Package ‘smooth’

June 17, 2020

Type Package

Title Forecasting Using State Space Models

Version 2.6.0

Date 2020-06-16

URL https://github.com/config-i1/smooth

BugReports https://github.com/config-i1/smooth/issues

Language en-GB

Description Functions implementing Single Source of Error state space models for purposes of time series analysis and forecasting.
The package includes Exponential Smoothing (Hyndman et al., 2008, <doi: 10.1007/978-3-540-71918-2>),
SARIMA (Svetunkov & Boylan, 2019 <doi: 10.1080/00207543.2019.1600764>),
Complex Exponential Smoothing (Svetunkov & Kourentzes, 2018, <doi: 10.13140/RG.2.2.24986.29123>),
Simple Moving Average (Svetunkov & Petropoulos, 2018 <doi: 10.1080/00207543.2017.1380326>),
Vector Exponential Smoothing (de Silva et al., 2010, <doi: 10.1177/1471082X0901000401>) in state space forms,
several simulation functions and intermittent demand state space models. It also allows dealing with
intermittent demand based on the iETS framework (Svetunkov & Boylan, 2017, <doi: 10.13140/RG.2.2.35897.06242>).

License GPL (>= 2)

Depends R (>= 3.0.2), greybox (>= 0.5.9)

Imports Rcpp (>= 0.12.3), stats, graphics, forecast (>= 7.0), nloptr, utils, zoo

LinkingTo Rcpp, RcppArmadillo (>= 0.8.100.0.0)

Suggests Mcomp, numDeriv, testthat, knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 7.1.0

Encoding UTF-8

NeedsCompilation yes

Author Ivan Svetunkov [aut, cre] (Lecturer at Centre for Marketing Analytics and Forecasting, Lancaster University, UK)

Maintainer Ivan Svetunkov <ivan@svetunkov.ru>

Repository CRAN

Date/Publication 2020-06-17 04:10:03 UTC

R topics documented:

auto.ces *Complex Exponential Smoothing Auto*

Description

Function estimates CES in state space form with information potential equal to errors with different seasonality types and chooses the one with the lowest IC value.

Usage

```
auto.ces(y, models = c("none", "simple", "full"),
  initial = c("backcasting", "optimal"), ic = c("AICc", "AIC", "BIC",
  "BICc"), loss = c("MSE", "MAE", "HAM", "MSEh", "TMSE", "GTMSE", "MSCE"),
  h = 10, holdout = FALSE, cumulative = FALSE, interval = c("none",
  "parametric", "likelihood", "semiparametric", "nonparametric"),
  level = 0.95, occurrence = c("none", "auto", "fixed", "general",
  "odds-ratio", "inverse-odds-ratio", "direct"), oesmodel = "MNN",
  bounds = c("admissible", "none"), silent = c("all", "graph", "legend",
  "output", "none"), xreg = NULL, xregDo = c("use", "select"),
  initialX = NULL, updateX = FALSE, persistenceX = NULL,
  transitionX = NULL, ...)
```

Arguments

y	Vector or ts object, containing data needed to be forecasted.
models	The vector containing several types of seasonality that should be used in CES selection. See ces for more details about the possible types of seasonal models.
initial	Can be either character or a vector of initial states. If it is character, then it can be "optimal", meaning that the initial states are optimised, or "backcasting", meaning that the initials are produced using backcasting procedure.
ic	The information criterion used in the model selection procedure.
loss	The type of Loss Function used in optimization. loss can be: MSE (Mean Squared Error), MAE (Mean Absolute Error), HAM (Half Absolute Moment), TMSE - Trace Mean Squared Error, GTMSE - Geometric Trace Mean Squared Error, MSEh - optimisation using only h-steps ahead error, MSCE - Mean Squared Cumulative Error. If loss!="MSE", then likelihood and model selection is done based on equivalent MSE. Model selection in this cases becomes not optimal. There are also available analytical approximations for multistep functions: aMSEh, aTMSE and aGTMSE. These can be useful in cases of small samples. Finally, just for fun the absolute and half analogues of multistep estimators are available: MAEh, TMAE, GTMAE, MACE, TMAE, HAMh, THAM, GTHAM, CHAM.
h	Length of forecasting horizon.
holdout	If TRUE, holdout sample of size h is taken from the end of the data.
cumulative	If TRUE, then the cumulative forecast and prediction interval are produced instead of the normal ones. This is useful for inventory control systems.

`interval` Type of interval to construct. This can be:

- `"none"`, aka `"n"` - do not produce prediction interval.
- `"parametric"`, `"p"` - use state-space structure of ETS. In case of mixed models this is done using simulations, which may take longer time than for the pure additive and pure multiplicative models. This type of interval relies on unbiased estimate of in-sample error variance, which divides the sume of squared errors by T-k rather than just T.
- `"likelihood"`, `"l"` - these are the same as `"p"`, but relies on the biased estimate of variance from the likelihood (division by T, not by T-k).
- `"semiparametric"`, `"sp"` - interval based on covariance matrix of 1 to h steps ahead errors and assumption of normal / log-normal distribution (depending on error type).
- `"nonparametric"`, `"np"` - interval based on values from a quantile regression on error matrix (see Taylor and Bunn, 1999). The model used in this process is e[j] = a j^b, where j=1,..,h.

The parameter also accepts `TRUE` and `FALSE`. The former means that parametric interval are constructed, while the latter is equivalent to `none`. If the forecasts of the models were combined, then the interval are combined quantile-wise (Lichtendahl et al., 2013).

`level` Confidence level. Defines width of prediction interval.

`occurrence` The type of model used in probability estimation. Can be `"none"` - none, `"fixed"` - constant probability, `"general"` - the general Beta model with two parameters, `"odds-ratio"` - the Odds-ratio model with b=1 in Beta distribution, `"inverse-odds-ratio"` - the model with a=1 in Beta distribution, `"direct"` - the TSB-like (Teunter et al., 2011) probability update mechanism a+b=1, `"auto"` - the automatically selected type of occurrence model.

`oesmodel` The type of ETS model used for the modelling of the time varying probability. Object of the class "oes" can be provided here, and its parameters would be used in iETS model.

`bounds` What type of bounds to use in the model estimation. The first letter can be used instead of the whole word.

`silent` If `silent="none"`, then nothing is silent, everything is printed out and drawn. `silent="all"` means that nothing is produced or drawn (except for warnings). In case of `silent="graph"`, no graph is produced. If `silent="legend"`, then legend of the graph is skipped. And finally `silent="output"` means that nothing is printed out in the console, but the graph is produced. `silent` also accepts `TRUE` and `FALSE`. In this case `silent=TRUE` is equivalent to `silent="all"`, while `silent=FALSE` is equivalent to `silent="none"`. The parameter also accepts first letter of words ("n", "a", "g", "l", "o").

`xreg` The vector (either numeric or time series) or the matrix (or data.frame) of exogenous variables that should be included in the model. If matrix included than columns should contain variables and rows - observations. Note that `xreg` should have number of observations equal either to in-sample or to the whole series. If the number of observations in `xreg` is equal to in-sample, then values for the holdout sample are produced using es function.

`xregDo` The variable defines what to do with the provided xreg: `"use"` means that all of the data should be used, while `"select"` means that a selection using `ic` should be done. `"combine"` will be available at some point in future...

`initialX` The vector of initial parameters for exogenous variables. Ignored if `xreg` is NULL.

`updateX` If `TRUE`, transition matrix for exogenous variables is estimated, introducing non-linear interactions between parameters. Prerequisite - non-NULL `xreg`.

`persistenceX` The persistence vector g_X, containing smoothing parameters for exogenous variables. If `NULL`, then estimated. Prerequisite - non-NULL `xreg`.

`transitionX` The transition matrix F_x for exogenous variables. Can be provided as a vector. Matrix will be formed using the default `matrix(transition,nc,nc)`, where `nc` is number of components in state vector. If `NULL`, then estimated. Prerequisite - non-NULL `xreg`.

`...` Other non-documented parameters. For example `FI=TRUE` will make the function produce Fisher Information matrix, which then can be used to calculated variances of parameters of the model.

Details

The function estimates several Complex Exponential Smoothing in the state space 2 described in Svetunkov, Kourentzes (2015) with the information potential equal to the approximation error using different types of seasonality and chooses the one with the lowest value of information criterion.

For some more information about the model and its implementation, see the vignette: `vignette("ces","smooth")`

Value

Object of class "smooth" is returned. See ces for details.

Author(s)

Ivan Svetunkov, `<ivan@svetunkov.ru>`

References

- Svetunkov, I., Kourentzes, N. (February 2015). Complex exponential smoothing. Working Paper of Department of Management Science, Lancaster University 2015:1, 1-31.
- Svetunkov I., Kourentzes N. (2017) Complex Exponential Smoothing for Time Series Forecasting. Not yet published.

See Also

`ces`, `ets`, `forecast`, `ts`

Examples

```
y <- ts(rnorm(100,10,3),frequency=12)
# CES with and without holdout
auto.ces(y,h=20,holdout=TRUE)
auto.ces(y,h=20,holdout=FALSE)

library("Mcomp")
## Not run: y <- ts(c(M3$N0740$x,M3$N0740$xx),start=start(M3$N0740$x),frequency=frequency(M3$N0740$x))
# Selection between "none" and "full" seasonalities
auto.ces(y,h=8,holdout=TRUE,models=c("n","f"),interval="p",level=0.8,ic="AIC")
## End(Not run)

ourModel <- auto.ces(M3[[1683]],interval="sp")

summary(ourModel)
forecast(ourModel)
plot(forecast(ourModel))
```

---

auto.gum *Automatic GUM*

Description

Function selects the order of GUM model based on information criteria, using fancy branch and bound mechanism.

Usage

```
auto.gum(y, orders = 3, lags = frequency(y), type = c("additive",
  "multiplicative", "select"), initial = c("backcasting", "optimal"),
  ic = c("AICc", "AIC", "BIC", "BICc"), loss = c("MSE", "MAE", "HAM",
  "MSEh", "TMSE", "GTMSE", "MSCE"), h = 10, holdout = FALSE,
  cumulative = FALSE, interval = c("none", "parametric", "likelihood",
  "semiparametric", "nonparametric"), level = 0.95, occurrence = c("none",
  "auto", "fixed", "general", "odds-ratio", "inverse-odds-ratio", "direct"),
  oesmodel = "MNN", bounds = c("restricted", "admissible", "none"),
  silent = c("all", "graph", "legend", "output", "none"), xreg = NULL,
  xregDo = c("use", "select"), initialX = NULL, updateX = FALSE,
  persistenceX = NULL, transitionX = NULL, ...)
```

Arguments

`y`	Vector or ts object, containing data needed to be forecasted.
`orders`	The value of the max order to check. This is the upper bound of orders, but the real orders could be lower than this because of the increasing number of parameters in the models with higher orders.

lags — The value of the maximum lag to check. This should usually be a maximum frequency of the data.

type — Type of model. Can either be "additive" or "multiplicative". The latter means that the GUM is fitted on log-transformed data. If "select", then this is selected automatically, which may slow down things twice.

initial — Can be either character or a vector of initial states. If it is character, then it can be "optimal", meaning that the initial states are optimised, or "backcasting", meaning that the initials are produced using backcasting procedure.

ic — The information criterion used in the model selection procedure.

loss — The type of Loss Function used in optimization. loss can be: MSE (Mean Squared Error), MAE (Mean Absolute Error), HAM (Half Absolute Moment), TMSE - Trace Mean Squared Error, GTMSE - Geometric Trace Mean Squared Error, MSEh - optimisation using only h-steps ahead error, MSCE - Mean Squared Cumulative Error. If loss!="MSE", then likelihood and model selection is done based on equivalent MSE. Model selection in this cases becomes not optimal.

There are also available analytical approximations for multistep functions: aMSEh, aTMSE and aGTMSE. These can be useful in cases of small samples.

Finally, just for fun the absolute and half analogues of multistep estimators are available: MAEh, TMAE, GTMAE, MACE, TMAE, HAMh, THAM, GTHAM, CHAM.

h — Length of forecasting horizon.

holdout — If TRUE, holdout sample of size h is taken from the end of the data.

cumulative — If TRUE, then the cumulative forecast and prediction interval are produced instead of the normal ones. This is useful for inventory control systems.

interval — Type of interval to construct. This can be:

- "none", aka "n" - do not produce prediction interval.
- "parametric", "p" - use state-space structure of ETS. In case of mixed models this is done using simulations, which may take longer time than for the pure additive and pure multiplicative models. This type of interval relies on unbiased estimate of in-sample error variance, which divides the sume of squared errors by T-k rather than just T.
- "likelihood", "l" - these are the same as "p", but relies on the biased estimate of variance from the likelihood (division by T, not by T-k).
- "semiparametric", "sp" - interval based on covariance matrix of 1 to h steps ahead errors and assumption of normal / log-normal distribution (depending on error type).
- "nonparametric", "np" - interval based on values from a quantile regression on error matrix (see Taylor and Bunn, 1999). The model used in this process is e[j] = a j^b, where j=1,..,h.

The parameter also accepts TRUE and FALSE. The former means that parametric interval are constructed, while the latter is equivalent to none. If the forecasts of the models were combined, then the interval are combined quantile-wise (Lichtendahl et al., 2013).

level — Confidence level. Defines width of prediction interval.

occurrence
: The type of model used in probability estimation. Can be "none" - none, "fixed" - constant probability, "general" - the general Beta model with two parameters, "odds-ratio" - the Odds-ratio model with b=1 in Beta distribution, "inverse-odds-ratio" - the model with a=1 in Beta distribution, "direct" - the TSB-like (Teunter et al., 2011) probability update mechanism a+b=1, "auto" - the automatically selected type of occurrence model.

oesmodel
: The type of ETS model used for the modelling of the time varying probability. Object of the class "oes" can be provided here, and its parameters would be used in iETS model.

bounds
: What type of bounds to use in the model estimation. The first letter can be used instead of the whole word.

silent
: If silent="none", then nothing is silent, everything is printed out and drawn. silent="all" means that nothing is produced or drawn (except for warnings). In case of silent="graph", no graph is produced. If silent="legend", then legend of the graph is skipped. And finally silent="output" means that nothing is printed out in the console, but the graph is produced. silent also accepts TRUE and FALSE. In this case silent=TRUE is equivalent to silent="all", while silent=FALSE is equivalent to silent="none". The parameter also accepts first letter of words ("n", "a", "g", "l", "o").

xreg
: The vector (either numeric or time series) or the matrix (or data.frame) of exogenous variables that should be included in the model. If matrix included than columns should contain variables and rows - observations. Note that xreg should have number of observations equal either to in-sample or to the whole series. If the number of observations in xreg is equal to in-sample, then values for the holdout sample are produced using es function.

xregDo
: The variable defines what to do with the provided xreg: "use" means that all of the data should be used, while "select" means that a selection using ic should be done. "combine" will be available at some point in future...

initialX
: The vector of initial parameters for exogenous variables. Ignored if xreg is NULL.

updateX
: If TRUE, transition matrix for exogenous variables is estimated, introducing non-linear interactions between parameters. Prerequisite - non-NULL xreg.

persistenceX
: The persistence vector g_X, containing smoothing parameters for exogenous variables. If NULL, then estimated. Prerequisite - non-NULL xreg.

transitionX
: The transition matrix F_x for exogenous variables. Can be provided as a vector. Matrix will be formed using the default matrix(transition,nc,nc), where nc is number of components in state vector. If NULL, then estimated. Prerequisite - non-NULL xreg.

