

Package: fable.rjd3 (via r-universe)

October 30, 2026

Type Package

Title 'JDemetra+' Seasonal Adjustment Models for 'fable'

Version 0.1.0

Description Provides an interface to JDemetra+ v3 seasonal adjustment algorithms (X-13ARIMA-SEATS and TRAMO-SEATS) for use with the 'fable' forecasting and decomposition framework. It enables seasonal adjustment on 'tsibble' objects using the standard 'model()' and 'components()' workflow.

Depends R (>= 4.1.0)

Imports rjd3jars(>= 0.0.6), rjd3toolkit (>= 3.0.0), rjd3x13 (>= 3.9.0.9000), rjd3tramoseats (>= 3.9.0.9000), rjd3workspace (>= 3.9.0), rJava, fabletools (>= 0.3.0), tsibble (>= 1.1.0), generics, distributional, rlang, stats, tibble, dplyr

VignetteBuilder knitr

Suggests feasts, knitr, rmarkdown, ggtime, ggplot2

SystemRequirements Java (>= 17)

License EUPL

URL <https://aqlt.github.io/fable.rjd3/>,
<https://github.com/AQLT/fable.rjd3>

BugReports <https://github.com/AQLT/fable.rjd3/issues>

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.1.1

Remotes github::rjdverse/rjd3jars, github::rjdverse/rjd3providers,
github::rjdverse/rjd3toolkit, github::rjdverse/rjd3tramoseats,
github::rjdverse/rjd3x13, github::rjdverse/rjd3workspace

Config/pak/sysreqs make default-jdk libicu-dev libprotobuf-dev protobuf-compiler libprotoc-dev

Repository <https://aqlt.r-universe.dev>

Date/Publication 2026-10-30 06:53:48 UTC
RemoteUrl <https://github.com/AQLT/fable.rjd3>
RemoteRef HEAD
RemoteSha 16b67b2bfe86b0c47b5abd4515ca8aed5c266e55

Contents

backcast	2
export_jws	3
forecast.jd3_model	4
glance.jd3_model	5
import_jws	7
jd3_dictionary	8
jd3_indicator	9
jd3_model	10
jd3_model_methods_extra	11
jd3_specials	13
outliers.jd3_model	14
REGARIMA	15
TRAMO	17
TRAMOSEATS	18
X13	19
Index	21

backcast	<i>Backcast of the raw series</i>
----------	-----------------------------------

Description

Backcasts the raw series, i.e. predicts the periods preceding its start, from the pre-adjustment model of a model fitted with [X13\(\)](#), [TRAMOSEATS\(\)](#), [REGARIMA\(\)](#) or [TRAMO\(\)](#). **fabletools** has no backcast generic, so this function plays the role of [forecast\(\)](#) for the past. The distribution is built as for forecasts (log-normal for log-transformed series, normal otherwise; see the *Distribution* section of [forecast.jd3_model](#)).

Usage

```
backcast(object, h, ...)  
  
## S3 method for class 'jd3_model'  
backcast(object, h, ...)  
  
## S3 method for class 'mdl_ts'  
backcast(object, h, ...)
```

```
## S3 method for class 'mdl_df'
backcast(object, h, ...)
```

Arguments

object	A fitted jd3_model (usually a column of a mable).
h	Number of periods to backcast, immediately before the start of the series.
...	Other arguments passed by forecast() (e.g. times), ignored.

Value

For a jd3_model or a mdl_ts, a tsibble with the h dates preceding the series and a distribution column dist (see **distributional**). For a mable, a tibble with the key columns, .model, the dates and dist, one block per model. Use mean() or median() on dist to get a point backcast; the median equals JDemetra+'s backcast.

See Also

[forecast.jd3_model](#), [jd3_indicator\(\)](#) for the raw indicators (y_b, y_eb...).

Examples

```
y <- tsibble::as_tsibble(AirPassengers)
mb <- model(y, x13 = X13(value))

bc <- backcast(mb, h = 12)
bc

# A single model
bc1 <- backcast(mb$x13[[1]], h = 12)
median(bc1$dist)
```

export_jws

Export a mable of JDemetra+ models as a new workspace

Description

Writes a new workspace readable by the JDemetra+ graphical interface, with one SA-item per model: raw series, estimation specification, reference specification, and the modelling context (calendars and regressors). A workspace has a single context: regressors of different models that share the same name but have different values (e.g. xreg(col) in several series) are renamed <name>_<sap>_<series>, in the context and in the specifications. Results, comments, priorities and links to data providers are not exported; the series are stored as static data.

