Package ‘Rlgt’

September 16, 2023

Type Package
Title Bayesian Exponential Smoothing Models with Trend Modifications
Version 0.2-1
URL https://github.com/cbergmeir/Rlgt
Date 2023-09-14
Description An implementation of a number of Global Trend models for time series forecasting that are Bayesian generalizations and extensions of some Exponential Smoothing models. The main differences/additions include 1) nonlinear global trend, 2) Student-t error distribution, and 3) a function for the error size, so heteroscedasticity. The methods are particularly useful for short time series. When tested on the well-known M3 dataset, they are able to outperform all classical time series algorithms. The models are fitted with MCMC using the 'rstan' package.

License GPL-3
Encoding UTF-8
LazyData true
ByteCompile true
Depends R (>= 3.4.0), Rcpp (>= 0.12.0), methods, rstantools, forecast, truncnorm
Imports rstan (>= 2.26.0), sn
LinkingTo StanHeaders (>= 2.26.0), rstan (>= 2.26.0), BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.0), RcppParallel (>= 5.0.2)
SystemRequirements GNU make
NeedsCompilation yes
RoxygenNote 7.2.3
Suggests knitr, rmarkdown
VignetteBuilder knitr
Author Slawek Smyl [aut], Christoph Bergmeir [aut, cre], Erwin Wibowo [aut],
Rlgt-package

Description

An implementation of Bayesian ETS models named LGT (for non-seasonal time series data) and SGT (for seasonal time series data). These models have been tested on the M3 competition dataset in which they outperform all of the models originally participating in the competition.

Getting started

The best way to get started with the package is to have a look at the vignettes and the various demos that ship with the package. There is a vignette with examples of how to use the various methods included in the package, and a vignette that discusses some of the theoretical background.

As to the demos, you can find their source code in the "demo" subfolder in the package sources (available on CRAN). There are some basic demos and other more advanced ones that run on subsets of the M3 dataset and run potentially for hours.

The package contains models for seasonal and non-seasonal data, allows for external regressors, and different error distributions. In the following, we briefly also present some of the theoretical background of the methods.
LGT (Local and Global Trend)

The LGT model is constructed based on Holt’s linear trend method. The model is designed to allow for a more general term of error by allowing for heteroscedasticity and an addition of constant "global" trend in the model.

Model Equations:
In terms of mathematical notation, the model can be fully represented as follows:

\[ y_{t+1} \sim \text{Student}(\nu, \hat{y}_{t+1}, \sigma_{t+1}) \quad (eq.1.1) \]
\[ \hat{y}_{t+1} = l_t + \gamma l_t^\rho + \lambda b_t \quad (eq.1.2) \]
\[ l_{t+1} = \alpha y_{t+1} + (1 - \alpha) (l_t) \quad (eq.1.3) \]
\[ b_{t+1} = \beta (l_{t+1} - l_t) + (1 - \beta) b_t \quad (eq.1.4) \]
\[ \hat{\sigma}_{t+1} = \sigma l_t^\tau + \xi \quad (eq.1.5) \]

Notations:
y\_t value of the dependent variable of interest at time t
\[ \hat{y}_{t+1} \] forecasted value of y at time t+1 given information up to time t
\[ \hat{\sigma}_{t+1} \] forecasted deviation at time t+1 given information up to time t
\[ l_t \] level at time t
\[ b_t \] local trend at time t

Parameters:
\[ \nu \] degrees of freedom of the t-distribution
\[ \gamma \] coefficient of the global trend
\[ \rho \] power coefficient of the global trend
\[ \lambda \] damping coefficient of the local trend
\[ \alpha \] smoothing parameter for the level term
\[ \beta \] smoothing parameter for the local trend term
\[ \sigma \] coefficient of the size of error function
\[ \tau \] power coefficient of the size of error function
\[ \xi \] base or minimum value of the size of error function

SGT (Seasonal, Global Trend)

The SGT model was designed as a seasonal counterpart to the LGT model. Similar to LGT, this model is devised to allow for a global trend term and heteroscedastic error.