...
: Other non-documented parameters. For example FI=TRUE will make the function also produce Fisher Information matrix, which then can be used to calculated variances of parameters of the model.

Details

The function checks several GUM models (see gum documentation) and selects the best one based on the specified information criterion.

The resulting model can be complicated and not straightforward, because GUM allows capturing hidden orders that no ARIMA model can. It is advised to use `initial="b"`, because optimising GUM of arbitrary order is not a simple task.

For some more information about the model and its implementation, see the vignette: `vignette("gum","smooth")`

Value

Object of class "smooth" is returned. See gum for details.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

References

- Snyder, R. D., 1985. Recursive Estimation of Dynamic Linear Models. Journal of the Royal Statistical Society, Series B (Methodological) 47 (2), 272-276.
- Hyndman, R.J., Koehler, A.B., Ord, J.K., and Snyder, R.D. (2008) Forecasting with exponential smoothing: the state space approach, Springer-Verlag. http://dx.doi.org/10.1007/978-3-540-71918-2.
- Svetunkov Ivan and Boylan John E. (2017). Multiplicative State-Space Models for Intermittent Time Series. Working Paper of Department of Management Science, Lancaster University, 2017:4 , 1-43.
- Teunter R., Syntetos A., Babai Z. (2011). Intermittent demand: Linking forecasting to inventory obsolescence. European Journal of Operational Research, 214, 606-615.
- Croston, J. (1972) Forecasting and stock control for intermittent demands. Operational Research Quarterly, 23(3), 289-303.
- Syntetos, A., Boylan J. (2005) The accuracy of intermittent demand estimates. International Journal of Forecasting, 21(2), 303-314.

See Also

`gum`, `ets`, `es`, `ces`, `sim.es`, `ssarima`

Examples

```
x <- rnorm(50,100,3)

# The best GUM model for the data
ourModel <- auto.gum(x,orders=2,lags=4,h=18,holdout=TRUE,interval="np")

summary(ourModel)
forecast(ourModel)
plot(forecast(ourModel))
```

auto.msarima *Automatic Multiple Seasonal ARIMA*

Description

Function selects the best State Space ARIMA based on information criteria, using fancy branch and bound mechanism. The resulting model can be not optimal in IC meaning, but it is usually reasonable. This mechanism is described in Svetunkov & Boylan (2019).

Usage

```
auto.msarima(y, orders = list(ar = c(3, 3), i = c(2, 1), ma = c(3, 3)),
  lags = c(1, frequency(y)), combine = FALSE, fast = TRUE,
  constant = NULL, initial = c("backcasting", "optimal"), ic = c("AICc",
  "AIC", "BIC", "BICc"), loss = c("MSE", "MAE", "HAM", "MSEh", "TMSE",
  "GTMSE", "MSCE"), h = 10, holdout = FALSE, cumulative = FALSE,
  interval = c("none", "parametric", "likelihood", "semiparametric",
  "nonparametric"), level = 0.95, occurrence = c("none", "auto", "fixed",
  "general", "odds-ratio", "inverse-odds-ratio", "direct"), oesmodel = "MNN",
  bounds = c("admissible", "none"), silent = c("all", "graph", "legend",
  "output", "none"), xreg = NULL, xregDo = c("use", "select"),
  initialX = NULL, updateX = FALSE, persistenceX = NULL,
  transitionX = NULL, ...)
```

Arguments

y	Vector or ts object, containing data needed to be forecasted.
orders	List of maximum orders to check, containing vector variables `ar`, `i` and `ma`. If a variable is not provided in the list, then it is assumed to be equal to zero. At least one variable should have the same length as `lags`.
lags	Defines lags for the corresponding orders (see examples). The length of `lags` must correspond to the length of `orders`. There is no restrictions on the length of `lags` vector.
combine	If `TRUE`, then resulting ARIMA is combined using AIC weights.
fast	If `TRUE`, then some of the orders of ARIMA are skipped. This is not advised for models with `lags` greater than 12.
constant	If `NULL`, then the function will check if constant is needed. if `TRUE`, then constant is forced in the model. Otherwise constant is not used.
initial	Can be either character or a vector of initial states. If it is character, then it can be `"optimal"`, meaning that the initial states are optimised, or `"backcasting"`, meaning that the initials are produced using backcasting procedure.
ic	The information criterion used in the model selection procedure.

`loss` — The type of Loss Function used in optimization. `loss` can be: `MSE` (Mean Squared Error), `MAE` (Mean Absolute Error), `HAM` (Half Absolute Moment), `TMSE` - Trace Mean Squared Error, `GTMSE` - Geometric Trace Mean Squared Error, `MSEh` - optimisation using only h-steps ahead error, `MSCE` - Mean Squared Cumulative Error. If `loss!="MSE"`, then likelihood and model selection is done based on equivalent `MSE`. Model selection in this cases becomes not optimal.

There are also available analytical approximations for multistep functions: `aMSEh`, `aTMSE` and `aGTMSE`. These can be useful in cases of small samples.

Finally, just for fun the absolute and half analogues of multistep estimators are available: `MAEh`, `TMAE`, `GTMAE`, `MACE`, `TMAE`, `HAMh`, `THAM`, `GTHAM`, `CHAM`.

`h` — Length of forecasting horizon.

`holdout` — If `TRUE`, holdout sample of size `h` is taken from the end of the data.

`cumulative` — If `TRUE`, then the cumulative forecast and prediction interval are produced instead of the normal ones. This is useful for inventory control systems.

`interval` — Type of interval to construct. This can be:

- `"none"`, aka `"n"` - do not produce prediction interval.
- `"parametric"`, `"p"` - use state-space structure of ETS. In case of mixed models this is done using simulations, which may take longer time than for the pure additive and pure multiplicative models. This type of interval relies on unbiased estimate of in-sample error variance, which divides the sume of squared errors by T-k rather than just T.
- `"likelihood"`, `"l"` - these are the same as `"p"`, but relies on the biased estimate of variance from the likelihood (division by T, not by T-k).
- `"semiparametric"`, `"sp"` - interval based on covariance matrix of 1 to h steps ahead errors and assumption of normal / log-normal distribution (depending on error type).
- `"nonparametric"`, `"np"` - interval based on values from a quantile regression on error matrix (see Taylor and Bunn, 1999). The model used in this process is e[j] = a j^b, where j=1,..,h.

The parameter also accepts `TRUE` and `FALSE`. The former means that parametric interval are constructed, while the latter is equivalent to `none`. If the forecasts of the models were combined, then the interval are combined quantile-wise (Lichtendahl et al., 2013).

`level` — Confidence level. Defines width of prediction interval.

`occurrence` — The type of model used in probability estimation. Can be `"none"` - none, `"fixed"` - constant probability, `"general"` - the general Beta model with two parameters, `"odds-ratio"` - the Odds-ratio model with b=1 in Beta distribution, `"inverse-odds-ratio"` - the model with a=1 in Beta distribution, `"direct"` - the TSB-like (Teunter et al., 2011) probability update mechanism a+b=1, `"auto"` - the automatically selected type of occurrence model.

`oesmodel` — The type of ETS model used for the modelling of the time varying probability. Object of the class "oes" can be provided here, and its parameters would be used in iETS model.

`bounds` — What type of bounds to use in the model estimation. The first letter can be used instead of the whole word.

silent
: If `silent="none"`, then nothing is silent, everything is printed out and drawn. `silent="all"` means that nothing is produced or drawn (except for warnings). In case of `silent="graph"`, no graph is produced. If `silent="legend"`, then legend of the graph is skipped. And finally `silent="output"` means that nothing is printed out in the console, but the graph is produced. `silent` also accepts `TRUE` and `FALSE`. In this case `silent=TRUE` is equivalent to `silent="all"`, while `silent=FALSE` is equivalent to `silent="none"`. The parameter also accepts first letter of words ("n", "a", "g", "l", "o").

xreg
: The vector (either numeric or time series) or the matrix (or data.frame) of exogenous variables that should be included in the model. If matrix included than columns should contain variables and rows - observations. Note that `xreg` should have number of observations equal either to in-sample or to the whole series. If the number of observations in `xreg` is equal to in-sample, then values for the holdout sample are produced using es function.

xregDo
: The variable defines what to do with the provided xreg: `"use"` means that all of the data should be used, while `"select"` means that a selection using `ic` should be done. `"combine"` will be available at some point in future...

initialX
: The vector of initial parameters for exogenous variables. Ignored if `xreg` is NULL.

updateX
: If `TRUE`, transition matrix for exogenous variables is estimated, introducing non-linear interactions between parameters. Prerequisite - non-NULL `xreg`.

persistenceX
: The persistence vector g_X, containing smoothing parameters for exogenous variables. If `NULL`, then estimated. Prerequisite - non-NULL `xreg`.

transitionX
: The transition matrix F_x for exogenous variables. Can be provided as a vector. Matrix will be formed using the default `matrix(transition,nc,nc)`, where `nc` is number of components in state vector. If `NULL`, then estimated. Prerequisite - non-NULL `xreg`.

...
: Other non-documented parameters. For example `FI=TRUE` will make the function also produce Fisher Information matrix, which then can be used to calculated variances of parameters of the model. Maximum orders to check can also be specified separately, however `orders` variable must be set to `NULL`: `ar.orders` - Maximum order of AR term. Can be vector, defining max orders of AR, SAR etc. `i.orders` - Maximum order of I. Can be vector, defining max orders of I, SI etc. `ma.orders` - Maximum order of MA term. Can be vector, defining max orders of MA, SMA etc.

Details

The function constructs bunch of ARIMAs in Single Source of Error state space form (see msarima documentation) and selects the best one based on information criterion. It works faster than auto.ssarima on large datasets and high frequency data.

Due to the flexibility of the model, multiple seasonalities can be used. For example, something crazy like this can be constructed: SARIMA(1,1,1)(0,1,1)[24](2,0,1)[24*7](0,0,1)[24*30], but the estimation may take some time...

For some more information about the model and its implementation, see the vignette: `vignette("ssarima","smooth")`

Value

Object of class "smooth" is returned. See msarima for details.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

References

- Snyder, R. D., 1985. Recursive Estimation of Dynamic Linear Models. Journal of the Royal Statistical Society, Series B (Methodological) 47 (2), 272-276.
- Hyndman, R.J., Koehler, A.B., Ord, J.K., and Snyder, R.D. (2008) Forecasting with exponential smoothing: the state space approach, Springer-Verlag. http://dx.doi.org/10.1007/978-3-540-71918-2.
- Svetunkov Ivan and Boylan John E. (2017). Multiplicative State-Space Models for Intermittent Time Series. Working Paper of Department of Management Science, Lancaster University, 2017:4 , 1-43.
- Teunter R., Syntetos A., Babai Z. (2011). Intermittent demand: Linking forecasting to inventory obsolescence. European Journal of Operational Research, 214, 606-615.
- Croston, J. (1972) Forecasting and stock control for intermittent demands. Operational Research Quarterly, 23(3), 289-303.
- Syntetos, A., Boylan J. (2005) The accuracy of intermittent demand estimates. International Journal of Forecasting, 21(2), 303-314.
- Svetunkov, I., & Boylan, J. E. (2019). State-space ARIMA for supply-chain forecasting. International Journal of Production Research, 0(0), 1–10. https://doi.org/10.1080/00207543.2019.1600764

See Also

ets, es, ces, sim.es, gum, msarima

Examples

```
x <- rnorm(118,100,3)

# The best ARIMA for the data
ourModel <- auto.msarima(x,orders=list(ar=c(2,1),i=c(1,1),ma=c(2,1)),lags=c(1,12),
                     h=18,holdout=TRUE,interval="np")

# The other one using optimised states
## Not run: auto.msarima(x,orders=list(ar=c(3,2),i=c(2,1),ma=c(3,2)),lags=c(1,12),
                     initial="o",h=18,holdout=TRUE)
## End(Not run)

# And now combined ARIMA
## Not run: auto.msarima(x,orders=list(ar=c(3,2),i=c(2,1),ma=c(3,2)),lags=c(1,12),
```

```
                        combine=TRUE,h=18,holdout=TRUE)
## End(Not run)

summary(ourModel)
forecast(ourModel)
plot(forecast(ourModel))
```

`auto.ssarima` *State Space ARIMA*

Description

Function selects the best State Space ARIMA based on information criteria, using fancy branch and bound mechanism. The resulting model can be not optimal in IC meaning, but it is usually reasonable.