Usage

```
export_jws(mb, file, replace = FALSE, sap = "sap", series = "series")
```

Arguments

mb	A mable of models fitted with X13() or TRAMOSEATS() .
file	Path of the workspace (xml master file) to create.
replace	Replace the workspace if file already exists. By default an existing file is never overwritten.
sap, series	Names of the key columns giving the SA-processing and the SA-item. Without them, a single SA-processing is created and the SA-item names are built from the other keys.

Value

file, invisibly.

See Also

[import_jws\(\)](#)

Examples

```
mb <- model(as_tsibble(AirPassengers), sa = X13(value))
export_jws(mb, tempfile(fileext = ".xml"))
```

forecast.jd3_model	<i>Forecast of the raw series</i>
--------------------	-----------------------------------

Description

Forecasts the raw series from the pre-adjustment model (RegARIMA / TRAMO) of a model fitted with [X13\(\)](#), [TRAMOSEATS\(\)](#), [REGARIMA\(\)](#) or [TRAMO\(\)](#). The values are read from the stored JDeme-tra+ result, so the seasonal adjustment itself is not affected. For the past, see [backcast\(\)](#).

Usage

```
## S3 method for class 'jd3_model'
forecast(object, new_data = NULL, ...)
```

Arguments

object	A fitted jd3_model (usually a column of a mable).
new_data	A tsibble of future dates, built by fabletools from the horizon h. It must start right after the end of the series.
...	Other arguments passed by forecast() (e.g. times), ignored.

Value

A distribution vector (see **distributional**) with one element per row of `new_data`: log-normal if the series is log-transformed, normal otherwise. Called through `forecast()` on a mable, the result is a fable.

Distribution

For log-transformed series, the forecast distribution is log-normal, with a median equal to JDemetra+'s forecast (a plain back-transformation of the forecast of the log). Its mean is therefore the bias-corrected forecast, and the difference grows with the horizon. For other series, the distribution is normal and its mean equals the median.

The default point forecast `.mean` of the resulting fable is the mean of the distribution. Use the `point_forecast` argument of `forecast()` to choose another summary, e.g. `point_forecast = list(.median = median)` to obtain JDemetra+'s forecast.

See Also

`backcast()`, `X13()`, `TRAMOSEATS()`, `jd3_indicator()` to extract the raw indicators (`y_f`, `y_ef`...).

Examples

```
y <- tsibble::as_tsibble(AirPassengers)
mb <- model(y, x13 = X13(value))

# The mean is the bias-corrected forecast
forecast(mb, h = 12)

# JDemetra+'s own forecast (median of the distribution)
forecast(mb, h = 12, point_forecast = list(.median = median))
```

glance.jd3_model

Methods for fitted JDemetra+ models

Description

Standard fabletools / generics verbs for models fitted with `X13()`, `TRAMOSEATS()`, `REGARIMA()` and `TRAMO()`. They can be applied directly on a mable. `components()` is only available for the seasonal adjustment models (`X13()` and `TRAMOSEATS()`).

Usage

```
## S3 method for class 'jd3_model'
glance(x, ids = NULL, ...)

## S3 method for class 'jd3_sa_model'
components(object, ...)
```

```
## S3 method for class 'jd3_model'
model_sum(x)

## S3 method for class 'jd3_model'
tidy(x, ...)

## S3 method for class 'jd3_model'
report(
  object,
  digits = max(3L, getOption("digits") - 3L),
  signif.stars = getOption("show.signif.stars"),
  thresholds_pval = getOption("fable.rjd3.thresholds_pval"),
  ...
)
```

Arguments

<code>ids</code>	For <code>glance()</code> : named or unnamed character vector of indicator ids (see jd3_dictionary()); the names become the column names. Numbers and strings give one column; tests give two columns, <code><name>_stat</code> and <code><name>_pvalue</code> ; other indicators give NA. If NULL, the default set of the method is used (information criteria, number of observations, tests on the residuals and on the seasonally adjusted series, M and Q statistics for X-13, SEATS tests for TRAMO-SEATS).
<code>...</code>	Currently unused.
<code>object, x</code>	A <code>jd3_model</code> (usually accessed through a mable).
<code>digits</code>	minimum number of significant digits to be used for most numbers.
<code>signif.stars</code>	logical; if TRUE, P-values are additionally encoded visually as ‘significance stars’ in order to help scanning of long coefficient tables. It defaults to the <code>show.signif.stars</code> slot of options .
<code>thresholds_pval</code>	Threshold used to interpret p-values statistic test. By default <code>c(Severe = 0.001, Bad = 0.01, Uncertain = 0.05, Good = Inf)</code> .