Model Equations:

\[ y_{t+1} \sim \text{Student}(\nu, \hat{y}_{t+1}, \sigma_{t+1}) \quad (eq.2.1) \]
\[ \hat{y}_{t+1} = (l_t + \gamma l_t^\rho) s_{t+1} \quad (eq.2.2) \]
\[ l_{t+1} = \alpha \frac{y_{t+1}}{s_{t+1}} + (1 - \alpha) (l_t) \quad (eq.2.3) \]
\[ s_{t+m+1} = \zeta \frac{y_{t+1}}{l_{t+1}} + (1 - \zeta) s_{t+1} \quad (eq.2.4) \]
\[ \hat{\sigma}_{t+1} = \sigma \hat{y}_{t+1}^\tau + \xi \quad (eq.2.5) \]
Additional Notations:

$s_t$  seasonality factor at time $t$

$m$  number of seasons in the data (e.g. 12 for monthly, 4 for quarterly)

Additional Parameters:

$\zeta$  smoothing parameter for the seasonality terms

S2GT (Double Seasonal, Global Trend)

S2GT is designed as an extension to SGT for time series data which exhibit two seasonality patterns.

Model Equations:

\[
y_{t+1} \sim \text{Student} \left( \nu, \hat{y}_{t+1}, \sigma_{t+1} \right) \quad (eq.3.1)
\]

\[
\hat{y}_{t+1} = (l_t + \gamma l_t^\prime) s_{t+1} w_{t+1} \quad (eq.3.2)
\]

\[
l_t = \alpha \frac{y_t}{s_t w_t} + (1 - \alpha) (l_{t-1}) \quad (eq.3.3)
\]

\[
s_{t+m} = \zeta \frac{y_t}{l_t w_t} + (1 - \zeta) s_t \quad (eq.3.4)
\]

\[
w_{t+d} = \delta \frac{y_t}{l_t s_t} + (1 - \delta) w_t \quad (eq.3.5)
\]

\[
\hat{\sigma}_{t+1} = \sigma y_{t+1} + \xi \quad (eq.3.6)
\]

Additional Notations:

$w_t$  second seasonality factor prevailing at time $t$

$d$  number of (second) seasons in a complete period (e.g. 12 for monthly, 4 for quarterly)

Additional Parameters:

$\delta$  smoothing parameters for the second seasonality factors

NA

The best way to get started with the package is to have a look at the vignettes and the various demos that ship with the package. There is a vignette with examples of how to use the various methods included in the package, and a vignette that discusses some of the theoretical background.

As to the demos, you can find their source code in the "demo" subfolder in the package sources (available on CRAN). There are some basic demos and other more advanced ones that run on subsets of the M3 dataset and run potentially for hours.

The package contains models for seasonal and non-seasonal data, allows for external regressors, and different error distributions. In the following, we briefly also present some of the theoretical background of the methods.
Description

produce forecasts from an `rlgtfit` object

Usage

```r
## S3 method for class 'rlgtfit'
forecast(
  object,
  xreg = NULL,
  h = ifelse(frequency(object$x) > 1, 2 * frequency(object$x), 10),
  level = c(80, 95),
  NUM_OF_TRIALS = 2000,
  ...
)
```

Arguments

- `object`: `rlgtfit` object
- `xreg`: input regression matrix
- `h`: forecasting horizon (the default is 10 for annual and 2*periods otherwise)
- `level`: confidence levels for prediction intervals a.k.a. coverage percentiles. Must be between 0 and 100.
- `NUM_OF_TRIALS`: number of simulations to run. Suggested range is between (1000,5000), but it needs to be higher for good coverage for very high levels, e.g. 99.8.
- `...`: currently not used

Value

returns a forecast object compatible with the forecast package in R

Examples

```r
# The following is a toy example that runs within a few seconds. To get good fitting results the number of iterations should be set to at least 2000, and 4 chains should be used (the default). To speed up computation the number of cores should also be adjusted (default is 4).

rlgt_model <- rlgt(lynx,
  control=rlgt.control(MAX_NUM_OF_REPEATS=1, NUM_OF_ITER=50, NUM_OF_CHAINS = 1,
                       NUM_OF_CORES = 1), verbose=TRUE)

# print the model details
print(rlgt_model)
```
# Produce Forecasts for the next 10 years
forecast_result <- forecast(rlgt_model, h = 10, level=c(80, 95, 98))

plot(forecast_result,main="Forecasting lynx dataset with LGT model")

### iclaims.example Weekly Initial Claims of US Unemployment Benefits & Google Trends Queries

**Description**
A dataset containing the weekly initial claims for US unemployment benefits against a few related Google trend queries from Jan 2010 - June 2018. This aims to mimick the dataset from Scott and Varian (2014).