Usage

```
auto.ssarima(y, orders = list(ar = c(3, 3), i = c(2, 1), ma = c(3, 3)),
  lags = c(1, frequency(y)), combine = FALSE, fast = TRUE,
  constant = NULL, initial = c("backcasting", "optimal"), ic = c("AICc",
  "AIC", "BIC", "BICc"), loss = c("MSE", "MAE", "HAM", "MSEh", "TMSE",
  "GTMSE", "MSCE"), h = 10, holdout = FALSE, cumulative = FALSE,
  interval = c("none", "parametric", "likelihood", "semiparametric",
  "nonparametric"), level = 0.95, occurrence = c("none", "auto", "fixed",
  "general", "odds-ratio", "inverse-odds-ratio", "direct"), oesmodel = "MNN",
  bounds = c("admissible", "none"), silent = c("all", "graph", "legend",
  "output", "none"), xreg = NULL, xregDo = c("use", "select"),
  initialX = NULL, updateX = FALSE, persistenceX = NULL,
  transitionX = NULL, ...)
```

Arguments

`y`	Vector or ts object, containing data needed to be forecasted.
`orders`	List of maximum orders to check, containing vector variables `ar`, `i` and `ma`. If a variable is not provided in the list, then it is assumed to be equal to zero. At least one variable should have the same length as `lags`.
`lags`	Defines lags for the corresponding orders (see examples). The length of `lags` must correspond to the length of `orders`. There is no restrictions on the length of `lags` vector.
`combine`	If `TRUE`, then resulting ARIMA is combined using AIC weights.
`fast`	If `TRUE`, then some of the orders of ARIMA are skipped. This is not advised for models with `lags` greater than 12.
`constant`	If `NULL`, then the function will check if constant is needed. if `TRUE`, then constant is forced in the model. Otherwise constant is not used.

initial
: Can be either character or a vector of initial states. If it is character, then it can be "optimal", meaning that the initial states are optimised, or "backcasting", meaning that the initials are produced using backcasting procedure.

ic
: The information criterion used in the model selection procedure.

loss
: The type of Loss Function used in optimization. loss can be: MSE (Mean Squared Error), MAE (Mean Absolute Error), HAM (Half Absolute Moment), TMSE - Trace Mean Squared Error, GTMSE - Geometric Trace Mean Squared Error, MSEh - optimisation using only h-steps ahead error, MSCE - Mean Squared Cumulative Error. If loss!="MSE", then likelihood and model selection is done based on equivalent MSE. Model selection in this cases becomes not optimal.

  There are also available analytical approximations for multistep functions: aMSEh, aTMSE and aGTMSE. These can be useful in cases of small samples.

  Finally, just for fun the absolute and half analogues of multistep estimators are available: MAEh, TMAE, GTMAE, MACE, TMAE, HAMh, THAM, GTHAM, CHAM.

h
: Length of forecasting horizon.

holdout
: If TRUE, holdout sample of size h is taken from the end of the data.

cumulative
: If TRUE, then the cumulative forecast and prediction interval are produced instead of the normal ones. This is useful for inventory control systems.

interval
: Type of interval to construct. This can be:

  - "none", aka "n" - do not produce prediction interval.
  - "parametric", "p" - use state-space structure of ETS. In case of mixed models this is done using simulations, which may take longer time than for the pure additive and pure multiplicative models. This type of interval relies on unbiased estimate of in-sample error variance, which divides the sume of squared errors by T-k rather than just T.
  - "likelihood", "l" - these are the same as "p", but relies on the biased estimate of variance from the likelihood (division by T, not by T-k).
  - "semiparametric", "sp" - interval based on covariance matrix of 1 to h steps ahead errors and assumption of normal / log-normal distribution (depending on error type).
  - "nonparametric", "np" - interval based on values from a quantile regression on error matrix (see Taylor and Bunn, 1999). The model used in this process is e[j] = a j^b, where j=1,..,h.

  The parameter also accepts TRUE and FALSE. The former means that parametric interval are constructed, while the latter is equivalent to none. If the forecasts of the models were combined, then the interval are combined quantile-wise (Lichtendahl et al., 2013).

level
: Confidence level. Defines width of prediction interval.

occurrence
: The type of model used in probability estimation. Can be "none" - none, "fixed" - constant probability, "general" - the general Beta model with two parameters, "odds-ratio" - the Odds-ratio model with b=1 in Beta distribution, "inverse-odds-ratio" - the model with a=1 in Beta distribution, "direct" - the TSB-like (Teunter et al., 2011) probability update mechanism a+b=1, "auto" - the automatically selected type of occurrence model.

oesmodel
: The type of ETS model used for the modelling of the time varying probability. Object of the class "oes" can be provided here, and its parameters would be used in iETS model.

bounds
: What type of bounds to use in the model estimation. The first letter can be used instead of the whole word.

silent
: If `silent="none"`, then nothing is silent, everything is printed out and drawn. `silent="all"` means that nothing is produced or drawn (except for warnings). In case of `silent="graph"`, no graph is produced. If `silent="legend"`, then legend of the graph is skipped. And finally `silent="output"` means that nothing is printed out in the console, but the graph is produced. `silent` also accepts `TRUE` and `FALSE`. In this case `silent=TRUE` is equivalent to `silent="all"`, while `silent=FALSE` is equivalent to `silent="none"`. The parameter also accepts first letter of words ("n", "a", "g", "l", "o").

xreg
: The vector (either numeric or time series) or the matrix (or data.frame) of exogenous variables that should be included in the model. If matrix included than columns should contain variables and rows - observations. Note that `xreg` should have number of observations equal either to in-sample or to the whole series. If the number of observations in `xreg` is equal to in-sample, then values for the holdout sample are produced using es function.

xregDo
: The variable defines what to do with the provided xreg: `"use"` means that all of the data should be used, while `"select"` means that a selection using `ic` should be done. `"combine"` will be available at some point in future...

initialX
: The vector of initial parameters for exogenous variables. Ignored if `xreg` is NULL.

updateX
: If `TRUE`, transition matrix for exogenous variables is estimated, introducing non-linear interactions between parameters. Prerequisite - non-NULL `xreg`.

persistenceX
: The persistence vector g_X, containing smoothing parameters for exogenous variables. If `NULL`, then estimated. Prerequisite - non-NULL `xreg`.

transitionX
: The transition matrix F_x for exogenous variables. Can be provided as a vector. Matrix will be formed using the default `matrix(transition,nc,nc)`, where `nc` is number of components in state vector. If `NULL`, then estimated. Prerequisite - non-NULL `xreg`.

...
: Other non-documented parameters. For example `FI=TRUE` will make the function also produce Fisher Information matrix, which then can be used to calculated variances of parameters of the model. Maximum orders to check can also be specified separately, however `orders` variable must be set to `NULL`: `ar.orders` - Maximum order of AR term. Can be vector, defining max orders of AR, SAR etc. `i.orders` - Maximum order of I. Can be vector, defining max orders of I, SI etc. `ma.orders` - Maximum order of MA term. Can be vector, defining max orders of MA, SMA etc.

Details

The function constructs bunch of ARIMAs in Single Source of Error state space form (see ssarima documentation) and selects the best one based on information criterion. The mechanism is described in Svetunkov & Boylan (2019).

Due to the flexibility of the model, multiple seasonalities can be used. For example, something crazy like this can be constructed: SARIMA(1,1,1)(0,1,1)[24](2,0,1)[24*7](0,0,1)[24*30], but the estimation may take a lot of time... It is recommended to use auto.msarima in cases with more than one seasonality and high frequencies.

For some more information about the model and its implementation, see the vignette: `vignette("ssarima","smooth")`

Value

Object of class "smooth" is returned. See ssarima for details.

Author(s)

Ivan Svetunkov, `<ivan@svetunkov.ru>`

References

- Snyder, R. D., 1985. Recursive Estimation of Dynamic Linear Models. Journal of the Royal Statistical Society, Series B (Methodological) 47 (2), 272-276.
- Hyndman, R.J., Koehler, A.B., Ord, J.K., and Snyder, R.D. (2008) Forecasting with exponential smoothing: the state space approach, Springer-Verlag. `http://dx.doi.org/10.1007/978-3-540-71918-2`.
- Svetunkov Ivan and Boylan John E. (2017). Multiplicative State-Space Models for Intermittent Time Series. Working Paper of Department of Management Science, Lancaster University, 2017:4 , 1-43.
- Teunter R., Syntetos A., Babai Z. (2011). Intermittent demand: Linking forecasting to inventory obsolescence. European Journal of Operational Research, 214, 606-615.
- Croston, J. (1972) Forecasting and stock control for intermittent demands. Operational Research Quarterly, 23(3), 289-303.
- Syntetos, A., Boylan J. (2005) The accuracy of intermittent demand estimates. International Journal of Forecasting, 21(2), 303-314.
- Svetunkov, I., & Boylan, J. E. (2019). State-space ARIMA for supply-chain forecasting. International Journal of Production Research, 0(0), 1–10. `https://doi.org/10.1080/00207543.2019.1600764`

See Also

`ets`, `es`, `ces`, `sim.es`, `gum`, `ssarima`

Examples

```
x <- rnorm(118,100,3)

# The best ARIMA for the data
ourModel <- auto.ssarima(x,orders=list(ar=c(2,1),i=c(1,1),ma=c(2,1)),lags=c(1,12),
                     h=18,holdout=TRUE,interval="np")
```

```
# The other one using optimised states
## Not run: auto.ssarima(x,orders=list(ar=c(3,2),i=c(2,1),ma=c(3,2)),lags=c(1,12),
                     initial="o",h=18,holdout=TRUE)
## End(Not run)

# And now combined ARIMA
## Not run: auto.ssarima(x,orders=list(ar=c(3,2),i=c(2,1),ma=c(3,2)),lags=c(1,12),
                      combine=TRUE,h=18,holdout=TRUE)
## End(Not run)

summary(ourModel)
forecast(ourModel)
plot(forecast(ourModel))
```

---

ces *Complex Exponential Smoothing*

Description

Function estimates CES in state space form with information potential equal to errors and returns several variables.

Usage

```
ces(y, seasonality = c("none", "simple", "partial", "full"),
  initial = c("backcasting", "optimal"), a = NULL, b = NULL,
  ic = c("AICc", "AIC", "BIC", "BICc"), loss = c("MSE", "MAE", "HAM",
  "MSEh", "TMSE", "GTMSE", "MSCE"), h = 10, holdout = FALSE,
  cumulative = FALSE, interval = c("none", "parametric", "likelihood",
  "semiparametric", "nonparametric"), level = 0.95, occurrence = c("none",
  "auto", "fixed", "general", "odds-ratio", "inverse-odds-ratio", "direct"),
  oesmodel = "MNN", bounds = c("admissible", "none"), silent = c("all",
  "graph", "legend", "output", "none"), xreg = NULL, xregDo = c("use",
  "select"), initialX = NULL, updateX = FALSE, persistenceX = NULL,
  transitionX = NULL, ...)
```

Arguments

`y` Vector or ts object, containing data needed to be forecasted.

`seasonality` The type of seasonality used in CES. Can be: none - No seasonality; `simple` - Simple seasonality, using lagged CES (based on `t-m` observation, where `m` is the seasonality lag); `partial` - Partial seasonality with real seasonal components (equivalent to additive seasonality); `full` - Full seasonality with complex seasonal components (can do both multiplicative and additive seasonality, depending on the data). First letter can be used instead of full words. Any seasonal CES can only be constructed for time series vectors.

initial
: Can be either character or a vector of initial states. If it is character, then it can be "optimal", meaning that the initial states are optimised, or "backcasting", meaning that the initials are produced using backcasting procedure.

a
: First complex smoothing parameter. Should be a complex number.

  NOTE! CES is very sensitive to a and b values so it is advised either to leave them alone, or to use values from previously estimated model.

b
: Second complex smoothing parameter. Can be real if seasonality="partial". In case of seasonality="full" must be complex number.

ic
: The information criterion used in the model selection procedure.

loss
: The type of Loss Function used in optimization. loss can be: MSE (Mean Squared Error), MAE (Mean Absolute Error), HAM (Half Absolute Moment), TMSE - Trace Mean Squared Error, GTMSE - Geometric Trace Mean Squared Error, MSEh - optimisation using only h-steps ahead error, MSCE - Mean Squared Cumulative Error. If loss!="MSE", then likelihood and model selection is done based on equivalent MSE. Model selection in this cases becomes not optimal.

  There are also available analytical approximations for multistep functions: aMSEh, aTMSE and aGTMSE. These can be useful in cases of small samples.

  Finally, just for fun the absolute and half analogues of multistep estimators are available: MAEh, TMAE, GTMAE, MACE, TMAE, HAMh, THAM, GTHAM, CHAM.

h
: Length of forecasting horizon.

holdout
: If TRUE, holdout sample of size h is taken from the end of the data.

cumulative
: If TRUE, then the cumulative forecast and prediction interval are produced instead of the normal ones. This is useful for inventory control systems.

interval
: Type of interval to construct. This can be:

  - "none", aka "n" - do not produce prediction interval.
  - "parametric", "p" - use state-space structure of ETS. In case of mixed models this is done using simulations, which may take longer time than for the pure additive and pure multiplicative models. This type of interval relies on unbiased estimate of in-sample error variance, which divides the sume of squared errors by T-k rather than just T.
  - "likelihood", "l" - these are the same as "p", but relies on the biased estimate of variance from the likelihood (division by T, not by T-k).
  - "semiparametric", "sp" - interval based on covariance matrix of 1 to h steps ahead errors and assumption of normal / log-normal distribution (depending on error type).
  - "nonparametric", "np" - interval based on values from a quantile regression on error matrix (see Taylor and Bunn, 1999). The model used in this process is e[j] = a j^b, where j=1,..,h.

  The parameter also accepts TRUE and FALSE. The former means that parametric interval are constructed, while the latter is equivalent to none. If the forecasts of the models were combined, then the interval are combined quantile-wise (Lichtendahl et al., 2013).

level
: Confidence level. Defines width of prediction interval.

occurrence
: The type of model used in probability estimation. Can be `"none"` - none, `"fixed"` - constant probability, `"general"` - the general Beta model with two parameters, `"odds-ratio"` - the Odds-ratio model with b=1 in Beta distribution, `"inverse-odds-ratio"` - the model with a=1 in Beta distribution, `"direct"` - the TSB-like (Teunter et al., 2011) probability update mechanism a+b=1, `"auto"` - the automatically selected type of occurrence model.

oesmodel
: The type of ETS model used for the modelling of the time varying probability. Object of the class "oes" can be provided here, and its parameters would be used in iETS model.

bounds
: What type of bounds to use in the model estimation. The first letter can be used instead of the whole word.

silent
: If `silent="none"`, then nothing is silent, everything is printed out and drawn. `silent="all"` means that nothing is produced or drawn (except for warnings). In case of `silent="graph"`, no graph is produced. If `silent="legend"`, then legend of the graph is skipped. And finally `silent="output"` means that nothing is printed out in the console, but the graph is produced. `silent` also accepts `TRUE` and `FALSE`. In this case `silent=TRUE` is equivalent to `silent="all"`, while `silent=FALSE` is equivalent to `silent="none"`. The parameter also accepts first letter of words ("n", "a", "g", "l", "o").

xreg
: The vector (either numeric or time series) or the matrix (or data.frame) of exogenous variables that should be included in the model. If matrix included than columns should contain variables and rows - observations. Note that `xreg` should have number of observations equal either to in-sample or to the whole series. If the number of observations in `xreg` is equal to in-sample, then values for the holdout sample are produced using es function.

xregDo
: The variable defines what to do with the provided xreg: `"use"` means that all of the data should be used, while `"select"` means that a selection using `ic` should be done. `"combine"` will be available at some point in future...

initialX
: The vector of initial parameters for exogenous variables. Ignored if `xreg` is NULL.

updateX
: If `TRUE`, transition matrix for exogenous variables is estimated, introducing nonlinear interactions between parameters. Prerequisite - non-NULL `xreg`.

persistenceX
: The persistence vector g_X, containing smoothing parameters for exogenous variables. If `NULL`, then estimated. Prerequisite - non-NULL `xreg`.

transitionX
: The transition matrix F_x for exogenous variables. Can be provided as a vector. Matrix will be formed using the default `matrix(transition,nc,nc)`, where `nc` is number of components in state vector. If `NULL`, then estimated. Prerequisite - non-NULL `xreg`.