Value

- `components()`: a dable with the response and the columns trend, seasonal, irregular and season_adjust.
- `glance()`: a one-row tibble of diagnostics.
- `tidy()`: a tibble with one row per coefficient (ARIMA parameters, regression variables, outliers, calendar effects) and the columns term, estimate, std.error, statistic and p.value. Coefficients fixed in the specification are included, with missing standard errors and tests.
- `report()`: invisibly, the model; prints the decomposition mode or the transformation, the number of observations, the coefficients, the tests on the residuals and, for seasonal adjustment models, the quality diagnostics (M and Q statistics, residual seasonality tests on the adjusted series).
- `model_sum()`: a short string describing the model.

Examples

```
mb <- model(tsibble::as_tsibble(AirPassengers), x13 = X13(value))
components(mb)
glance(mb)
tidy(mb)
report(mb)
```

import_jws

*Import a JDemetra+ workspace as a mable***Description**

Reads all the SA-items of a workspace and fits them with [X13\(\)](#) or [TRAMOSEATS\(\)](#) using the specifications stored in the workspace: the estimation specification is the model specification, the reference specification is kept for the refresh policies (see [jd3_model_methods_extra](#)).

Usage

```
import_jws(file, compute = TRUE)
```

Arguments

file	Path to the master xml file of the workspace.
compute	Compute the workspace before reading it (required to access the SA-items).

Value

A mable with the key columns `sap` (SA-processing) and `series` (SA-item), and a model column `sa` holding X13 and TRAMOSEATS models. SA-items with another type of specification (e.g. RegARIMA only) are skipped with a warning.

See Also

[export_jws\(\)](#)

Examples

```
file <- system.file("workspaces", "workspace_test.xml", package = "rjd3workspace")
mb <- import_jws(file)
mb
```

jd3_dictionary	<i>List the indicators available in a fitted model</i>
----------------	--

Description

Wraps `rjd3toolkit::dictionary()` on the Java result of the model. The returned ids can be passed to `jd3_indicator()`.

Usage

```
jd3_dictionary(object, ...)  
  
## S3 method for class 'jd3_model'  
jd3_dictionary(object, ...)  
  
## S3 method for class 'mdl_ts'  
jd3_dictionary(object, ...)  
  
## S3 method for class 'mdl_df'  
jd3_dictionary(object, ...)
```

Arguments

<code>object</code>	A fitted mable (<code>mdl_df</code>), a <code>mdl_ts</code> or a <code>jd3_model</code> .
<code>...</code>	Currently unused.

Value

A data frame (class `JD3_FULL_DICTIONARY`) with one row per indicator: its id (name), description and output type. For a mable, the dictionary of the first model is returned.

See Also

[jd3_indicator\(\)](#)

Examples

```
mb <- model(tsibble::as_tsibble(AirPassengers), x13 = X13(value))  
head(jd3_dictionary(mb))
```

jd3_indicator

*Extract any JDemetra+ indicator from a fitted model***Description**

Wraps `rjd3toolkit::result()` on the Java result of the model. If the Java pointer is no longer valid (e.g. after `saveRDS()/readRDS()`), the model is transparently re-estimated (once per loaded object). The ids available are listed by `jd3_dictionary()`.

Usage

```
jd3_indicator(object, id, ...)

## S3 method for class 'jd3_model'
jd3_indicator(object, id, drop = TRUE, ...)

## S3 method for class 'mdl_ts'
jd3_indicator(object, id, drop = TRUE, ...)

## S3 method for class 'mdl_df'
jd3_indicator(object, id, simplify = TRUE, drop = TRUE, ...)
```

Arguments

<code>object</code>	A fitted mable (<code>mdl_df</code>), a <code>mdl_ts</code> or a <code>jd3_model</code> .
<code>id</code>	Indicator id (see <code>jd3_dictionary()</code>).
<code>...</code>	Currently unused.
<code>drop</code>	If TRUE (default) and a single <code>id</code> is given, the indicator itself is returned instead of a list of length one. Several ids always give a named list.
<code>simplify</code>	For a mable only: simplify the result according to the type of the indicator (see <i>Value</i>). FALSE always returns a list-column.

Value

For a `mdl_ts` or `jd3_model`, the indicator as returned by `rjd3toolkit::result()`: a number, a string, a ts, a matrix, a JD3_TEST...

For a mable, a tibble with the key columns, `.model`, `id` and:

- numbers or strings: a column value;
- ts series: one row per date, with columns `index` and `value`;
- tests: columns `statistic`, `p.value` and `distribution`;
- anything else (or `simplify = FALSE`): a list-column value with one element per model. With several ids (or `drop = FALSE`), a named list of such tibbles, one per id.