**Usage**

data("iclaims.example")

**Format**
A data frame with 443 rows and 5 variables with log-transformation

- **week** date of records starting by Mondays with US calendar format
- **claims** weekly initial claims of unemployment benefits in thousands
- **trend.unemploy** normalized trend queries retrieved from gtrendsR API
- **trend.filling** normalized trend queries retrieved from gtrendsR API
- **trend.job** normalized trend queries retrieved from gtrendsR API

**References**

U.S. Employment and Training Administration, Initial Claims [ICNSA], retrieved from FRED, Federal Reserve Bank of St. Louis; [https://fred.stlouisfed.org/series/ICNSA](https://fred.stlouisfed.org/series/ICNSA), October 27, 2018.


An interface for retrieving and displaying the information returned online by Google Trends is provided. Trends (number of hits) over the time as well as geographic representation of the results can be displayed. [https://CRAN.R-project.org/package=gtrendsR](https://CRAN.R-project.org/package=gtrendsR)

initModel

Initialize a model from the Rlgt family

Description
This is an internal function that usually won’t be called by users directly. It validates the model type and generates the corresponding list of parameters for the model.

Usage

```
initModel(
  model.type = NULL,
  use.regression = FALSE,
  seasonalityMethodId = 0,
  levelMethodId = 0,
  useSmoothingMethodForError = FALSE
)
```

Arguments

- `model.type`: type of the forecasting model selected, a character object
- `use.regression`: binary parameter indicating whether additional regressors will be used for forecasting in multivariate settings.
- `seasonalityMethodId`: Seasonality method Id (0- HW, 1- generalized).
- `levelMethodId`: Level method Id.
- `useSmoothingMethodForError`: if the non-standard function for error size should be used, one based on smoothed innovations or surprises

Value

an Rlgt skeleton model

posterior_interval.rlgtfit

rlgtfit posterior interval

Description
This is a method of the link{rlgtfit} class to produce posterior intervals

Usage

```
## S3 method for class 'rlgtfit'
posterior_interval(object, prob = 0.9, type = "central", ...)
```
Arguments

object   an object of class rlgtfit
prob     percentile level to be generated (multiple values can be accepted as a vector)
type     currently only central is available
...      currently not in use

Value

certainty interval

Examples

# The following is a toy example that runs within a few seconds. To get good
# fitting results the number of iterations should be set to at least 2000, and
# 4 chains should be used (the default). To speed up computation the number of
# cores should also be adjusted (default is 4).

rlgt_model <- rlgt(lynx,
    control=rlgt.control(MAX_NUM_OF_REPEATS=1, NUM_OF_ITER=50, NUM_OF_CHAINS = 1,
                        NUM_OF_CORES = 1), verbose=TRUE)

# print the model details
posterior_interval(rlgt_model)
rlgt

Fit an Rlgt model

Description

The main function to fit an rlgt model. It fits the parameter values with MCMC.

Usage

rlgt(
  y,
  seasonality = 1,
  seasonality2 = 1,
  seasonality.type = c("multiplicative", "generalized"),
  error.size.method = c("std", "innov"),
  level.method = c("HW", "seasAvg", "HW_sAvg"),
  xreg = NULL,
  control = rlgt.control(),
  verbose = FALSE,
  method = "Stan",
  experimental = ""
)

Arguments

y  time-series data for training (provided as a numeric vector, or a ts, or msts object).
seasonality This specification of seasonality will be overridden by frequency of y, if y is of
ts or msts class. 1 by default, i.e. no seasonality.
seasonality2 Second seasonality. If larger than 1, a dual seasonality model will be used.
However, this is experimental. If not specified and multiple seasonality time
series (of msts class) is used, a single seasonality model will be applied, one
with seasonality equal to the largest of seasonalities of the time series. 1 by
default, i.e. no seasonality or single seasonality.
seasonality.type Either "multiplicative" (default) or "generalized". The latter seasonality gener-
alizes additive and multiplicative seasonality types.
error.size.method Function providing size of the error. Either "std" (monotonically, but slower
than proportionally, growing with the series values) or "innov" (proportional to
a smoothed abs size of innovations, i.e. surprises)
level.method "HW", "seasAvg", "HW_sAvg". Here, "HW" follows Holt-Winters approach.
"seasAvg" calculates level as a smoothed average of the last seasonality num-
ber of points (or seasonality2 of them for the dual seasonality model), and
HW_sAvg is an weighted average of HW and seasAvg methods.
xreg
- Optionally, a vector or matrix of external regressors, which must have the same number of rows as y.

control
- list of control parameters, e.g. hyperparameter values for the model’s prior distributions, number of fitting interations etc.