...
: Other non-documented parameters. For example parameter `model` can accept a previously estimated CES model and use all its parameters. `FI=TRUE` will make the function produce Fisher Information matrix, which then can be used to calculated variances of parameters of the model.

Details

The function estimates Complex Exponential Smoothing in the state space 2 described in Svetunkov, Kourentzes (2017) with the information potential equal to the approximation error. The estimation of initial states of xt is done using backcast.

For some more information about the model and its implementation, see the vignette: `vignette("ces","smooth")`

Value

Object of class "smooth" is returned. It contains the list of the following values:

- `model` - type of constructed model.
- `timeElapsed` - time elapsed for the construction of the model.
- `states` - the matrix of the components of CES. The included minimum is "level" and "potential". In the case of seasonal model the seasonal component is also included. In the case of exogenous variables the estimated coefficients for the exogenous variables are also included.
- `a` - complex smoothing parameter in the form a0 + ia1
- `b` - smoothing parameter for the seasonal component. Can either be real (if `seasonality="P"`) or complex (if `seasonality="F"`) in a form b0 + ib1.
- `persistence` - persistence vector. This is the place, where smoothing parameters live.
- `transition` - transition matrix of the model.
- `measurement` - measurement vector of the model.
- `initialType` - Type of the initial values used.
- `initial` - the initial values of the state vector (non-seasonal).
- `nParam` - table with the number of estimated / provided parameters. If a previous model was reused, then its initials are reused and the number of provided parameters will take this into account.
- `fitted` - the fitted values of CES.
- `forecast` - the point forecast of CES.
- `lower` - the lower bound of prediction interval. When `interval="none"` then NA is returned.
- `upper` - the upper bound of prediction interval. When `interval="none"` then NA is returned.
- `residuals` - the residuals of the estimated model.
- `errors` - The matrix of 1 to h steps ahead errors.
- `s2` - variance of the residuals (taking degrees of freedom into account).
- `interval` - type of interval asked by user.
- `level` - confidence level for interval.
- `cumulative` - whether the produced forecast was cumulative or not.
- `y` - The data provided in the call of the function.
- `holdout` - the holdout part of the original data.
- `occurrence` - model of the class "oes" if the occurrence model was estimated. If the model is non-intermittent, then occurrence is `NULL`.
- `xreg` - provided vector or matrix of exogenous variables. If `xregDo="s"`, then this value will contain only selected exogenous variables.
- `updateX` - boolean, defining, if the states of exogenous variables were estimated as well.
- `initialX` - initial values for parameters of exogenous variables.

- persistenceX - persistence vector g for exogenous variables.
- transitionX - transition matrix F for exogenous variables.
- ICs - values of information criteria of the model. Includes AIC, AICc, BIC and BICc.
- logLik - log-likelihood of the function.
- lossValue - Cost function value.
- loss - Type of loss function used in the estimation.
- FI - Fisher Information. Equal to NULL if FI=FALSE or when FI is not provided at all.
- accuracy - vector of accuracy measures for the holdout sample. In case of non-intermittent data includes: MPE, MAPE, SMAPE, MASE, sMAE, RelMAE, sMSE and Bias coefficient (based on complex numbers). In case of intermittent data the set of errors will be: sMSE, sPIS, sCE (scaled cumulative error) and Bias coefficient. This is available only when holdout=TRUE.
- B - the vector of all the estimated parameters.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

References

- Svetunkov, I., Kourentzes, N. (February 2015). Complex exponential smoothing. Working Paper of Department of Management Science, Lancaster University 2015:1, 1-31.
- Svetunkov I., Kourentzes N. (2017) Complex Exponential Smoothing for Time Series Forecasting. Not yet published.

See Also

ets, forecast, ts, auto.ces

Examples

```
y <- rnorm(100,10,3)
ces(y,h=20,holdout=TRUE)
ces(y,h=20,holdout=FALSE)

y <- 500 - c(1:100)*0.5 + rnorm(100,10,3)
ces(y,h=20,holdout=TRUE,interval="p",bounds="a")

library("Mcomp")
y <- ts(c(M3$N0740$x,M3$N0740$xx),start=start(M3$N0740$x),frequency=frequency(M3$N0740$x))
ces(y,h=8,holdout=TRUE,seasonality="s",interval="sp",level=0.8)

## Not run: y <- ts(c(M3$N1683$x,M3$N1683$xx),start=start(M3$N1683$x),frequency=frequency(M3$N1683$x))
ces(y,h=18,holdout=TRUE,seasonality="s",interval="sp")
ces(y,h=18,holdout=TRUE,seasonality="p",interval="np")
ces(y,h=18,holdout=TRUE,seasonality="f",interval="p")
## End(Not run)
```

```
## Not run: x <- cbind(c(rep(0,25),1,rep(0,43)),c(rep(0,10),1,rep(0,58)))
ces(ts(c(M3$N1457$x,M3$N1457$xx),frequency=12),h=18,holdout=TRUE,
    interval="np",xreg=x,loss="TMSE")
## End(Not run)

# Exogenous variables in CES
## Not run: x <- cbind(c(rep(0,25),1,rep(0,43)),c(rep(0,10),1,rep(0,58)))
ces(ts(c(M3$N1457$x,M3$N1457$xx),frequency=12),h=18,holdout=TRUE,xreg=x)
ourModel <- ces(ts(c(M3$N1457$x,M3$N1457$xx),frequency=12),h=18,holdout=TRUE,xreg=x,updateX=TRUE)
# This will be the same model as in previous line but estimated on new portion of data
ces(ts(c(M3$N1457$x,M3$N1457$xx),frequency=12),model=ourModel,h=18,holdout=FALSE)
## End(Not run)

# Intermittent data example
x <- rpois(100,0.2)
# Best type of the occurrence model based on iETS(Z,Z,N)
ourModel <- ces(x,occurrence="auto")

summary(ourModel)
forecast(ourModel)
plot(forecast(ourModel))
```

---

cma *Centered Moving Average*

Description

Function constructs centered moving average based on state space SMA

Usage

```
cma(y, order = NULL, silent = TRUE, ...)
```

Arguments

y	Vector or ts object, containing data needed to be smoothed.
order	Order of centered moving average. If NULL, then the function will try to select order of SMA based on information criteria. See sma for details.
silent	If TRUE, then plot is not produced. Otherwise, there is a plot...
...	Nothing. Needed only for the transition to the new name of variables.

Details

If the order is odd, then the function constructs SMA(order) and shifts it back in time. Otherwise an AR(order+1) model is constructed with the preset parameters:

phi_i = 0.5,1,1,...,0.5 / order

This then corresponds to the centered MA with 0.5 weight for the first observation and 0.5 weight for an additional one. e.g. if this is monthly data and we use order=12, then half of the first january and half of the new one is taken.

This is not a forecasting tool. This is supposed to smooth the time series in order to find trend. So don't expect any forecasts from this function!

Value

Object of class "smooth" is returned. It contains the list of the following values:

- `model` - the name of the estimated model.
- `timeElapsed` - time elapsed for the construction of the model.
- `order` - order of the moving average.
- `nParam` - table with the number of estimated / provided parameters. If a previous model was reused, then its initials are reused and the number of provided parameters will take this into account.
- `fitted` - the fitted values, shifted in time.
- `forecast` - NAs, because this function does not produce forecasts.
- `residuals` - the residuals of the SMA / AR model.
- `s2` - variance of the residuals (taking degrees of freedom into account) of the SMA / AR model.
- `y` - the original data.
- `ICs` - values of information criteria from the respective SMA or AR model. Includes AIC, AICc, BIC and BICc.
- `logLik` - log-likelihood of the SMA / AR model.
- `lossValue` - Cost function value (for the SMA / AR model).
- `loss` - Type of loss function used in the estimation.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

References

- Svetunkov I. (2015 - Inf) "smooth" package for R - series of posts about the underlying models and how to use them: https://forecasting.svetunkov.ru/en/tag/smooth/.
- Svetunkov I. (2017). Statistical models underlying functions of 'smooth' package for R. Working Paper of Department of Management Science, Lancaster University 2017:1, 1-52.

See Also

`ma`, `es`, `ssarima`

Examples

```
# CMA of specific order
ourModel <- cma(rnorm(118,100,3),order=12)

# CMA of arbitrary order
ourModel <- cma(rnorm(118,100,3))

summary(ourModel)
```

---

es *Exponential Smoothing in SSOE state space model*

Description

Function constructs ETS model and returns forecast, fitted values, errors and matrix of states.

Usage

```
es(y, model = "ZZZ", persistence = NULL, phi = NULL,
  initial = c("optimal", "backcasting"), initialSeason = NULL,
  ic = c("AICc", "AIC", "BIC", "BICc"), loss = c("MSE", "MAE", "HAM",
  "MSEh", "TMSE", "GTMSE", "MSCE"), h = 10, holdout = FALSE,
  cumulative = FALSE, interval = c("none", "parametric", "likelihood",
  "semiparametric", "nonparametric"), level = 0.95, occurrence = c("none",
  "auto", "fixed", "general", "odds-ratio", "inverse-odds-ratio", "direct"),
  oesmodel = "MNN", bounds = c("usual", "admissible", "none"),
  silent = c("all", "graph", "legend", "output", "none"), xreg = NULL,
  xregDo = c("use", "select"), initialX = NULL, updateX = FALSE,
  persistenceX = NULL, transitionX = NULL, ...)
```

Arguments

`y` Vector or ts object, containing data needed to be forecasted.

`model` The type of ETS model. The first letter stands for the type of the error term ("A" or "M"), the second (and sometimes the third as well) is for the trend ("N", "A", "Ad", "M" or "Md"), and the last one is for the type of seasonality ("N", "A" or "M"). So, the function accepts words with 3 or 4 characters: `ANN`, `AAN`, `AAdN`, `AAA`, `AAdA`, `MAdM` etc. `ZZZ` means that the model will be selected based on the chosen information criteria type. Models pool can be restricted with additive only components. This is done via `model="XXX"`. For example, making selection between models with none / additive / damped additive trend component only (i.e. excluding multiplicative trend) can be done with `model="ZXZ"`. Furthermore, selection between multiplicative models (excluding additive components) is regulated using `model="YYY"`. This can be useful for positive data with low values (for example, slow moving products). Finally, if `model="CCC"`, then all the models are estimated and combination of their forecasts using AIC weights is produced (Kolassa, 2011). This can also be regulated. For example, `model="CCN"` will combine forecasts of all non-seasonal models and `model="CXY"` will combine forecasts of all the models with non-multiplicative trend and non-additive seasonality with either additive or multiplicative error. Not sure why anyone would need this thing, but it is available.

The parameter `model` can also be a vector of names of models for a finer tuning (pool of models). For example, `model=c("ANN","AAA")` will estimate only two models and select the best of them.

Also `model` can accept a previously estimated ES or ETS (from forecast package) model and use all its parameters.

Keep in mind that model selection with "Z" components uses Branch and Bound algorithm and may skip some models that could have slightly smaller information criteria.

`persistence` Persistence vector g, containing smoothing parameters. If `NULL`, then estimated.

`phi` Value of damping parameter. If `NULL` then it is estimated.

`initial` Can be either character or a vector of initial states. If it is character, then it can be `"optimal"`, meaning that the initial states are optimised, or `"backcasting"`, meaning that the initials are produced using backcasting procedure (advised for data with high frequency). If character, then `initialSeason` will be estimated in the way defined by `initial`.

`initialSeason` Vector of initial values for seasonal components. If `NULL`, they are estimated during optimisation.

`ic` The information criterion used in the model selection procedure.

`loss` The type of Loss Function used in optimization. `loss` can be: `MSE` (Mean Squared Error), `MAE` (Mean Absolute Error), `HAM` (Half Absolute Moment), `TMSE` - Trace Mean Squared Error, `GTMSE` - Geometric Trace Mean Squared Error, `MSEh` - optimisation using only h-steps ahead error, `MSCE` - Mean Squared Cumulative Error. If `loss!="MSE"`, then likelihood and model selection is done based on equivalent `MSE`. Model selection in this cases becomes not optimal.

There are also available analytical approximations for multistep functions: `aMSEh`, `aTMSE` and `aGTMSE`. These can be useful in cases of small samples.

Finally, just for fun the absolute and half analogues of multistep estimators are available: `MAEh`, `TMAE`, `GTMAE`, `MACE`, `TMAE`, `HAMh`, `THAM`, `GTHAM`, `CHAM`.

`h` Length of forecasting horizon.

`holdout` If `TRUE`, holdout sample of size `h` is taken from the end of the data.

`cumulative` If `TRUE`, then the cumulative forecast and prediction interval are produced instead of the normal ones. This is useful for inventory control systems.

`interval` Type of interval to construct. This can be:

- `"none"`, aka `"n"` - do not produce prediction interval.
- `"parametric"`, `"p"` - use state-space structure of ETS. In case of mixed models this is done using simulations, which may take longer time than for the pure additive and pure multiplicative models. This type of interval relies on unbiased estimate of in-sample error variance, which divides the sume of squared errors by T-k rather than just T.

- "likelihood", "l" - these are the same as "p", but relies on the biased estimate of variance from the likelihood (division by T, not by T-k).
- "semiparametric", "sp" - interval based on covariance matrix of 1 to h steps ahead errors and assumption of normal / log-normal distribution (depending on error type).
- "nonparametric", "np" - interval based on values from a quantile regression on error matrix (see Taylor and Bunn, 1999). The model used in this process is e[j] = a j^b, where j=1,..,h.