See Also

`jd3_dictionary()`

Examples

```
mb <- model(as_tsibble(AirPassengers), x13 = X13(value), ts = TRAMOSEATS(value))

jd3_indicator(mb, "likelihood.aic")      # one number per model
jd3_indicator(mb, "sa")                  # seasonally adjusted series, long format
jd3_indicator(mb, "residuals.kurtosis")  # test: statistic and p-value

# The raw object of a single model
jd3_indicator(mb$x13[[1]], "sa")
```

jd3_model

Extract the model object from a fitted mable cell

Description

Returns the `jd3_model` object stored in a fitted model. It is mostly useful for programming; `jd3_indicator()` and `jd3_dictionary()` call it internally.

Usage

```
jd3_model(x)
```

Arguments

`x` A `mdl_ts` (e.g. `mb$x13[[1]]`) or a `jd3_model`.

Value

A `jd3_model` object: a list holding the data (a `tsibble`), the series (`ts`), the final specification (`spec`), the reference specification (`ref_spec`), the modelling context, the engine, the name of the response and a cache for the Java result.

Examples

```
mb <- model(tsibble::as_tsibble(AirPassengers), x13 = X13(value))
m <- jd3_model(mb$x13[[1]])
class(m)
```

jd3_model_methods_extra

Fitted values, residuals, interpolation and updating of JDemetra+ models

Description

Methods for models fitted with [X13\(\)](#), [TRAMOSEATS\(\)](#), [REGARIMA\(\)](#) or [TRAMO\(\)](#). They can be applied directly on a mable.

Usage

```
## S3 method for class 'jd3_model'
fitted(object, ...)

## S3 method for class 'jd3_model'
residuals(object, type = c("innovation", "regression"), ...)

## S3 method for class 'jd3_model'
interpolate(object, new_data = NULL, specials = NULL, ...)

## S3 method for class 'jd3_model'
refit(
  object,
  new_data,
  specials = NULL,
  policy = c("Fixed", "FreeParameters", "Complete", "Outliers_StochasticComponent",
    "Outliers", "FixedParameters", "FixedAutoRegressiveParameters", "Current"),
  period = 0,
  start = NULL,
  end = NULL,
  ...
)

## S3 method for class 'jd3_model'
stream(
  object,
  new_data,
  specials = NULL,
  policy = c("Fixed", "FreeParameters", "Complete", "Outliers_StochasticComponent",
    "Outliers", "FixedParameters", "FixedAutoRegressiveParameters", "Current"),
  period = 0,
  start = NULL,
  end = NULL,
  ...
)
```

Arguments

<code>object</code>	A fitted <code>jd3_model</code> (usually a column of a mable).
<code>...</code>	Currently unused.
<code>type</code>	Residual type: "innovation" (default) or "regression".
<code>new_data</code>	A tibble. For <code>interpolate()</code> , the data with missing values (same dates as the data of the model). For <code>refit()</code> , the new full data. For <code>stream()</code> , the new observations only.
<code>specials</code>	Currently unused.
<code>policy</code>	Refresh policy, a character string (see the functions above for the available policies; default "Fixed": the estimated parameters are kept).
<code>period, start, end</code>	additional parameters used to specify the span when <code>policy = "Current"</code> or <code>policy = "Outliers"</code> or <code>policy = "Outliers_StochasticComponent"</code> <code>period</code> : numeric, number of observations in a year (12, 4...), compulsory, if false or missing, re-estimation with refreshed specification won't work. When <code>policy = "Outliers"</code> or <code>policy = "Outliers_StochasticComponent"</code> <code>start</code> has to be specified as the date from which outliers will be re-identified <code>end</code> is not used, if specified it will be ignored. When <code>policy = "Current"</code> <code>start</code> and <code>end</code> have to be both specified and indicate the span on which additive outliers (AO) will be added. Span definition: <code>start</code> and <code>end</code> : defined as arrays of two elements: year and first period (for example, <code>period = 12</code> and <code>start=c(1980, 1)</code> stands for January 1980)

Details

- `fitted()` returns the one-step-ahead fitted values of the pre-adjustment model, on the scale of the series. Together with `residuals()`, it makes `augment()` and `accuracy()` work.
- `residuals()` returns the innovation residuals of the pre-adjustment model (`type = "innovation"`) or the linearized series the ARIMA part is applied to (`type = "regression"`). Dates for which a residual is not defined (e.g. lost to differencing) are NA.
- `interpolate()` replaces the missing values of the series by the estimates of the model. `new_data` must cover the same dates as the data used to fit the model.
- `refit()` applies the model to new data, using a JDemetra+ refresh policy (see [rjd3x13::x13_refresh\(\)](#) and [rjd3tramoseats::tramoseats_refresh\(\)](#)): the specification is refreshed from the estimated model and the reference specification stored in the model (see the `ref_spec` argument of [X13\(\)](#)).
- `stream()` appends new observations to the data of the model and refreshes it in the same way. `new_data` contains the new observations only.