verbose
- whether verbose information should be printed (Boolean value only), default FALSE.

method
- Sampling method, default Stan.

experimental
- Run different versions ("nostudent", "noglobal", "nohet", "ets") for ablation studies

Value

rlgtfit object

Examples

# The following is a toy example that runs within a few seconds. To get good fitting results the number of iterations should be set to at least 2000, and # 4 chains should also be adjusted (default is 4).

rlgt_model <- rlgt(lynx,
  control=rlgt.control(MAX_NUM_OF_REPEATS=1, NUM_OF_ITER=50, NUM_OF_CHAINS = 1,
                        NUM_OF_CORES = 1), verbose=TRUE)

# print the model details
print(rlgt_model)

## Not run: demo(exampleScript)

rlgt.control

Sets and initializes the control parameters

Description

This function initializes and sets the control parameters, i.e. hyperparameter values which control the prior distribution of the rlgtfit model. The purpose of this function is mainly to provide a default value for each of the hyperparameters. The function also accepts a customised set of values of the parameters as provided in the input of this function. This function is used in conjunction with the rlgt function.

Usage

rlgt.control(
  ADAPT_DELTA = 0.9,
  MAX_TREEDEPTH = 12,
NUM_OF_CHAINS = 4,
NUM_OF_CORES = 4,
ADD_JITTER = TRUE,
CAUCHY_SD_DIV = 150,
NUM_OF_ITER = 5000,
MAX_NUM_OF_REPEATS = 2,
MAX_RHAT_ALLOWED = 1.006,
NUM_OF_SEASON_INIT_CYCLES = 3,
MIN_NU = 2,
MAX_NU = 20,
MIN_POW_TREND = -0.5,
MAX_POW_TREND = 1,
POW_TREND_ALPHA = 1,
POW_TREND_BETA = 1,
POW_SEASON_ALPHA = 1,
POW_SEASON_BETA = 1,
MIN_SIGMA = 1e-10,
MIN_VAL = 1e-30,
MAX_VAL = 1e+38
)

Arguments

ADAPT_DELTA Target Metropolis acceptance rate. See Stan manual. Suggested range is between (0.85-0.97).
MAX_TREE_DEPTH NUTS maximum tree depth. See Stan manual for more details. Suggested range is between (10-15), default is 12.
NUM_OF_CHAINS Number of MCMC chains. Suggested range is 3 to 4. Default is 4.
NUM_OF_CORES Number of cores used for calculations. It can be smaller than NUM_OF_CHAINS, but for best computational speed, it should be equal to NUM_OF_CHAINS. Default is 4.
ADD_JITTER Whether to add a very small amount (sd=min(y)*0.0001) of jitter to the input series. It is sometimes useful in cases of series with some perfectly flat sections. Default is TRUE.
CAUCHY_SD_DIV Cauchy distribution is used for some parameters with non-obvious range. The error size hyperparameter of this distribution is calculated by dividing the max value of the time series by this constant. Suggested range is between (100,300). Default 150.
NUM_OF_ITER Starting number of iterations for each chain. Suggested range is between (2000,10000). Default is 5000. Generally, the longer the series, the smaller is the value to reach convergence. Some models e.g. those with “innov” error size method are more difficult to fit and require more iterations.
MAX_NUM_OF_REPEATS Maximum number of the sampling procedure repeats if the fit is unsatisfactorily, i.e. avgRHat>MAX_RHAT_ALLOWED. Each round will double the number of iterations which could potentially double the running time. Suggested range is between (2,4). Default is 2.
MAX_RHAT_ALLOWED

Maximum average value of Rhat’s that suggests a good fit, i.e. the threshold below which the fit is considered as acceptable. Consult Stan’s manual for more details on Rhat. Suggested range is between (1.005,1.02). Default is 1.006.

NUM_OF_SEASON_INIT_CYCLES

For seasonal models, number of seasonality periods used for establishing initial seasonality coefficients. Default is 3.

MIN_NU

Minimum degrees of freedom of the Student’s distribution that is used in most models. Suggested range(1.2, 5). Default 2.

MAX_NU

Maximum degrees of freedom of the Student’s distribution. Suggested range is between (15,30). Default 20.

MIN_POW_TREND

Minimum value of the global trend power coefficient. Suggested range is between (-1.0). Default -.5

MAX_POW_TREND

Maximum value of the global trend power coefficient. It should be 1 to allow the model to approach exponential growth when needed. Default is 1.