The parameter also accepts TRUE and FALSE. The former means that parametric interval are constructed, while the latter is equivalent to none. If the forecasts of the models were combined, then the interval are combined quantile-wise (Lichtendahl et al., 2013).

level
: Confidence level. Defines width of prediction interval.

occurrence
: The type of model used in probability estimation. Can be "none" - none, "fixed" - constant probability, "general" - the general Beta model with two parameters, "odds-ratio" - the Odds-ratio model with b=1 in Beta distribution, "inverse-odds-ratio" - the model with a=1 in Beta distribution, "direct" - the TSB-like (Teunter et al., 2011) probability update mechanism a+b=1, "auto" - the automatically selected type of occurrence model.

oesmodel
: The type of ETS model used for the modelling of the time varying probability. Object of the class "oes" can be provided here, and its parameters would be used in iETS model.

bounds
: What type of bounds to use in the model estimation. The first letter can be used instead of the whole word.

silent
: If silent="none", then nothing is silent, everything is printed out and drawn. silent="all" means that nothing is produced or drawn (except for warnings). In case of silent="graph", no graph is produced. If silent="legend", then legend of the graph is skipped. And finally silent="output" means that nothing is printed out in the console, but the graph is produced. silent also accepts TRUE and FALSE. In this case silent=TRUE is equivalent to silent="all", while silent=FALSE is equivalent to silent="none". The parameter also accepts first letter of words ("n", "a", "g", "l", "o").

xreg
: The vector (either numeric or time series) or the matrix (or data.frame) of exogenous variables that should be included in the model. If matrix included than columns should contain variables and rows - observations. Note that xreg should have number of observations equal either to in-sample or to the whole series. If the number of observations in xreg is equal to in-sample, then values for the holdout sample are produced using es function.

xregDo
: The variable defines what to do with the provided xreg: "use" means that all of the data should be used, while "select" means that a selection using ic should be done. "combine" will be available at some point in future...

initialX
: The vector of initial parameters for exogenous variables. Ignored if xreg is NULL.

updateX
: If TRUE, transition matrix for exogenous variables is estimated, introducing non-linear interactions between parameters. Prerequisite - non-NULL xreg.

persistenceX: The persistence vector g_X, containing smoothing parameters for exogenous variables. If NULL, then estimated. Prerequisite - non-NULL xreg.

transitionX: The transition matrix F_x for exogenous variables. Can be provided as a vector. Matrix will be formed using the default matrix(transition,nc,nc), where nc is number of components in state vector. If NULL, then estimated. Prerequisite - non-NULL xreg.

...: Other non-documented parameters. For example FI=TRUE will make the function also produce Fisher Information matrix, which then can be used to calculated variances of smoothing parameters and initial states of the model. Parameters B, lb and ub can be passed via ellipsis as well. In this case they will be used for optimisation. B sets the initial values before the optimisation, lb and ub define lower and upper bounds for the search inside of the specified bounds. These values should have length equal to the number of parameters to estimate. You can also pass two parameters to the optimiser: 1. maxeval - maximum number of evaluations to carry on; 2. xtol_rel - the precision of the optimiser. The default values used in es() are maxeval=500 and xtol_rel=1e-8. You can read more about these parameters in the documentation of nloptr function.

Details

Function estimates ETS in a form of the Single Source of Error state space model of the following type:

$$y_t = o_t(w(v_{t-l}) + h(x_t, a_{t-1}) + r(v_{t-l})\epsilon_t)$$

$$v_t = f(v_{t-l}) + g(v_{t-l})\epsilon_t$$

$$a_t = F_X a_{t-1} + g_X \epsilon_t / x_t$$

Where o_t is the Bernoulli distributed random variable (in case of normal data it equals to 1 for all observations), v_t is the state vector and l is the vector of lags, x_t is the vector of exogenous variables. w(.) is the measurement function, r(.) is the error function, f(.) is the transition function, g(.) is the persistence function and h(.) is the explanatory variables function. a_t is the vector of parameters for exogenous variables, F_X is the transitionX matrix and g_X is the persistenceX matrix. Finally, ϵ_t is the error term.

For the details see Hyndman et al.(2008).

For some more information about the model and its implementation, see the vignette: vignette("es","smooth").

Also, there are posts about the functions of the package smooth on the website of Ivan Svetunkov: https://forecasting.svetunkov.ru/en/tag/smooth/ - they explain the underlying models and how to use the functions.

Value

Object of class "smooth" is returned. It contains the list of the following values for classical ETS models:

- `model` - type of constructed model.
- `formula` - mathematical formula, describing interactions between components of es() and exogenous variables.
- `timeElapsed` - time elapsed for the construction of the model.
- `states` - matrix of the components of ETS.
- `persistence` - persistence vector. This is the place, where smoothing parameters live.
- `phi` - value of damping parameter.
- `transition` - transition matrix of the model.
- `measurement` - measurement vector of the model.
- `initialType` - type of the initial values used.
- `initial` - initial values of the state vector (non-seasonal).
- `initialSeason` - initial values of the seasonal part of state vector.
- `nParam` - table with the number of estimated / provided parameters. If a previous model was reused, then its initials are reused and the number of provided parameters will take this into account.
- `fitted` - fitted values of ETS. In case of the intermittent model, the fitted are multiplied by the probability of occurrence.
- `forecast` - point forecast of ETS.
- `lower` - lower bound of prediction interval. When `interval="none"` then NA is returned.
- `upper` - higher bound of prediction interval. When `interval="none"` then NA is returned.
- `residuals` - residuals of the estimated model.
- `errors` - trace forecast in-sample errors, returned as a matrix. In the case of trace forecasts this is the matrix used in optimisation. In non-trace estimations it is returned just for the information.
- `s2` - variance of the residuals (taking degrees of freedom into account). This is an unbiased estimate of variance.
- `interval` - type of interval asked by user.
- `level` - confidence level for interval.
- `cumulative` - whether the produced forecast was cumulative or not.
- `y` - original data.
- `holdout` - holdout part of the original data.
- `occurrence` - model of the class "oes" if the occurrence model was estimated. If the model is non-intermittent, then occurrence is `NULL`.
- `xreg` - provided vector or matrix of exogenous variables. If `xregDo="s"`, then this value will contain only selected exogenous variables.
- `updateX` - boolean, defining, if the states of exogenous variables were estimated as well.
- `initialX` - initial values for parameters of exogenous variables.
- `persistenceX` - persistence vector g for exogenous variables.
- `transitionX` - transition matrix F for exogenous variables.

- ICs - values of information criteria of the model. Includes AIC, AICc, BIC and BICc.
- logLik - concentrated log-likelihood of the function.
- lossValue - loss function value.
- loss - type of loss function used in the estimation.
- FI - Fisher Information. Equal to NULL if FI=FALSE or when FI is not provided at all.
- accuracy - vector of accuracy measures for the holdout sample. In case of non-intermittent data includes: MPE, MAPE, SMAPE, MASE, sMAE, RelMAE, sMSE and Bias coefficient (based on complex numbers). In case of intermittent data the set of errors will be: sMSE, sPIS, sCE (scaled cumulative error) and Bias coefficient. This is available only when holdout=TRUE.
- B - the vector of all the estimated parameters.

If combination of forecasts is produced (using model="CCC"), then a shorter list of values is returned:

- model,
- timeElapsed,
- initialType,
- fitted,
- forecast,
- lower,
- upper,
- residuals,
- s2 - variance of additive error of combined one-step-ahead forecasts,
- interval,
- level,
- cumulative,
- y,
- holdout,
- occurrence,
- ICs - combined ic,
- ICw - ic weights used in the combination,
- loss,
- xreg,
- accuracy.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

References

- Snyder, R. D., 1985. Recursive Estimation of Dynamic Linear Models. Journal of the Royal Statistical Society, Series B (Methodological) 47 (2), 272-276.
- Hyndman, R.J., Koehler, A.B., Ord, J.K., and Snyder, R.D. (2008) Forecasting with exponential smoothing: the state space approach, Springer-Verlag. http://dx.doi.org/10.1007/978-3-540-71918-2.
- Svetunkov Ivan and Boylan John E. (2017). Multiplicative State-Space Models for Intermittent Time Series. Working Paper of Department of Management Science, Lancaster University, 2017:4 , 1-43.
- Teunter R., Syntetos A., Babai Z. (2011). Intermittent demand: Linking forecasting to inventory obsolescence. European Journal of Operational Research, 214, 606-615.
- Croston, J. (1972) Forecasting and stock control for intermittent demands. Operational Research Quarterly, 23(3), 289-303.
- Syntetos, A., Boylan J. (2005) The accuracy of intermittent demand estimates. International Journal of Forecasting, 21(2), 303-314.
- Kolassa, S. (2011) Combining exponential smoothing forecasts using Akaike weights. International Journal of Forecasting, 27, pp 238 - 251.
- Taylor, J.W. and Bunn, D.W. (1999) A Quantile Regression Approach to Generating Prediction Intervals. Management Science, Vol 45, No 2, pp 225-237.
- Lichtendahl Kenneth C., Jr., Grushka-Cockayne Yael, Winkler Robert L., (2013) Is It Better to Average Probabilities or Quantiles? Management Science 59(7):1594-1611. DOI: [10.1287/mnsc.1120.1667](https://doi.org/10.1287/mnsc.1120.1667)

See Also

ets, forecast, ts, sim.es

Examples

```
library(Mcomp)

# See how holdout and trace parameters influence the forecast
es(M3$N1245$x,model="AAdN",h=8,holdout=FALSE,loss="MSE")
## Not run: es(M3$N2568$x,model="MAM",h=18,holdout=TRUE,loss="TMSE")

# Model selection example
es(M3$N1245$x,model="ZZN",ic="AIC",h=8,holdout=FALSE,bounds="a")

# Model selection. Compare AICc of these two models:
## Not run: es(M3$N1683$x,"ZZZ",h=10,holdout=TRUE)
es(M3$N1683$x,"MAdM",h=10,holdout=TRUE)
## End(Not run)

# Model selection, excluding multiplicative trend
## Not run: es(M3$N1245$x,model="ZXZ",h=8,holdout=TRUE)
```

```
# Combination example
## Not run: es(M3$N1245$x,model="CCN",h=8,holdout=TRUE)

# Model selection using a specified pool of models
ourModel <- es(M3$N1587$x,model=c("ANN","AAM","AMdA"),h=18)

# Redo previous model and produce prediction interval
es(M3$N1587$x,model=ourModel,h=18,interval="p")

# Semiparametric interval example
## Not run: es(M3$N1587$x,h=18,holdout=TRUE,interval="sp")

# Exogenous variables in ETS example
## Not run: x <- cbind(c(rep(0,25),1,rep(0,43)),c(rep(0,10),1,rep(0,58)))
y <- ts(c(M3$N1457$x,M3$N1457$xx),frequency=12)
es(y,h=18,holdout=TRUE,xreg=x,loss="aTMSE",interval="np")
ourModel <- es(ts(c(M3$N1457$x,M3$N1457$xx),frequency=12),h=18,holdout=TRUE,xreg=x,updateX=TRUE)
## End(Not run)

# This will be the same model as in previous line but estimated on new portion of data
## Not run: es(ts(c(M3$N1457$x,M3$N1457$xx),frequency=12),model=ourModel,h=18,holdout=FALSE)

# Intermittent data example
x <- rpois(100,0.2)
# Odds ratio model with the best ETS for demand sizes
es(x,"ZZN",occurrence="o")
# Inverse odds ratio model (underlies Croston) on iETS(M,N,N)
es(x,"MNN",occurrence="i")
# Constant probability based on iETS(M,N,N)
es(x,"MNN",occurrence="fixed")
# Best type of occurrence model based on iETS(Z,Z,N)
ourModel <- es(x,"ZZN",occurrence="auto")
par(mfcol=c(2,2))
plot(ourModel)

summary(ourModel)
forecast(ourModel)
plot(forecast(ourModel))
```

`forecast.smooth` *Forecasting time series using smooth functions*

Description

This function is created in order for the package to be compatible with Rob Hyndman's "forecast" package

Usage

```
## S3 method for class 'smooth'
forecast(object, h = 10, interval = c("parametric",
  "semiparametric", "nonparametric", "none"), level = 0.95,
  side = c("both", "upper", "lower"), ...)

## S3 method for class 'oes'
forecast(object, h = 10, interval = c("parametric",
  "semiparametric", "nonparametric", "none"), level = 0.95,
  side = c("both", "upper", "lower"), ...)

## S3 method for class 'msdecompose'
forecast(object, h = 10, interval = c("parametric",
  "semiparametric", "nonparametric", "none"), level = 0.95, model = NULL,
  ...)
```

Arguments

`object`	Time series model for which forecasts are required.
`h`	Forecast horizon
`interval`	Type of interval to construct. See es for details.
`level`	Confidence level. Defines width of prediction interval.
`side`	Defines, whether to provide `"both"` sides of prediction interval or only `"upper"`, or `"lower"`.
`...`	Other arguments accepted by either es, ces, gum or ssarima.
`model`	The type of ETS model to fit on the decomposed trend. Only applicable to "msdecompose" class. This is then returned in parameter "esmodel". If `NULL`, then it will be selected automatically based on the type of the used decomposition (either among pure additive or among pure additive ETS models).

Details

This is not a compulsory function. You can simply use es, ces, gum or ssarima without `forecast.smooth`. But if you are really used to `forecast` function, then go ahead!

Value

Returns object of class "smooth.forecast", which contains:

- `model` - the estimated model (ES / CES / GUM / SSARIMA).
- `method` - the name of the estimated model (ES / CES / GUM / SSARIMA).
- `forecast` aka `mean` - point forecasts of the model (conditional mean).
- `lower` - lower bound of prediction interval.
- `upper` - upper bound of prediction interval.
- `level` - confidence level.
- `interval` - binary variable (whether interval were produced or not).

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

References

Hyndman, R.J., Koehler, A.B., Ord, J.K., and Snyder, R.D. (2008) Forecasting with exponential smoothing: the state space approach, Springer-Verlag. http://www.exponentialsmoothing.net.

See Also

ets, forecast

Examples

```
ourModel <- ces(rnorm(100,0,1),h=10)

forecast.smooth(ourModel,h=10)
forecast.smooth(ourModel,h=10,interval=TRUE)
plot(forecast.smooth(ourModel,h=10,interval=TRUE))
```

---

gum *Generalised Univariate Model*

Description

Function constructs Generalised Univariate Model, estimating matrices F, w, vector g and initial parameters.