Value

- `fitted()`, `residuals()`: a numeric vector with one value per observation (NA where undefined). Called on a mable, a tibble with the index and the `.fitted` or `.resid` column.
- `interpolate()`: `new_data` with the missing values filled in.
- `refit()`, `stream()`: a `jd3_model` fitted on the new data. Called on a mable, a mable.

See Also

[X13\(\)](#), [TRAMOSEATS\(\)](#), [forecast.jd3_model\(\)](#), [jd3_model_methods](#)

Examples

```
y <- as_tsibble(AirPassengers)
mb <- model(y, x13 = X13(value))

# Fitted values and residuals (also used by augment() and accuracy())
fitted(mb)
residuals(mb) # innovation residuals
residuals(mb, type = "regression") # linearized series
augment(mb)
accuracy(mb)

# Interpolation of missing values
y_na <- y
y_na$value[c(20, 21)] <- NA
mb_na <- model(y_na, x13 = X13(value))
interpolate(mb_na, y_na)

# Updating a model when new observations arrive
start_new <- yearmonth("1960 Jan")
y_old <- y[y$index < start_new, ]
y_new <- y[y$index >= start_new, ]
mb_old <- model(y_old, x13 = X13(value))

# stream(): append the new observations, estimated parameters kept
mb_ext <- stream(mb_old, y_new)
components(mb_ext)

# refit(): apply the model to the full data
refit(mb_old, y) # refresh policy "Fixed" (default)
```

jd3_specials

Specials available in the model formula

Description

Specials are used on the right-hand side of the formula of [X13\(\)](#), [TRAMOSEATS\(\)](#), [REGARIMA\(\)](#) or [TRAMO\(\)](#), e.g. `X13(y ~ transform(fun = "Log") + xreg(strike))`. Each one records a call to a function of **rjd3toolkit** that is applied to the base specification (the `spec` argument).

Specials

- `spec(x)`: base specification (name or specification object); equivalent to the `spec` argument of [X13\(\)](#), [TRAMOSEATS\(\)](#), [REGARIMA\(\)](#) or [TRAMO\(\)](#).
- `basic(...)`: arguments passed to [rjd3toolkit::set_basic\(\)](#).

- `transform(...)`: arguments passed to `rjd3toolkit::set_transform()`.
- `estimate(...)`: arguments passed to `rjd3toolkit::set_estimate()`.
- `automodel(...)`: arguments passed to `rjd3toolkit::set_automodel()`.
- `outlier(...)`: arguments passed to `rjd3toolkit::set_outlier()`.
- `add_outlier(...)`: arguments passed to `rjd3toolkit::add_outlier()` (e.g. `type = "AO"`, `date = "2020-04-01"`).
- `easter(...)`: arguments passed to `rjd3toolkit::set_easter()`.
- `tradingdays(...)`: arguments passed to `rjd3toolkit::set_tradingdays()`.
- `arima(...)`: arguments passed to `rjd3toolkit::set_arima()`.
- `ramp(...)`: arguments passed to `rjd3toolkit::add_ramp()`.
- `usrdefvar(...)`: arguments passed to `rjd3toolkit::add_usrdefvar()`.
- `x11(...)`: arguments passed to `rjd3x13::set_x11()` (X-11 decomposition parameters; use with `X13()` only).
- `seats(...)`: arguments passed to `rjd3tramoseats::set_seats()` (SEATS decomposition parameters; use with `TRAMOSEATS()` only).
- `benchmarking(...)`: arguments passed to `rjd3toolkit::set_benchmarking()` (use with `X13()` or `TRAMOSEATS()` only).
- `xreg(..., regeffect = NULL, group = "r", lag = 0, coef = NULL)`: user-defined regressors taken from columns of the tsibble and added to the context. By default (`regeffect = NULL`), the regressors are only added to the context (with the `group="r"`). If `regeffect` is equals to "Undefined", "Trend", "Seasonal", "Irregular", "Series" or "SeasonallyAdjusted" then variable is also added to the specification using the `rjd3toolkit::add_usrdefvar()` function with the parameters `lag` and `coeff`. If `regeffect = "Calendar"`, then the variable is added to the specification using the `rjd3toolkit::set_tradingdays()` function.

