https://github.com/mitchelloharawild/fable.prophet/issues

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Mitchell O'Hara-Wild [aut, cre], Sean Taylor [ctb] (Prophet library, https://facebook.github.io/prophet/), Ben Letham [ctb] (Prophet library, https://facebook.github.io/prophet/)

Maintainer Mitchell O'Hara-Wild <mail@mitchelloharawild.com>

Repository CRAN

Date/Publication 2026-10-11 04:40:02 UTC

Contents

fable.prophet-package 2
components.fbl_prophet 3
fitted.fbl_prophet 4
forecast.fbl_prophet 4
generate.fbl_prophet 5
glance.fbl_prophet 6
prophet 7
refit.fbl_prophet 10
residuals.fbl_prophet 11
tidy.fbl_prophet 12

Index **14**

`fable.prophet-package` *fable.prophet: Prophet Modelling Interface for 'fable'*

Description

Allows prophet models from the 'prophet' package to be used in a tidy workflow with the modelling interface of 'fabletools'. This extends 'prophet' to provide enhanced model specification and management, performance evaluation methods, and model combination tools.

Author(s)

Maintainer: Mitchell O'Hara-Wild <mail@mitchelloharawild.com>

Authors:

- Mitchell O'Hara-Wild <mail@mitchelloharawild.com>

Other contributors:

- Sean Taylor (Prophet library, https://facebook.github.io/prophet/) [contributor]
- Ben Letham (Prophet library, https://facebook.github.io/prophet/) [contributor]

See Also

Useful links:

- https://pkg.mitchelloharawild.com/fable.prophet/
- https://github.com/mitchelloharawild/fable.prophet
- Report bugs at https://github.com/mitchelloharawild/fable.prophet/issues

`components.fbl_prophet` *Extract meaningful components*

Description

A prophet model consists of terms which are additively or multiplicatively included in the model. Multiplicative terms are scaled proportionally to the estimated trend, while additive terms are not.

Usage

```
## S3 method for class 'fbl_prophet'
components(object, ...)
```

Arguments

`object`	An estimated model.
`...`	Unused.

Details

Holiday effects (`holidays`) and exogenous regressor effects (`extra_regressors_additive` and `extra_regressors_multiplicative`) are included when they are part of the model. Like the seasonal terms, these are contained within the model's `additive_terms` or `multiplicative_terms`.

Extracting a prophet model's components using this function allows you to visualise the components in a similar way to `prophet::prophet_plot_components()`.

Value

A `fabletools::dable()` containing estimated states.

Examples

```
if (requireNamespace("tsibbledata")) {
library(tsibble)
beer_components <- tsibbledata::aus_production %>%
  model(
    prophet = prophet(Beer ~ season("year", 4, type = "multiplicative"))
  ) %>%
  components()

beer_components

autoplot(beer_components)

library(ggplot2)
library(lubridate)
```

```
beer_components %>%
  ggplot(aes(x = quarter(Quarter), y = year, group = year(Quarter))) +
  geom_line()
}
```

`fitted.fbl_prophet` *Extract fitted values*

Description

Extracts the fitted values from an estimated Prophet model.

Usage

```
## S3 method for class 'fbl_prophet'
fitted(object, ...)
```

Arguments

`object`	A model for which forecasts are required.
`...`	Other arguments passed to methods

Value

A vector of fitted values.

`forecast.fbl_prophet` *Produce forecasts from the prophet model*

Description

If additional future information is required (such as exogenous variables or carrying capacities) by the model, then they should be included as variables of the `new_data` argument.

Usage

```
## S3 method for class 'fbl_prophet'
forecast(object, new_data, specials = NULL, times = 1000, ...)
```

Arguments

`object`	A model for which forecasts are required.
`new_data`	A tsibble containing the time points and exogenous regressors to produce forecasts for.
`specials`	(passed by `fabletools::forecast.mdl_df()`).
`times`	The number of sample paths simulated from the model's predictive distribution (default 1000). Each forecast is a sample distribution (`distributional::dist_sample()`) of this many paths, so `times` directly controls the time and memory used when forecasting many series or horizons (for example, with parallel workers via the **future** package). Reducing `times` (e.g. `times = 100`) speeds up forecasting and reduces the size of the forecast object, at the cost of noisier intervals. If `times = 0`, no paths are simulated and a point forecast is returned as a degenerate distribution (`distributional::dist_degenerate()`) of the model's predicted values, so prediction intervals are not available.
`...`	Currently unused and ignored.

Value

A list of forecasts.