Usage

```
gum(y, orders = c(1, 1), lags = c(1, frequency(y)), type = c("additive",
  "multiplicative"), persistence = NULL, transition = NULL,
  measurement = NULL, initial = c("optimal", "backcasting"),
  ic = c("AICc", "AIC", "BIC", "BICc"), loss = c("MSE", "MAE", "HAM",
  "MSEh", "TMSE", "GTMSE", "MSCE"), h = 10, holdout = FALSE,
  cumulative = FALSE, interval = c("none", "parametric", "likelihood",
  "semiparametric", "nonparametric"), level = 0.95, occurrence = c("none",
  "auto", "fixed", "general", "odds-ratio", "inverse-odds-ratio", "direct"),
  oesmodel = "MNN", bounds = c("restricted", "admissible", "none"),
  silent = c("all", "graph", "legend", "output", "none"), xreg = NULL,
  xregDo = c("use", "select"), initialX = NULL, updateX = FALSE,
  persistenceX = NULL, transitionX = NULL, ...)

ges(...)
```

Arguments

`y` — Vector or ts object, containing data needed to be forecasted.

`orders` — Order of the model. Specified as vector of number of states with different lags. For example, `orders=c(1,1)` means that there are two states: one of the first lag type, the second of the second type.

`lags` — Defines lags for the corresponding orders. If, for example, `orders=c(1,1)` and lags are defined as `lags=c(1,12)`, then the model will have two states: the first will have lag 1 and the second will have lag 12. The length of `lags` must correspond to the length of `orders`.

`type` — Type of model. Can either be `"A"` - additive - or `"M"` - multiplicative. The latter means that the GUM is fitted on log-transformed data.

`persistence` — Persistence vector g, containing smoothing parameters. If `NULL`, then estimated.

`transition` — Transition matrix F. Can be provided as a vector. Matrix will be formed using the default `matrix(transition,nc,nc)`, where `nc` is the number of components in state vector. If `NULL`, then estimated.

`measurement` — Measurement vector w. If `NULL`, then estimated.

`initial` — Can be either character or a vector of initial states. If it is character, then it can be `"optimal"`, meaning that the initial states are optimised, or `"backcasting"`, meaning that the initials are produced using backcasting procedure.

`ic` — The information criterion used in the model selection procedure.

`loss` — The type of Loss Function used in optimization. `loss` can be: `MSE` (Mean Squared Error), `MAE` (Mean Absolute Error), `HAM` (Half Absolute Moment), `TMSE` - Trace Mean Squared Error, `GTMSE` - Geometric Trace Mean Squared Error, `MSEh` - optimisation using only h-steps ahead error, `MSCE` - Mean Squared Cumulative Error. If `loss!="MSE"`, then likelihood and model selection is done based on equivalent `MSE`. Model selection in this cases becomes not optimal.

There are also available analytical approximations for multistep functions: `aMSEh`, `aTMSE` and `aGTMSE`. These can be useful in cases of small samples.

Finally, just for fun the absolute and half analogues of multistep estimators are available: `MAEh`, `TMAE`, `GTMAE`, `MACE`, `TMAE`, `HAMh`, `THAM`, `GTHAM`, `CHAM`.

`h` — Length of forecasting horizon.

`holdout` — If `TRUE`, holdout sample of size `h` is taken from the end of the data.

`cumulative` — If `TRUE`, then the cumulative forecast and prediction interval are produced instead of the normal ones. This is useful for inventory control systems.

`interval` — Type of interval to construct. This can be:

- `"none"`, aka `"n"` - do not produce prediction interval.
- `"parametric"`, `"p"` - use state-space structure of ETS. In case of mixed models this is done using simulations, which may take longer time than for the pure additive and pure multiplicative models. This type of interval relies on unbiased estimate of in-sample error variance, which divides the sume of squared errors by T-k rather than just T.
- `"likelihood"`, `"l"` - these are the same as `"p"`, but relies on the biased estimate of variance from the likelihood (division by T, not by T-k).

- "semiparametric", "sp" - interval based on covariance matrix of 1 to h steps ahead errors and assumption of normal / log-normal distribution (depending on error type).
- "nonparametric", "np" - interval based on values from a quantile regression on error matrix (see Taylor and Bunn, 1999). The model used in this process is e[j] = a j^b, where j=1,..,h.

The parameter also accepts TRUE and FALSE. The former means that parametric interval are constructed, while the latter is equivalent to none. If the forecasts of the models were combined, then the interval are combined quantile-wise (Lichtendahl et al., 2013).

`level` Confidence level. Defines width of prediction interval.

`occurrence` The type of model used in probability estimation. Can be "none" - none, "fixed" - constant probability, "general" - the general Beta model with two parameters, "odds-ratio" - the Odds-ratio model with b=1 in Beta distribution, "inverse-odds-ratio" - the model with a=1 in Beta distribution, "direct" - the TSB-like (Teunter et al., 2011) probability update mechanism a+b=1, "auto" - the automatically selected type of occurrence model.

`oesmodel` The type of ETS model used for the modelling of the time varying probability. Object of the class "oes" can be provided here, and its parameters would be used in iETS model.

`bounds` What type of bounds to use in the model estimation. The first letter can be used instead of the whole word.

`silent` If silent="none", then nothing is silent, everything is printed out and drawn. silent="all" means that nothing is produced or drawn (except for warnings). In case of silent="graph", no graph is produced. If silent="legend", then legend of the graph is skipped. And finally silent="output" means that nothing is printed out in the console, but the graph is produced. silent also accepts TRUE and FALSE. In this case silent=TRUE is equivalent to silent="all", while silent=FALSE is equivalent to silent="none". The parameter also accepts first letter of words ("n", "a", "g", "l", "o").

`xreg` The vector (either numeric or time series) or the matrix (or data.frame) of exogenous variables that should be included in the model. If matrix included than columns should contain variables and rows - observations. Note that xreg should have number of observations equal either to in-sample or to the whole series. If the number of observations in xreg is equal to in-sample, then values for the holdout sample are produced using es function.

`xregDo` The variable defines what to do with the provided xreg: "use" means that all of the data should be used, while "select" means that a selection using ic should be done. "combine" will be available at some point in future...

`initialX` The vector of initial parameters for exogenous variables. Ignored if xreg is NULL.

`updateX` If TRUE, transition matrix for exogenous variables is estimated, introducing non-linear interactions between parameters. Prerequisite - non-NULL xreg.

`persistenceX` The persistence vector g_X, containing smoothing parameters for exogenous variables. If NULL, then estimated. Prerequisite - non-NULL xreg.

`transitionX` The transition matrix F_x for exogenous variables. Can be provided as a vector. Matrix will be formed using the default `matrix(transition,nc,nc)`, where `nc` is number of components in state vector. If `NULL`, then estimated. Prerequisite - non-NULL `xreg`.

`...` Other non-documented parameters. For example parameter `model` can accept a previously estimated GUM model and use all its parameters. `FI=TRUE` will make the function produce Fisher Information matrix, which then can be used to calculated variances of parameters of the model. You can also pass two parameters to the optimiser: 1. `maxeval` - maximum number of evaluations to carry on; 2. `xtol_rel` - the precision of the optimiser. The default values used in es() are `maxeval=5000` and `xtol_rel=1e-8`. You can read more about these parameters in the documentation of nloptr function.

Details

The function estimates the Single Source of Error state space model of the following type:

$$y_t = o_t(w' v_{t-l} + x_t a_{t-1} + \epsilon_t)$$

$$v_t = F v_{t-l} + g \epsilon_t$$

$$a_t = F_X a_{t-1} + g_X \epsilon_t / x_t$$

Where o_t is the Bernoulli distributed random variable (in case of normal data equal to 1), v_t is the state vector (defined using `orders`) and l is the vector of `lags`, x_t is the vector of exogenous parameters. w is the `measurement` vector, F is the `transition` matrix, g is the `persistence` vector, a_t is the vector of parameters for exogenous variables, F_X is the `transitionX` matrix and g_X is the `persistenceX` matrix. Finally, ϵ_t is the error term.

For some more information about the model and its implementation, see the vignette: `vignette("gum","smooth")`

Value

Object of class "smooth" is returned. It contains:

- `model` - name of the estimated model.
- `timeElapsed` - time elapsed for the construction of the model.
- `states` - matrix of fuzzy components of GUM, where `rows` correspond to time and `cols` to states.
- `initialType` - Type of the initial values used.
- `initial` - initial values of state vector (extracted from `states`).
- `nParam` - table with the number of estimated / provided parameters. If a previous model was reused, then its initials are reused and the number of provided parameters will take this into account.
- `measurement` - matrix w.

- transition - matrix F.
- persistence - persistence vector. This is the place, where smoothing parameters live.
- fitted - fitted values.
- forecast - point forecast.
- lower - lower bound of prediction interval. When interval="none" then NA is returned.
- upper - higher bound of prediction interval. When interval="none" then NA is returned.
- residuals - the residuals of the estimated model.
- errors - matrix of 1 to h steps ahead errors.
- s2 - variance of the residuals (taking degrees of freedom into account).
- interval - type of interval asked by user.
- level - confidence level for interval.
- cumulative - whether the produced forecast was cumulative or not.
- y - original data.
- holdout - holdout part of the original data.
- occurrence - model of the class "oes" if the occurrence model was estimated. If the model is non-intermittent, then occurrence is NULL.
- xreg - provided vector or matrix of exogenous variables. If xregDo="s", then this value will contain only selected exogenous variables.
- updateX - boolean, defining, if the states of exogenous variables were estimated as well.
- initialX - initial values for parameters of exogenous variables.
- persistenceX - persistence vector g for exogenous variables.
- transitionX - transition matrix F for exogenous variables.
- ICs - values of information criteria of the model. Includes AIC, AICc, BIC and BICc.
- logLik - log-likelihood of the function.
- lossValue - Cost function value.
- loss - Type of loss function used in the estimation.
- FI - Fisher Information. Equal to NULL if FI=FALSE or when FI variable is not provided at all.
- accuracy - vector of accuracy measures for the holdout sample. In case of non-intermittent data includes: MPE, MAPE, SMAPE, MASE, sMAE, RelMAE, sMSE and Bias coefficient (based on complex numbers). In case of intermittent data the set of errors will be: sMSE, sPIS, sCE (scaled cumulative error) and Bias coefficient. This is available only when holdout=TRUE.
- B - the vector of all the estimated parameters.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

References

- Svetunkov I. (2015 - Inf) "smooth" package for R - series of posts about the underlying models and how to use them: https://forecasting.svetunkov.ru/en/tag/smooth/.
- Svetunkov I. (2017). Statistical models underlying functions of 'smooth' package for R. Working Paper of Department of Management Science, Lancaster University 2017:1, 1-52.
- Taylor, J.W. and Bunn, D.W. (1999) A Quantile Regression Approach to Generating Prediction Intervals. Management Science, Vol 45, No 2, pp 225-237.
- Lichtendahl Kenneth C., Jr., Grushka-Cockayne Yael, Winkler Robert L., (2013) Is It Better to Average Probabilities or Quantiles? Management Science 59(7):1594-1611. DOI: [10.1287/mnsc.1120.1667](https://doi.org/10.1287/mnsc.1120.1667)

See Also

ets, es, ces, sim.es

Examples

```
# Something simple:
gum(rnorm(118,100,3),orders=c(1),lags=c(1),h=18,holdout=TRUE,bounds="a",interval="p")

# A more complicated model with seasonality
## Not run: ourModel <- gum(rnorm(118,100,3),orders=c(2,1),lags=c(1,4),h=18,holdout=TRUE)

# Redo previous model on a new data and produce prediction interval
## Not run: gum(rnorm(118,100,3),model=ourModel,h=18,interval="sp")

# Produce something crazy with optimal initials (not recommended)
## Not run: gum(rnorm(118,100,3),orders=c(1,1,1),lags=c(1,3,5),h=18,holdout=TRUE,initial="o")

# Simpler model estiamted using trace forecast error loss function and its analytical analogue
## Not run: gum(rnorm(118,100,3),orders=c(1),lags=c(1),h=18,holdout=TRUE,bounds="n",loss="TMSE")
gum(rnorm(118,100,3),orders=c(1),lags=c(1),h=18,holdout=TRUE,bounds="n",loss="aTMSE")
## End(Not run)

# Introduce exogenous variables
## Not run: gum(rnorm(118,100,3),orders=c(1),lags=c(1),h=18,holdout=TRUE,xreg=c(1:118))

# Ask for their update
## Not run: gum(rnorm(118,100,3),orders=c(1),lags=c(1),h=18,holdout=TRUE,xreg=c(1:118),updateX=TRUE)

# Do the same but now let's shrink parameters...
## Not run: gum(rnorm(118,100,3),orders=c(1),lags=c(1),h=18,xreg=c(1:118),updateX=TRUE,loss="TMSE")
ourModel <- gum(rnorm(118,100,3),orders=c(1),lags=c(1),h=18,holdout=TRUE,loss="aTMSE")
## End(Not run)

# Or select the most appropriate one
## Not run: gum(rnorm(118,100,3),orders=c(1),lags=c(1),h=18,holdout=TRUE,xreg=c(1:118),xregDo="s")

summary(ourModel)
```

```
forecast(ourModel)
plot(forecast(ourModel))
## End(Not run)
```

`is.smooth` *Smooth classes checkers*

Description

Functions to check if an object is of the specified class

Usage

```
is.smooth(x)

is.vsmooth(x)

is.smoothC(x)

is.msarima(x)

is.iss(x)

is.oes(x)

is.oesg(x)

is.viss(x)

is.smooth.sim(x)

is.vsmooth.sim(x)

is.smooth.forecast(x)

is.msdecompose(x)

is.msdecompose.forecast(x)
```

Arguments

x	The object to check.

Details

The list of functions includes:

- `is.smooth()` tests if the object was produced by a smooth function (e.g. es / ces / ssarima / gum / sma / msarima);
- `is.msarima()` tests if the object was produced by the msarima function;
- `is.smoothC()` tests if the object was produced by a combination function (currently applies only to smoothCombine);
- `is.vsmooth()` tests if the object was produced by a vector model (e.g. ves);
- `is.oes()` tests if the object was produced by oes function;
- `is.viss()` tests if the object was produced by viss function;
- `is.smooth.sim()` tests if the object was produced by simulate functions (e.g. sim.es / sim.ces / sim.ssarima / sim.sma / sim.gum);
- `is.vsmooth.sim()` tests if the object was produced by the functions sim.ves;
- `is.smooth.forecast()` checks if the forecast was produced from a smooth function using forecast() function.

Value

`TRUE` if this is the specified class and `FALSE` otherwise.

Author(s)

Ivan Svetunkov, `<ivan@svetunkov.ru>`

Examples

```
ourModel <- msarima(rnorm(100,100,10))

is.smooth(ourModel)
is.iss(ourModel)
is.msarima(ourModel)
is.vsmooth(ourModel)
```

---

iss *Intermittent State Space*

Description

Function calculates the probability for intermittent state space model. This is needed in order to forecast intermittent demand using other functions.