Precedence

The base spec is applied first, then the specials in a fixed order (basic, transform, estimate, automodel, outlier, add_outlier, easter, tradingdays, arima, ramp, usrdefvar, benchmarking, x11, seats), then `xreg()`. Later settings override earlier ones. A special that does not apply to the method (e.g. `x11()` in `TRAMOSEATS()`) is ignored with a warning.

See Also

`X13()`, `TRAMOSEATS()`

outliers.jd3_model	<i>Identify the outliers of a JDemetra+ model</i>
--------------------	---

Description

Returns the outlying observations detected (or specified) in the pre-adjustment model of a model fitted with `X13()`, `TRAMOSEATS()`, `REGARIMA()` or `TRAMO()`: additive outliers (AO), level shifts (LS), transitory changes (TC) and seasonal outliers (SO). An outlier is dated at one observation, even when its effect (e.g. a level shift) lasts longer.

Usage

```
## S3 method for class 'jd3_model'
outliers(object, type = NULL, ...)
```

Arguments

<code>object</code>	A fitted <code>jd3_model</code> (usually a column of a mable).
<code>type</code>	Optional character vector of outlier types to keep, e.g. <code>c("AO", "LS")</code> . <code>NULL</code> (default) keeps all of them.
<code>...</code>	Currently unused.

Value

For a `jd3_model`, an integer vector with the row positions of the outliers in the data of the model (the contract of `fabletools::outliers()`). Called on a mable, `outliers()` returns the corresponding rows of the data, with the key columns and `.model`. The type, estimate and test statistics of the outliers are available in `tidy()`.

See Also

[X13\(\)](#), [TRAMOSEATS\(\)](#), [jd3_model_methods](#)

Examples

```
mb <- model(tsibble::as_tsibble(AirPassengers), x13 = X13(value))

# Rows of the data that are outliers
outliers(mb)

# Additive outliers only
outliers(mb, type = "AO")
```

REGARIMA

Reg-ARIMA Model

Description

Defines a Reg-ARIMA model (the pre-adjustment model used in X-13), without seasonal adjustment, to be used inside `fabletools::model()`. The model is estimated with `rjd3x13::jregarima()` and the Java result is kept in the fitted model. Arguments and specials are the same as for `X13()`, except `x11()`, `seats()` and `benchmarking()`, which do not apply and are ignored with a warning.

Usage

```
REGARIMA(
  formula,
  spec = c("rg4", "rg0", "rg1", "rg2c", "rg3", "rg5c"),
  context = NULL,
  ref_spec = spec,
  ...
)
```

Arguments

formula	Model formula: the name of the variable to adjust (y), or $y \sim \text{specials}$ (see jd3_specials).
spec	Base specification: a character name (see rjd3x13::regarima_spec() for the available names; the default is the first one) or a specification object from rjd3x13::regarima_spec() .
context	A modelling context created with rjd3toolkit::modelling_context() (e.g. for custom calendars), or NULL.
ref_spec	Reference specification stored in the model: it is the reference of the refresh policies used by refit() and stream() . Defaults to spec, that is the specification before its modification by the specials.
...	Currently unused.

Details

There is no decomposition, so [components\(\)](#) is not available. The other methods are: [fitted\(\)](#), [residuals\(\)](#), [forecast\(\)](#), [backcast\(\)](#), [outliers\(\)](#), [tidy\(\)](#), [glance\(\)](#) and [report\(\)](#).

Value

A model definition that can be passed to [fabletools::model\(\)](#). Once fitted, the mable column contains a `jd3_model` object.

See Also

[X13\(\)](#), [TRAMO\(\)](#), [jd3_specials](#), [jd3_model_methods](#)

Examples

```
y <- tsibble::as_tsibble(AirPassengers)
mb <- model(y, regarima = REGARIMA(value))
tidy(mb)
report(mb)

# Forecasts of the raw series
forecast(mb, h = 12)
```

TRAMO	<i>TRAMO Model</i>
-------	--------------------

Description

Defines a TRAMO model (the pre-adjustment model used in TRAMO-SEATS), without seasonal adjustment, to be used inside `fabletools::model()`. The model is estimated with `rjd3tramoseats::jtramo()` and the Java result is kept in the fitted model. Arguments and specials are the same as for `TRAMOSEATS()`, except `x11()`, `seats()` and `benchmarking()`, which do not apply and are ignored with a warning.