POW_TREND_ALPHA

Alpha parameter of Beta prior distribution. To make the forecast more upward curved, so to nudge it towards larger values, make the parameter larger. Suggested range is between (1.6) Default 1.

POW_TREND_BETA

Beta parameter of Beta prior distribution for the global trend power coefficient. 1 by default, see also above.

POW_SEASON_ALPHA

Alpha parameter of Beta distribution that is the prior of the power coefficient in the formula of the generalized seasonality in gSGT model. 1 by default, increasing it (say, to 3 or 5) will push the seasonality towards multiplicative behavior.

POW_SEASON_BETA

Beta parameter of Beta distribution that is the prior of the power coefficient in the formula of the generalized seasonality in gSGT model. 1 by default.

MIN_SIGMA

Minimum size of the fitted sigma, applied for numerical stability. Must be positive. 1e-10 by default.

MIN_VAL

Minimum value that forecast can take. Must be positive. 1e-30 by default.

MAX_VAL

Maximum value the forecast can take. 1e38 by default.

Value

list of control parameters

---

rlgtfit  

rlgtfit class

---

Description

A constructor function for objects of class rlgtfit, the main class of the package. Objects of this class are output from the rlgt function. This constructor will usually not be called by users directly.
Usage

rlgtfit(
  y,
  model.type,
  use.regression,
  seasonalityMethodId,
  levelMethodId,
  useSmoothingMethodForError = FALSE,
  seasonality,
  seasonality2,
  rlgtmodel,
  params,
  control,
  samples
)

Arguments

y  time series data for training (provided as a vector or a ts object).
model.type  the type of rlgt model, one of: "LGT", "SGT", "S2GT"
use.regression  whether the data has any additional variables to be used with forecasting, i.e. multivariate time-series.
seasonalityMethodId  Seasonality method Id (0- HW, 1- generalized).
levelMethodId  Level method Id.
useSmoothingMethodForError  if the non-standard function for error size should be used, one based on smoothed innovations or surprises
seasonality  This specification of seasonality will be overridden by frequency of y, if y is of ts or msts class. 1 by default, i.e. no seasonality.
seasonality2  Second seasonality. If larger than 1, a dual seasonality model will be used. This specification of seasonality will be overridden by the second seasonality of y, if y is of msts class. 1 by default, i.e. no seasonality or single seasonality.
rlgtmodel  an rlgt model.
params  list of parameters of the model (to be fitted).
control  list of control parameters, i.e. hyperparameter values for the model’s prior distribution. See rlgt.control
samples  stanfit object representing the MCMC samples

Value

an rlgtfit instance
Description

A dataset containing monthly University of Michigan survey of Consumer Sentiment along with a few related Google trend queries from January 2014 to June 2018. This aims to mimic the dataset from Scott and Varian (2014).

Usage

```r
data("umcsent.example")
```

Format

A data frame with 174 rows and 8 variables with log-transformation

- `date`: first date of each month in US calendar format
- `consumer.sent`: monthly initial claims of University of Michigan: Consumer Sentiment
- `search.engine`: normalized trend queries retrieved from gtrendsR API
- `financial.planning`: normalized trend queries retrieved from gtrendsR API
- `bus.news`: normalized trend queries retrieved from gtrendsR API
- `investing`: normalized trend queries retrieved from gtrendsR API
- `energy.utilities`: normalized trend queries retrieved from gtrendsR API

References

University of Michigan, University of Michigan: Consumer Sentiment [UMCSENT], retrieved from FRED, Federal Reserve Bank of St. Louis; [https://fred.stlouisfed.org/series/UMCSENT](https://fred.stlouisfed.org/series/UMCSENT), November 17, 2018.


An interface for retrieving and displaying the information returned online by Google Trends is provided. Trends (number of hits) over the time as well as geographic representation of the results can be displayed. [https://CRAN.R-project.org/package=gtrendsR](https://CRAN.R-project.org/package=gtrendsR)

Index

* datasets
  iclaims.example, 6
  umcsent.example, 14

* exponential
  Rlgt-package, 2

* forecasting
  Rlgt-package, 2

* smoothing
  Rlgt-package, 2

forecast.rlgf, 5

iclaims.example, 6
initModel, 7

posterior_interval.rlgf, 7
print.rlgf, 8

Rlg (Rlgt-package), 2
rlgt, 9, 10, 12
Rlgt-package, 2
rlgt.control, 10, 13
rlgtfit, 5, 8, 10, 12

summary.rlg (print.rlgf), 8

umcsent.example, 14