Usage

```
iss(data, intermittent = c("none", "fixed", "interval", "probability", "sba",
  "logistic"), ic = c("AICc", "AIC", "BIC", "BICc"), h = 10,
  holdout = FALSE, model = NULL, persistence = NULL, initial = NULL,
  initialSeason = NULL, xreg = NULL)
```

Arguments

data	Either numeric vector or time series vector.
intermittent	Type of method used in probability estimation. Can be "none" - none, "fixed" - constant probability, "croston" - estimated using Croston, 1972 method and "TSB" - Teunter et al., 2011 method., "sba" - Syntetos-Boylan Approximation for Croston's method (bias correction) discussed in Syntetos and Boylan, 2005, "logistic" - probability based on logit model.
ic	Information criteria to use in case of model selection.
h	Forecast horizon.
holdout	If TRUE, holdout sample of size h is taken from the end of the data.
model	Type of ETS model used for the estimation. Normally this should be either "ANN" or "MNN".
persistence	Persistence vector. If NULL, then it is estimated.
initial	Initial vector. If NULL, then it is estimated.
initialSeason	Initial vector of seasonal components. If NULL, then it is estimated.
xreg	Vector of matrix of exogenous variables, explaining some parts of occurrence variable (probability).

Details

The function estimates probability of demand occurrence, using one of the ETS state space models.

Value

The object of class "iss" is returned. It contains following list of values:

- `model` - the type of the estimated ETS model;
- `fitted` - fitted values of the constructed model;
- `forecast` - forecast for `h` observations ahead;
- `states` - values of states (currently level only);
- `variance` - conditional variance of the forecast;
- `logLik` - likelihood value for the model
- `nParam` - number of parameters used in the model;
- `residuals` - residuals of the model;
- `y` - actual values of probabilities (zeros and ones).
- `persistence` - the vector of smoothing parameters;
- `initial` - initial values of the state vector;
- `initialSeason` - the matrix of initials seasonal states;

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

References

- Svetunkov Ivan and Boylan John E. (2017). Multiplicative State-Space Models for Intermittent Time Series. Working Paper of Department of Management Science, Lancaster University, 2017:4 , 1-43.
- Teunter R., Syntetos A., Babai Z. (2011). Intermittent demand: Linking forecasting to inventory obsolescence. European Journal of Operational Research, 214, 606-615.
- Croston, J. (1972) Forecasting and stock control for intermittent demands. Operational Research Quarterly, 23(3), 289-303.
- Syntetos, A., Boylan J. (2005) The accuracy of intermittent demand estimates. International Journal of Forecasting, 21(2), 303-314.

See Also

ets, forecast, es

Examples

```
y <- rpois(100,0.1)
iss(y, intermittent="p")

iss(y, intermittent="i", persistence=0.1)
```

---

msarima *Multiple Seasonal ARIMA*

Description

Function constructs Multiple Seasonal State Space ARIMA, estimating AR, MA terms and initial states.

Usage

```
msarima(y, orders = list(ar = c(0), i = c(1), ma = c(1)), lags = c(1),
  constant = FALSE, AR = NULL, MA = NULL, initial = c("backcasting",
  "optimal"), ic = c("AICc", "AIC", "BIC", "BICc"), loss = c("MSE", "MAE",
  "HAM", "MSEh", "TMSE", "GTMSE", "MSCE"), h = 10, holdout = FALSE,
  cumulative = FALSE, interval = c("none", "parametric", "likelihood",
  "semiparametric", "nonparametric"), level = 0.95, occurrence = c("none",
  "auto", "fixed", "general", "odds-ratio", "inverse-odds-ratio", "direct"),
  oesmodel = "MNN", bounds = c("admissible", "none"), silent = c("all",
  "graph", "legend", "output", "none"), xreg = NULL, xregDo = c("use",
  "select"), initialX = NULL, updateX = FALSE, persistenceX = NULL,
  transitionX = NULL, ...)
```

Arguments

`y` Vector or ts object, containing data needed to be forecasted.

`orders` List of orders, containing vector variables `ar`, `i` and `ma`. Example: `orders=list(ar=c(1,2),i=c(1),ma=` If a variable is not provided in the list, then it is assumed to be equal to zero. At least one variable should have the same length as `lags`. Another option is to specify orders as a vector of a form `orders=c(p,d,q)`. The non-seasonal ARIMA(p,d,q) is constructed in this case.

`lags` Defines lags for the corresponding orders (see examples above). The length of `lags` must correspond to the length of either `ar`, `i` or `ma` in `orders` variable. There is no restrictions on the length of `lags` vector. It is recommended to order `lags` ascending. The orders are set by a user. If you want the automatic order selection, then use auto.ssarima function instead.

`constant` If `TRUE`, constant term is included in the model. Can also be a number (constant value).

`AR` Vector or matrix of AR parameters. The order of parameters should be lag-wise. This means that first all the AR parameters of the firs lag should be passed, then for the second etc. AR of another ssarima can be passed here.

`MA` Vector or matrix of MA parameters. The order of parameters should be lag-wise. This means that first all the MA parameters of the firs lag should be passed, then for the second etc. MA of another ssarima can be passed here.

`initial` Can be either character or a vector of initial states. If it is character, then it can be `"optimal"`, meaning that the initial states are optimised, or `"backcasting"`, meaning that the initials are produced using backcasting procedure.

`ic` The information criterion used in the model selection procedure.

`loss` The type of Loss Function used in optimization. `loss` can be: `MSE` (Mean Squared Error), `MAE` (Mean Absolute Error), `HAM` (Half Absolute Moment), `TMSE` - Trace Mean Squared Error, `GTMSE` - Geometric Trace Mean Squared Error, `MSEh` - optimisation using only h-steps ahead error, `MSCE` - Mean Squared Cumulative Error. If `loss!="MSE"`, then likelihood and model selection is done based on equivalent `MSE`. Model selection in this cases becomes not optimal.

There are also available analytical approximations for multistep functions: `aMSEh`, `aTMSE` and `aGTMSE`. These can be useful in cases of small samples.

Finally, just for fun the absolute and half analogues of multistep estimators are available: `MAEh`, `TMAE`, `GTMAE`, `MACE`, `TMAE`, `HAMh`, `THAM`, `GTHAM`, `CHAM`.

`h` Length of forecasting horizon.

`holdout` If `TRUE`, holdout sample of size `h` is taken from the end of the data.

`cumulative` If `TRUE`, then the cumulative forecast and prediction interval are produced instead of the normal ones. This is useful for inventory control systems.

`interval` Type of interval to construct. This can be:

- `"none"`, aka `"n"` - do not produce prediction interval.
- `"parametric"`, `"p"` - use state-space structure of ETS. In case of mixed models this is done using simulations, which may take longer time than for the pure additive and pure multiplicative models. This type of interval relies on unbiased estimate of in-sample error variance, which divides the sume of squared errors by T-k rather than just T.

- "likelihood", "l" - these are the same as "p", but relies on the biased estimate of variance from the likelihood (division by T, not by T-k).
- "semiparametric", "sp" - interval based on covariance matrix of 1 to h steps ahead errors and assumption of normal / log-normal distribution (depending on error type).
- "nonparametric", "np" - interval based on values from a quantile regression on error matrix (see Taylor and Bunn, 1999). The model used in this process is e[j] = a j^b, where j=1,..,h.

The parameter also accepts TRUE and FALSE. The former means that parametric interval are constructed, while the latter is equivalent to none. If the forecasts of the models were combined, then the interval are combined quantile-wise (Lichtendahl et al., 2013).

level
: Confidence level. Defines width of prediction interval.

occurrence
: The type of model used in probability estimation. Can be "none" - none, "fixed" - constant probability, "general" - the general Beta model with two parameters, "odds-ratio" - the Odds-ratio model with b=1 in Beta distribution, "inverse-odds-ratio" - the model with a=1 in Beta distribution, "direct" - the TSB-like (Teunter et al., 2011) probability update mechanism a+b=1, "auto" - the automatically selected type of occurrence model.

oesmodel
: The type of ETS model used for the modelling of the time varying probability. Object of the class "oes" can be provided here, and its parameters would be used in iETS model.

bounds
: What type of bounds to use in the model estimation. The first letter can be used instead of the whole word.

silent
: If silent="none", then nothing is silent, everything is printed out and drawn. silent="all" means that nothing is produced or drawn (except for warnings). In case of silent="graph", no graph is produced. If silent="legend", then legend of the graph is skipped. And finally silent="output" means that nothing is printed out in the console, but the graph is produced. silent also accepts TRUE and FALSE. In this case silent=TRUE is equivalent to silent="all", while silent=FALSE is equivalent to silent="none". The parameter also accepts first letter of words ("n", "a", "g", "l", "o").

xreg
: The vector (either numeric or time series) or the matrix (or data.frame) of exogenous variables that should be included in the model. If matrix included than columns should contain variables and rows - observations. Note that xreg should have number of observations equal either to in-sample or to the whole series. If the number of observations in xreg is equal to in-sample, then values for the holdout sample are produced using es function.

xregDo
: The variable defines what to do with the provided xreg: "use" means that all of the data should be used, while "select" means that a selection using ic should be done. "combine" will be available at some point in future...

initialX
: The vector of initial parameters for exogenous variables. Ignored if xreg is NULL.

updateX
: If TRUE, transition matrix for exogenous variables is estimated, introducing non-linear interactions between parameters. Prerequisite - non-NULL xreg.

persistenceX The persistence vector g_X, containing smoothing parameters for exogenous variables. If NULL, then estimated. Prerequisite - non-NULL xreg.

transitionX The transition matrix F_x for exogenous variables. Can be provided as a vector. Matrix will be formed using the default matrix(transition,nc,nc), where nc is number of components in state vector. If NULL, then estimated. Prerequisite - non-NULL xreg.

... Other non-documented parameters.

Parameter model can accept a previously estimated SARIMA model and use all its parameters.

FI=TRUE will make the function produce Fisher Information matrix, which then can be used to calculated variances of parameters of the model.

Details

The model, implemented in this function differs from the one in ssarima function (Svetunkov & Boylan, 2019), but it is more efficient and better fitting the data (which might be a limitation).

The basic ARIMA(p,d,q) used in the function has the following form:

$(1-B)^d(1-a_1B-a_2B^2-...-a_pB^p)y_[t]=(1+b_1B+b_2B^2+...+b_qB^q)\epsilon_[t]+c$

where $y_[t]$ is the actual values, $\epsilon_[t]$ is the error term, a_i, b_j are the parameters for AR and MA respectively and c is the constant. In case of non-zero differences c acts as drift.

This model is then transformed into ARIMA in the Single Source of Error State space form (based by Snyder, 1985, but in a slightly different formulation):

$y_t = o_t(w'v_{t-l} + x_t a_{t-1} + \epsilon_t)$

$v_t = F v_{t-l} + g\epsilon_t$

$a_t = F_X a_{t-1} + g_X \epsilon_t / x_t$

Where o_t is the Bernoulli distributed random variable (in case of normal data equal to 1), v_t is the state vector (defined based on orders) and l is the vector of lags, x_t is the vector of exogenous parameters. w is the measurement vector, F is the transition matrix, g is the persistence vector, a_t is the vector of parameters for exogenous variables, F_X is the transitionX matrix and g_X is the persistenceX matrix. The main difference from ssarima function is that this implementation skips zero polynomials, substantially decreasing the dimension of the transition matrix. As a result, this function works faster than ssarima on high frequency data, and it is more accurate.

Due to the flexibility of the model, multiple seasonalities can be used. For example, something crazy like this can be constructed: SARIMA(1,1,1)(0,1,1)[24](2,0,1)[24*7](0,0,1)[24*30], but the estimation may take some time... Still this should be estimated in finite time (not like with ssarima).

For some additional details see the vignette: vignette("ssarima","smooth")

Value

Object of class "smooth" is returned. It contains the list of the following values:

- model - the name of the estimated model.
- timeElapsed - time elapsed for the construction of the model.
- states - the matrix of the fuzzy components of ssarima, where rows correspond to time and cols to states.

- transition - matrix F.
- persistence - the persistence vector. This is the place, where smoothing parameters live.
- measurement - measurement vector of the model.
- AR - the matrix of coefficients of AR terms.
- I - the matrix of coefficients of I terms.
- MA - the matrix of coefficients of MA terms.
- constant - the value of the constant term.
- initialType - Type of the initial values used.
- initial - the initial values of the state vector (extracted from states).
- nParam - table with the number of estimated / provided parameters. If a previous model was reused, then its initials are reused and the number of provided parameters will take this into account.
- fitted - the fitted values.
- forecast - the point forecast.
- lower - the lower bound of prediction interval. When interval="none" then NA is returned.
- upper - the higher bound of prediction interval. When interval="none" then NA is returned.
- residuals - the residuals of the estimated model.
- errors - The matrix of 1 to h steps ahead errors.
- s2 - variance of the residuals (taking degrees of freedom into account).
- interval - type of interval asked by user.
- level - confidence level for interval.
- cumulative - whether the produced forecast was cumulative or not.
- y - the original data.
- holdout - the holdout part of the original data.
- occurrence - model of the class "oes" if the occurrence model was estimated. If the model is non-intermittent, then occurrence is NULL.
- xreg - provided vector or matrix of exogenous variables. If xregDo="s", then this value will contain only selected exogenous variables.
- updateX - boolean, defining, if the states of exogenous variables were estimated as well.
- initialX - initial values for parameters of exogenous variables.
- persistenceX - persistence vector g for exogenous variables.
- transitionX - transition matrix F for exogenous variables.
- ICs - values of information criteria of the model. Includes AIC, AICc, BIC and BICc.
- logLik - log-likelihood of the function.
- lossValue - Cost function value.
- loss - Type of loss function used in the estimation.
- FI - Fisher Information. Equal to NULL if FI=FALSE or when FI is not provided at all.
- accuracy - vector of accuracy measures for the holdout sample. In case of non-intermittent data includes: MPE, MAPE, SMAPE, MASE, sMAE, RelMAE, sMSE and Bias coefficient (based on complex numbers). In case of intermittent data the set of errors will be: sMSE, sPIS, sCE (scaled cumulative error) and Bias coefficient. This is available only when holdout=TRUE.
- B - the vector of all the estimated parameters.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