Usage

```
TRAMO(
  formula,
  spec = c("trfull", "tr0", "tr1", "tr2", "tr3", "tr4", "tr5"),
  context = NULL,
  ref_spec = spec,
  ...
)
```

Arguments

formula	Model formula: the name of the variable to adjust (y), or $y \sim \text{specials}$ (see jd3_specials).
spec	Base specification: a character name (see <code>rjd3tramoseats::tramo_spec()</code> for the available names; the default is the first one) or a specification object from <code>rjd3tramoseats::tramo_spec()</code> .
context	A modelling context created with <code>rjd3toolkit::modelling_context()</code> (e.g. for custom calendars), or NULL.
ref_spec	Reference specification stored in the model: it is the reference of the refresh policies used by <code>refit()</code> and <code>stream()</code> . Defaults to spec, that is the specification before its modification by the specials.
...	Currently unused.

Details

There is no decomposition, so `components()` is not available. The other methods are: `fitted()`, `residuals()`, `forecast()`, `backcast()`, `outliers()`, `tidy()`, `glance()` and `report()`.

Value

A model definition that can be passed to `fabletools::model()`. Once fitted, the mable column contains a `jd3_model` object.

See Also

[TRAMOSEATS\(\)](#), [REGARIMA\(\)](#), [jd3_specials](#), [jd3_model_methods](#)

Examples

```
y <- tsibble::as_tsibble(AirPassengers)
mb <- model(y, tramo = TRAMO(value))
tidy(mb)
report(mb)

# Forecasts of the raw series
forecast(mb, h = 12)
```

TRAMOSEATS

TRAMO-SEATS seasonal adjustment (JDemetra+ v3)

Description

Defines a TRAMO-SEATS model to be used inside `fabletools::model()`. The model is estimated with `rjd3tramoseats::jtramoseats()` and the Java result is kept in the fitted model. Arguments and specials are the same as for `X13()`.

Usage

```
TRAMOSEATS(
  formula,
  spec = c("rsafull", "rsa0", "rsa1", "rsa2", "rsa3", "rsa4", "rsa5"),
  context = NULL,
  ref_spec = spec,
  ...
)
```

Arguments

<code>formula</code>	Model formula: the name of the variable to adjust (y), or $y \sim \text{specials}$ (see jd3_specials).
<code>spec</code>	Base specification: a character name or a specification object from <code>rjd3tramoseats::tramoseats_spec</code> .
<code>context</code>	A modelling context created with <code>rjd3toolkit::modelling_context()</code> (e.g. for custom calendars), or NULL.
<code>ref_spec</code>	Reference specification stored in the model: it is the reference of the refresh policies used by <code>refit()</code> and <code>stream()</code> . Defaults to <code>spec</code> , that is the specification before its modification by the specials.
<code>...</code>	Currently unused.

Value

A model definition that can be passed to `fabletools::model()`. Once fitted, the `mable` column contains a `jd3_model` object. Use `components()` to get the decomposition as a `dable` (columns `trend`, `seasonal`, `irregular` and `season_adjust`).

See Also

[X13\(\)](#), [TRAMO\(\)](#), [jd3_specials](#), [jd3_model_methods](#)

Examples

```
y <- tsibble::as_tsibble(AirPassengers)

mb <- model(y, ts = TRAMOSEATS(value))
components(mb)

# Fit both methods side by side and plot the seasonally adjusted series
mb <- model(y, x13 = X13(value), ts = TRAMOSEATS(value))
if (requireNamespace("ggtime", quietly = TRUE) && requireNamespace("ggplot2", quietly = TRUE)) {
  ggplot2::autoplot(components(mb))
}
```

X13

*X-13ARIMA seasonal adjustment (JDemetra+ v3)***Description**

Defines an X-13ARIMA model to be used inside [fabletools::model\(\)](#). The model is estimated with [rjd3x13::jx13\(\)](#) and the Java result is kept in the fitted model, so that all JDemetra+ indicators can be extracted afterwards (see [jd3_indicator\(\)](#)).

Usage

```
X13(
  formula,
  spec = c("rsa4", "rsa0", "rsa1", "rsa2c", "rsa3", "rsa5c"),
  context = NULL,
  ref_spec = spec,
  ...
)
```

Arguments

formula	Model formula: the name of the variable to adjust (y), or $y \sim \text{specials}$ (see jd3_specials).
spec	Base specification: a character name (see rjd3x13::x13_spec() for the available names; the default is the first one) or a specification object from rjd3x13::x13_spec() . The specials of the formula are applied to it.
context	A modelling context created with rjd3toolkit::modelling_context() (e.g. for custom calendars), or NULL.

<code>ref_spec</code>	Reference specification stored in the model: it is the reference of the refresh policies used by <code>refit()</code> and <code>stream()</code> . Defaults to <code>spec</code> , that is the specification before its modification by the specials.
<code>...</code>	Currently unused.