See Also

`prophet::predict.prophet()`

Examples

```
if (requireNamespace("tsibbledata")) {
library(tsibble)
tsibbledata::aus_production %>%
  model(
    prophet = prophet(Beer ~ season("year", 4, type = "multiplicative"))
  ) %>%
  forecast()
}
```

---

generate.fbl_prophet *Generate responses from a prophet model*

Description

Simulates future (or in-sample) sample paths from a prophet model. Each path consists of a simulated trend (including future trend changes), the model's seasonal, holiday and regressor effects, and observation noise.

Usage

```
## S3 method for class 'fbl_prophet'
generate(x, new_data = NULL, bootstrap = FALSE, specials = NULL, ...)
```

Arguments

x	A fitted prophet model.
new_data	The data to simulate over. When used via fabletools::generate() it contains a .rep key for the replications, which are all simulated over the same time points.
bootstrap	If TRUE, then forecast distributions are computed using simulation with resampled errors.
specials	(passed by fabletools::forecast.mdl_df()).
...	Currently unused and ignored.

Details

By default the observation noise is normally distributed with the estimated standard deviation of the model (sigma_obs). If bootstrap = TRUE, the noise is instead resampled from the model's (mean-centred) residuals. When used via fabletools::generate(), the bootstrap argument of that function takes precedence and times sets the number of paths.

As with forecast.fbl_prophet(), any variables required by the model (regressors, carrying capacities and conditions) must be included in new_data. Estimated models using MCMC simulate each path using one (randomly selected) posterior draw of the parameters.

Value

A tsibble with the index and key of new_data, and the simulated values in .sim.

Examples

```
library(tsibble)
fit <- as_tsibble(USAccDeaths) %>%
  model(prophet(value ~ season("year", 4)))

generate(fit, h = 12, times = 5)
```

---

glance.fbl_prophet *Glance a prophet model*

Description

A glance of a prophet provides the residual's standard deviation (sigma), and a tibble containing the selected changepoints with their trend adjustments.

Usage

```
## S3 method for class 'fbl_prophet'
glance(x, ...)
```

Arguments

x	model or other R object to convert to single-row data frame
...	other arguments passed to methods

Value

A one row tibble summarising the model's fit.

Examples

```
if (requireNamespace("tsibbledata")) {
library(tsibble)
library(dplyr)
fit <- tsibbledata::aus_production %>%
  model(
    prophet = prophet(Beer ~ season("year", 4, type = "multiplicative"))
  )

glance(fit)
}
```

---

prophet *Prophet procedure modelling*

Description

Prepares a prophet model specification for use within the fable package.

Usage

```
prophet(formula, ...)
```

Arguments

formula	A symbolic description of the model to be fitted of class formula.
...	Additional arguments for estimating the model. These are mcmc.samples and backend (described below), with any others passed on to prophet::fit.prophet() and then to the Stan algorithm (for example algorithm, iter or init for optimisation, or control when mcmc.samples > 0), for example prophet(y ~ season("year"), algorithm = "Newton").

Details

The prophet modelling interface uses a `formula` based model specification (`y ~ x`), where the left of the formula specifies the response variable, and the right specifies the model's predictive terms. Like any model in the fable framework, it is possible to specify transformations on the response.

A prophet model supports piecewise linear or exponential growth (trend), additive or multiplicative seasonality, holiday effects and exogenous regressors. These can be specified using the 'specials' functions detailed below. The introduction vignette provides more details on how to model data using this interface to prophet: `vignette("intro", package="fable.prophet")`.

Estimation

By default the model is estimated by maximum a posteriori (MAP) optimisation. Two arguments of `prophet::prophet()` can be given in `...` of `prophet()`:

- `mcmc.samples`: If greater than 0, the model is estimated using this many MCMC (Hamiltonian Monte Carlo) iterations, which is slower. Parameters are then summarised by their posterior means in `tidy()` and `glance()`, and forecast sample paths include parameter uncertainty.
- `backend`: The Stan backend, either `"rstan"` or `"cmdstanr"` (which requires the **cmdstanr** package). If `NULL` (the default), the backend is chosen by `prophet::prophet()`, using `"rstan"` unless the environment variable `R_STAN_BACKEND` is set to `"CMDSTANR"`.