Details

The specification can be set in two (combinable) ways:

1. **Ex ante**, with the `spec` argument: a name or a specification object created with `rjd3x13::x13_spec()` and modified with the `set_*`() functions of **rjd3toolkit**.
2. **In the formula**, with specials such as `transform()` or `xreg()` (see [jd3_specials](#)). This is the only way to use regressors that are columns of the tsibble.

Specials override the base spec.

Value

A model definition that can be passed to `fabletools::model()`. Once fitted, the mable column contains a `jd3_model` object. Use `components()` to get the decomposition as a dable (columns trend, seasonal, irregular and season_adjust).

See Also

[TRAMOSEATS\(\)](#), [REGARIMA\(\)](#), [jd3_specials](#), [jd3_model_methods](#), [jd3_indicator\(\)](#), [jd3_dictionary\(\)](#)

Examples

```
y <- tsibble::as_tsibble(AirPassengers)

# Default specification
mb <- model(y, x13 = X13(value))
components(mb)

# Specification created ex ante with rjd3toolkit
my_spec <- rjd3x13::x13_spec("rsa3")
my_spec <- rjd3toolkit::set_transform(my_spec, fun = "Log")
mb <- model(y, x13 = X13(value, spec = my_spec))

# Specification set in the formula with specials
mb <- model(
  y,
  x13 = X13(value ~ transform(fun = "Log") +
    outlier(outliers.type = c("A0", "LS")))
)

# Regressor taken from a column of the tsibble
y$shock <- as.numeric(y$index == tsibble::yearmonth("1955 Jan"))
mb <- model(y, x13 = X13(value ~ xreg(shock, regeffect = "Series")))
```

Index

backcast, 2
backcast(), 4, 5

components(), 18, 20
components.jd3_sa_model
 (glance.jd3_model), 5

export_jws, 3
export_jws(), 7

fabletools::model(), 15–20
fabletools::outliers(), 15
fitted.jd3_model
 (jd3_model_methods_extra), 11
forecast(), 2
forecast.jd3_model, 2, 3, 4
forecast.jd3_model(), 13

glance.jd3_model, 5

import_jws, 7
import_jws(), 4
interpolate.jd3_model
 (jd3_model_methods_extra), 11

jd3_dictionary, 8
jd3_dictionary(), 6, 9, 10, 20
jd3_indicator, 9
jd3_indicator(), 3, 5, 8, 10, 19, 20
jd3_model, 10
jd3_model_methods, 13, 15–17, 19, 20
jd3_model_methods(glance.jd3_model), 5
jd3_model_methods_extra, 7, 11
jd3_specials, 13, 16–20

model_sum.jd3_model(glance.jd3_model),
 5

options, 6
outliers.jd3_model, 14
readRDS(), 9

refit.jd3_model
 (jd3_model_methods_extra), 11
REGARIMA, 15
REGARIMA(), 2, 4, 5, 11, 13, 14, 17, 20
report.jd3_model(glance.jd3_model), 5
residuals.jd3_model
 (jd3_model_methods_extra), 11
rjd3toolkit::add_outlier(), 14
rjd3toolkit::add_ramp(), 14
rjd3toolkit::add_usrdefvar(), 14
rjd3toolkit::dictionary(), 8
rjd3toolkit::modelling_context(),
 16–19
rjd3toolkit::result(), 9
rjd3toolkit::set_arma(), 14
rjd3toolkit::set_automodel(), 14
rjd3toolkit::set_basic(), 13
rjd3toolkit::set_benchmarking(), 14
rjd3toolkit::set_easter(), 14
rjd3toolkit::set_estimate(), 14
rjd3toolkit::set_outlier(), 14
rjd3toolkit::set_tradingdays(), 14
rjd3toolkit::set_transform(), 14
rjd3tramoseats::jtramo(), 17
rjd3tramoseats::set_seats(), 14
rjd3tramoseats::tramo_spec(), 17
rjd3tramoseats::tramoseats_refresh(),
 12
rjd3tramoseats::tramoseats_spec(), 18
rjd3x13::jregarima(), 15
rjd3x13::jx13(), 19
rjd3x13::regarima_spec(), 16
rjd3x13::set_x11(), 14
rjd3x13::x13_refresh(), 12
rjd3x13::x13_spec(), 19, 20

saveRDS(), 9
stream.jd3_model
 (jd3_model_methods_extra), 11

`tidy()`, 15
`tidy.jd3_model (glance.jd3_model)`, 5
`TRAMO`, 17
`TRAMO()`, 2, 4, 5, 11, 13, 14, 16, 19
`TRAMOSEATS`, 18
`TRAMOSEATS()`, 2, 4, 5, 7, 11, 13–15, 17, 20

`X13`, 19
`X13()`, 2, 4, 5, 7, 11–16, 18, 19