Specials

growth: The `growth` special is used to specify the trend parameters.

```
growth(type = c("linear", "logistic", "flat"), capacity = NULL, floor = NULL,
       changepoints = NULL, n_changepoints = 25, changepoint_range = 0.8,
       changepoint_prior_scale = 0.05)
```

`type`	The type of trend (linear, logistic or flat). A flat trend is a constant level with no changepoints.
`capacity`	The carrying capacity for when `type` is "logistic".
`floor`	The saturating minimum for when `type` is "logistic".
`changepoints`	A vector of dates/times for changepoints. If `NULL`, changepoints are automatically selected.
`n_changepoints`	The total number of changepoints to be selected if `changepoints` is `NULL`
`changepoint_range`	Proportion of the start of the time series where changepoints are automatically selected.
`changepoint_prior_scale`	Controls the flexibility of the trend.

season: The `season` special is used to specify a seasonal component. This special can be used multiple times for different seasonalities.

Warning: The inputs controlling the seasonal `period` is specified is different than `prophet::prophet()`. Numeric inputs are treated as the number of observations in each seasonal period, not the number of days.

```
season(period = NULL, order = NULL, prior_scale = 10,
       type = c("additive", "multiplicative"), name = NULL, condition = NULL)
```

`period`	The periodic nature of the seasonality. If a number is given, it will specify the number of observations in each
`order`	The number of terms in the partial Fourier sum. The higher the `order`, the more flexible the seasonality can b

prior_scale	Used to control the amount of regularisation applied. Reducing this will dampen the seasonal effect.
type	The nature of the seasonality. If "additive", the variability in the seasonal pattern is fixed. If "multiplicative", t
name	The name of the seasonal term (allowing you to name an annual pattern as 'annual' instead of 'year' or 365.2
condition	A bare column name of a logical variable, for conditional seasonality (see prophet::add_seasonality()).

holiday: The holiday special is used to specify a tsibble containing holidays for the model, and/or a country whose built-in holidays are included.

```
holiday(holidays = NULL, prior_scale = 10L, country = NULL)
```

holidays	A tsibble containing a set of holiday events. The event name is given in the 'holiday' column, and the event
prior_scale	Used to control the amount of regularisation applied. Reducing this will dampen the holiday effect.
country	A country name or code (e.g. "AU") for which to include prophet's built-in holidays, see prophet::add_cou

xreg: The xreg special is used to include exogenous regressors in the model. This special can be used multiple times for different regressors with different arguments. Exogenous regressors can also be used in the formula without explicitly using the xreg() special, which will then use the default arguments.

```
xreg(..., prior_scale = NULL, standardize = "auto", type = NULL)
```

...	A set of bare expressions that are evaluated as exogenous regressors
prior_scale	Used to control the amount of regularisation applied. Reducing this will dampen the regressor effect.
standardize	Should the regressor be standardised before fitting? If "auto", it will standardise if the regressor is not binary.
type	Does the effect of the regressor vary proportionally to the level of the series? If so, "multiplicative" is best. Ot

Non-daily data

A model without any terms (prophet(y), a bare response with no right hand side) enables prophet's yearly, weekly and daily seasonalities, as in prophet::prophet(). Unlike prophet's default ("auto"), these are switched on regardless of the data's frequency or length, so for monthly or quarterly data (or any data coarser than daily) the weekly and daily seasonal terms are not meaningful and prophet(y) is not recommended. Instead, specify the seasonality explicitly with season(). Any formula (such as prophet(y ~ season(...))) uses only the seasonalities that you specify, with no automatic seasonalities. For example, use prophet(y ~ season(period = "year", order = 4)) for quarterly data, or order = 6 for monthly data. A season() period given as a string is converted using the data's interval, and a number is the count of observations per period. A Fourier order is required for periods other than a year, week or day.

Performance and memory

Fitting prophet models is relatively slow, and each forecast stores times simulated sample paths (see forecast.fbl_prophet()). When modelling many series, consider the following:

- Reduce times in forecast() (for example times = 100), or use times = 0 for point forecasts only. This reduces time and the size of the forecast object.

- Fitted models drop the raw Stan output, but keep the data used to fit the model, so very large mables can be made smaller by forecasting then discarding the models.
- Models are independent, so series can be fit and forecast in parallel using the **future** package, for example with `future::plan(future::multisession)`.
- MCMC estimation (`mcmc.samples`) is much slower than the default optimisation.

Model evaluation

Prophet's `cross_validation()` and `performance_metrics()` are not needed for this interface. Use `tsibble::stretch_tsibble()` to create cross-validation folds, fit models on them and `fabletools::accuracy()` to evaluate the forecasts.

See Also

- `prophet::prophet()`
- Prophet homepage
- Prophet R package
- Prophet Python package

Examples

```
library(tsibble)
as_tsibble(USAccDeaths) %>%
  model(
    prophet = prophet(value ~ season("year", 4, type = "multiplicative"))
  )
```

`refit.fbl_prophet` *Refit a prophet model*

Description

Applies a prophet model to a new dataset. By default (`reestimate = FALSE`) the estimated parameters (including the trend changepoints and the scaling of the data) are kept, and only the fitted values, residuals and components are recomputed for `new_data`. If `reestimate = TRUE`, the model is instead estimated again on `new_data`, using the same specification and estimation arguments (such as `mcmc.samples`, `backend` and any others given to `prophet()`) as the original model.

Usage

```
## S3 method for class 'fbl_prophet'
refit(object, new_data, specials = NULL, reestimate = FALSE, ...)
```

Arguments

object	A model for which forecasts are required.
new_data	A tsibble containing the time points and exogenous regressors to produce forecasts for.
specials	(passed by fabletools::forecast.mdl_df()).
reestimate	If TRUE, the coefficients for the fitted model will be re-estimated to suit the new data.
...	Currently unused and ignored.

Details

Prophet does not support updating a fitted model with new observations, so re-estimating is a fresh fit (the previous parameters are not used to initialise it), and no stream() method is defined.

With reestimate = FALSE, the data should span the time period that the model is to be evaluated over, and any variables required by the model (regressors, carrying capacities and conditions) must be in new_data.

Value

A refitted model.

Examples

```
library(tsibble)
fit <- as_tsibble(USAccDeaths) %>%
  dplyr::filter(index < yearmonth("1977 Jan")) %>%
  model(prophet(value ~ season("year", 4)))

# Evaluate the estimated model on the full series
refit(fit, as_tsibble(USAccDeaths))

# Estimate the model again on the full series
refit(fit, as_tsibble(USAccDeaths), reestimate = TRUE)
```

---

residuals.fbl_prophet *Extract model residuals*

Description

Extracts the residuals from an estimated Prophet model.

Usage

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

Arguments

object	A model for which forecasts are required.
...	Other arguments passed to methods

Value

A vector of residuals.

---

tidy.fbl_prophet *Extract estimated coefficients from a prophet model*

Description

Extract estimated coefficients from a prophet model

Usage

```
## S3 method for class 'fbl_prophet'
tidy(x, ...)
```

Arguments

x	An object to be converted into a tidy tibble::tibble().
...	Additional arguments to tidying method.

Details

The estimate of the growth (base_growth, trend_offset), seasonal and holiday terms are the model's parameters on prophet's internal (scaled) scale. For extra regressors, the estimate is instead the coefficient on the scale of the original data: additive regressors are reported in units of the response per unit of the regressor, and multiplicative regressors as the proportional change in the trend per unit of the regressor. This is the same quantity described by prophet::regressor_coefficients() (whose returned coef, in prophet 1.1.7, is not yet rescaled).

Value

A tibble containing the model's estimated parameters.

Examples

```
if (requireNamespace("tsibbledata")) {
library(tsibble)
library(dplyr)
fit <- tsibbledata::aus_production %>%
  model(
    prophet = prophet(Beer ~ season("year", 4, type = "multiplicative"))
```

```
  )

tidy(fit) # coef(fit) or coefficients(fit) can also be used
}
```

Index

∗ **package**
fable.prophet-package, 2

components.fbl_prophet, 3

distributional::dist_degenerate(), *5*
distributional::dist_sample(), *5*

fable.prophet *(*fable.prophet-package*)*, 2
fable.prophet-package, 2
fabletools::accuracy(), *10*
fabletools::dable(), *3*
fabletools::forecast.mdl_df(), *5*, *6*, *11*
fabletools::generate(), *6*
fitted.fbl_prophet, 4
forecast.fbl_prophet, 4
forecast.fbl_prophet(), *6*, *9*

generate.fbl_prophet, 5
glance(), *8*
glance.fbl_prophet, 6

prophet, 7
prophet(), *10*
prophet::add_country_holidays(), *9*
prophet::add_seasonality(), *9*
prophet::fit.prophet(), *7*
prophet::predict.prophet(), *5*
prophet::prophet(), *8–10*
prophet::prophet_plot_components(), *3*
prophet::regressor_coefficients(), *12*

refit.fbl_prophet, 10
residuals.fbl_prophet, 11

tibble::tibble(), *12*
tidy(), *8*
tidy.fbl_prophet, 12
tsibble::stretch_tsibble(), *10*