References

- Taylor, J.W. and Bunn, D.W. (1999) A Quantile Regression Approach to Generating Prediction Intervals. Management Science, Vol 45, No 2, pp 225-237.
- Lichtendahl Kenneth C., Jr., Grushka-Cockayne Yael, Winkler Robert L., (2013) Is It Better to Average Probabilities or Quantiles? Management Science 59(7):1594-1611. DOI: [10.1287/mnsc.1120.1667](https://doi.org/10.1287/mnsc.1120.1667)
- Snyder, R. D., 1985. Recursive Estimation of Dynamic Linear Models. Journal of the Royal Statistical Society, Series B (Methodological) 47 (2), 272-276.
- Hyndman, R.J., Koehler, A.B., Ord, J.K., and Snyder, R.D. (2008) Forecasting with exponential smoothing: the state space approach, Springer-Verlag. http://dx.doi.org/10.1007/978-3-540-71918-2.
- Svetunkov, I., & Boylan, J. E. (2019). State-space ARIMA for supply-chain forecasting. International Journal of Production Research, 0(0), 1–10. https://doi.org/10.1080/00207543.2019.1600764

See Also

auto.msarima, orders, ssarima, auto.arima

Examples

```
# The previous one is equivalent to:
ourModel <- msarima(rnorm(118,100,3),orders=c(1,1,1),lags=1,h=18,holdout=TRUE,interval="p")

# Example of SARIMA(2,0,0)(1,0,0)[4]
msarima(rnorm(118,100,3),orders=list(ar=c(2,1)),lags=c(1,4),h=18,holdout=TRUE)

# SARIMA of a peculiar order on AirPassengers data with Fisher Information
ourModel <- msarima(AirPassengers,orders=list(ar=c(1,0,3),i=c(1,0,1),ma=c(0,1,2)),
                    lags=c(1,6,12),h=10,holdout=TRUE,FI=TRUE)

# Construct the 95% confidence interval for the parameters of the model
ourCoefs <- coef(ourModel)
ourCoefsSD <- sqrt(abs(diag(solve(ourModel$FI))))
# Sort values accordingly
ourCoefs <- ourCoefs[names(ourCoefsSD)]
ourConfInt <- cbind(ourCoefs + qt(0.025,nobs(ourModel)) * ourCoefsSD,
                    ourCoefs + qt(0.975,nobs(ourModel)) * ourCoefsSD)
colnames(ourConfInt) <- c("2.25%","97.5%")
ourConfInt

# ARIMA(1,1,1) with Mean Squared Trace Forecast Error
msarima(rnorm(118,100,3),orders=list(ar=1,i=1,ma=1),lags=1,h=18,holdout=TRUE,loss="TMSE")
```

```
msarima(rnorm(118,100,3),orders=list(ar=1,i=1,ma=1),lags=1,h=18,holdout=TRUE,loss="aTMSE")

# SARIMA(0,1,1) with exogenous variables with crazy estimation of xreg
ourModel <- msarima(rnorm(118,100,3),orders=list(i=1,ma=1),h=18,holdout=TRUE,
                    xreg=c(1:118),updateX=TRUE)

summary(ourModel)
forecast(ourModel)
plot(forecast(ourModel))
```

---

msdecompose *Multiple seasonal classical decomposition*

Description

Function decomposes multiple seasonal time series into components using the principles of classical decomposition.

Usage

```
msdecompose(y, lags = c(12), type = c("additive", "multiplicative"))
```

Arguments

`y`	Vector or ts object, containing data needed to be smoothed.
`lags`	Vector of lags, corresponding to the frequencies in the data.
`type`	The type of decomposition. If `"multiplicative"` is selected, then the logarithm of data is taken prior to the decomposition.

Details

The function applies centred moving averages based on ma function and order specified in `lags` variable in order to smooth the original series and obtain level, trend and seasonal components of the series.

Value

The object of the class "msdecompose" is return, containing:

- `y` - the original time series.
- `initial` - the estimates of the initial level and trend.
- `trend` - the long term trend in the data.
- `seasonal` - the list of seasonal parameters.
- `lags` - the provided lags.
- `type` - the selected type of the decomposition.
- `yName` - the name of the provided data.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

See Also

ma

Examples

```
# Decomposition of multiple frequency data
## Not run: ourModel <- msdecompose(forecast::taylor, lags=c(48,336), type="m")
ourModel <- msdecompose(AirPassengers, lags=c(12), type="m")

plot(ourModel)
plot(forecast(ourModel, model="AAN", h=12))
```

---

multicov — *Function returns the multiple steps ahead covariance matrix of forecast errors*

Description

This function extracts covariance matrix of 1 to h steps ahead forecast errors for `ssarima()`, `gum()`, `sma()`, `es()` and `ces()` models.

Usage

```
multicov(object, type = c("analytical", "empirical", "simulated"), ...)

## S3 method for class 'smooth'
multicov(object, type = c("analytical", "empirical", "simulated"), ...)
```

Arguments

`object` Model estimated using one of the functions of smooth package.

`type` What method to use in order to produce covariance matrix:

1. `analytical` - based on the state space structure of the model and the one-step-ahead forecast error. This works for pure additive and pure multiplicative models. The values for the mixed models might be off.
2. `empirical` - based on the in-sample 1 to h steps ahead forecast errors (works fine on larger samples);
3. `simulated` - the data is simulated from the estimated model, then the same model is applied to it and then the empirical 1 to h steps ahead forecast errors are produced;

... Other parameters passed to simulate function (if type="simulated" is used). These are obs, nsim and seed. By default obs=1000, nsim=100. This approach increases the accuracy of covariance matrix on small samples and intermittent data;

Details

The function returns either scalar (if it is a non-smooth model) or the matrix of (h x h) size with variances and covariances of 1 to h steps ahead forecast errors. This is currently done based on empirical values. The analytical ones are more complicated.

Value

Scalar in cases of non-smooth functions. (h x h) matrix otherwise.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

See Also

orders

Examples

```
x <- rnorm(100,0,1)

# A simple example with a 5x5 covariance matrix
ourModel <- ces(x, h=5)
multicov(ourModel)
```

---

oes *Occurrence ETS model*

Description

Function returns the occurrence part of iETS model with the specified probability update and model types.

Usage

```
oes(y, model = "MNN", persistence = NULL, initial = "o",
  initialSeason = NULL, phi = NULL, occurrence = c("fixed", "general",
  "odds-ratio", "inverse-odds-ratio", "direct", "auto", "none"),
  ic = c("AICc", "AIC", "BIC", "BICc"), h = 10, holdout = FALSE,
  interval = c("none", "parametric", "likelihood", "semiparametric",
  "nonparametric"), level = 0.95, bounds = c("usual", "admissible",
```

```
"none"), silent = c("all", "graph", "legend", "output", "none"),
xreg = NULL, xregDo = c("use", "select"), initialX = NULL,
updateX = FALSE, transitionX = NULL, persistenceX = NULL, ...)
```

Arguments

y	Either numeric vector or time series vector.
model	The type of ETS model used for the estimation. Normally this should be "MNN" or any other pure multiplicative or additive model. The model selection is available here (although it's not fast), so you can use, for example, "YYN" and "XXN" for selecting between the pure multiplicative and pure additive models respectively. Using mixed models is possible, but not recommended.
persistence	Persistence vector g, containing smoothing parameters. If NULL, then estimated.
initial	Can be either character or a vector of initial states. If it is character, then it can be "optimal", meaning that the initial states are optimised, or "backcasting", meaning that the initials are produced using backcasting procedure.
initialSeason	The vector of the initial seasonal components. If NULL, then it is estimated.
phi	The value of the dampening parameter. Used only for damped-trend models.
occurrence	The type of model used in probability estimation. Can be "none" - none, "fixed" - constant probability, "odds-ratio" - the Odds-ratio model with b=1 in Beta distribution, "inverse-odds-ratio" - the model with a=1 in Beta distribution, "direct" - the TSB-like (Teunter et al., 2011) probability update mechanism a+b=1, "auto" - the automatically selected type of occurrence model, "general" - the general Beta model with two parameters. This will call oesg() function with two similar ETS models and the same provided parameters (initials and smoothing).
ic	The information criteria to use in case of model selection.
h	The forecast horizon.
holdout	If TRUE, holdout sample of size h is taken from the end of the data.
interval	Type of interval to construct. This can be: • "none", aka "n" - do not produce prediction interval. • "parametric", "p" - use state-space structure of ETS. In case of mixed models this is done using simulations, which may take longer time than for the pure additive and pure multiplicative models. This type of interval relies on unbiased estimate of in-sample error variance, which divides the sume of squared errors by T-k rather than just T. • "likelihood", "l" - these are the same as "p", but relies on the biased estimate of variance from the likelihood (division by T, not by T-k). • "semiparametric", "sp" - interval based on covariance matrix of 1 to h steps ahead errors and assumption of normal / log-normal distribution (depending on error type). • "nonparametric", "np" - interval based on values from a quantile regression on error matrix (see Taylor and Bunn, 1999). The model used in this process is e[j] = a j^b, where j=1,..,h.

The parameter also accepts TRUE and FALSE. The former means that parametric interval are constructed, while the latter is equivalent to none. If the forecasts of the models were combined, then the interval are combined quantile-wise (Lichtendahl et al., 2013).

level
: Confidence level. Defines width of prediction interval.

bounds
: What type of bounds to use in the model estimation. The first letter can be used instead of the whole word.

silent
: If silent="none", then nothing is silent, everything is printed out and drawn. silent="all" means that nothing is produced or drawn (except for warnings). In case of silent="graph", no graph is produced. If silent="legend", then legend of the graph is skipped. And finally silent="output" means that nothing is printed out in the console, but the graph is produced. silent also accepts TRUE and FALSE. In this case silent=TRUE is equivalent to silent="all", while silent=FALSE is equivalent to silent="none". The parameter also accepts first letter of words ("n", "a", "g", "l", "o").

xreg
: The vector or the matrix of exogenous variables, explaining some parts of occurrence variable (probability).

xregDo
: Variable defines what to do with the provided xreg: "use" means that all of the data should be used, while "select" means that a selection using ic should be done. "combine" will be available at some point in future...

initialX
: The vector of initial parameters for exogenous variables. Ignored if xreg is NULL.

updateX
: If TRUE, transition matrix for exogenous variables is estimated, introducing non-linear interactions between parameters. Prerequisite - non-NULL xreg.

transitionX
: The transition matrix F_x for exogenous variables. Can be provided as a vector. Matrix will be formed using the default matrix(transition,nc,nc), where nc is number of components in state vector. If NULL, then estimated. Prerequisite - non-NULL xreg.

persistenceX
: The persistence vector g_X, containing smoothing parameters for exogenous variables. If NULL, then estimated. Prerequisite - non-NULL xreg.

...
: The parameters passed to the optimiser, such as maxeval, xtol_rel, algorithm and print_level. The description of these is printed out by nloptr.print.options() function from the nloptr package. The default values in the oes function are maxeval=500, xtol_rel=1E-8, algorithm="NLOPT_LN_SBPLX" and print_level=0.

Details

The function estimates probability of demand occurrence, using the selected ETS state space models.

For the details about the model and its implementation, see the respective vignette: vignette("oes","smooth")

Value

The object of class "occurrence" is returned. It contains following list of values:

- model - the type of the estimated ETS model;

- timeElapsed - the time elapsed for the construction of the model;
- fitted - the fitted values for the probability;
- fittedModel - the fitted values of the underlying ETS model, where applicable (only for occurrence=c("o","i","d"));
- forecast - the forecast of the probability for h observations ahead;
- forecastModel - the forecast of the underlying ETS model, where applicable (only for occurrence=c("o","i","d"));
- lower - the lower bound of the interval if interval!="none";
- upper - the upper bound of the interval if interval!="none";
- lowerModel - the lower bound of the interval of the undelying ETS model if interval!="none";
- upperModel - the upper bound of the interval of the undelying ETS model if interval!="none";
- states - the values of the state vector;
- logLik - the log-likelihood value of the model;
- nParam - the number of parameters in the model (the matrix is returned);
- residuals - the residuals of the model;
- y - actual values of occurrence (zeros and ones).
- persistence - the vector of smoothing parameters;
- phi - the value of the damped trend parameter;
- initial - initial values of the state vector;
- initialSeason - the matrix of initials seasonal states;
- occurrence - the type of the occurrence model;
- updateX - boolean, defining, if the states of exogenous variables were estimated as well.
- initialX - initial values for parameters of exogenous variables.
- persistenceX - persistence vector g for exogenous variables.
- transitionX - transition matrix F for exogenous variables.
- accuracy - The error measures for the forecast (in case of holdout=TRUE).
- B - the vector of all the estimated parameters (in case of "odds-ratio", "inverse-odds-ratio" and "direct" models).

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

References

- Svetunkov Ivan and Boylan John E. (2017). Multiplicative State-Space Models for Intermittent Time Series. Working Paper of Department of Management Science, Lancaster University, 2017:4 , 1-43.
- Teunter R., Syntetos A., Babai Z. (2011). Intermittent demand: Linking forecasting to inventory obsolescence. European Journal of Operational Research, 214, 606-615.
- Croston, J. (1972) Forecasting and stock control for intermittent demands. Operational Research Quarterly, 23(3), 289-303.
- Syntetos, A., Boylan J. (2005) The accuracy of intermittent demand estimates. International Journal of Forecasting, 21(2), 303-314.

See Also

ets, oesg, es

Examples

```
y <- rpois(100,0.1)
oes(y, occurrence="auto")

oes(y, occurrence="f")
```

---

oesg *Occurrence ETS, general model*

Description

Function returns the general occurrence model of the of iETS model.

Usage

```
oesg(y, modelA = "MNN", modelB = "MNN", persistenceA = NULL,
  persistenceB = NULL, phiA = NULL, phiB = NULL, initialA = "o",
  initialB = "o", initialSeasonA = NULL, initialSeasonB = NULL,
  ic = c("AICc", "AIC", "BIC", "BICc"), h = 10, holdout = FALSE,
  interval = c("none", "parametric", "likelihood", "semiparametric",
  "nonparametric"), level = 0.95, bounds = c("usual", "admissible",
  "none"), silent = c("all", "graph", "legend", "output", "none"),
  xregA = NULL, xregB = NULL, initialXA = NULL, initialXB = NULL,
  xregDoA = c("use", "select"), xregDoB = c("use", "select"),
  updateXA = FALSE, updateXB = FALSE, transitionXA = NULL,
  transitionXB = NULL, persistenceXA = NULL, persistenceXB = NULL, ...)
```

Arguments

y	Either numeric vector or time series vector.
modelA	The type of the ETS for the model A.
modelB	The type of the ETS for the model B.
persistenceA	The persistence vector g, containing smoothing parameters used in the model A. If NULL, then estimated.
persistenceB	The persistence vector g, containing smoothing parameters used in the model B. If NULL, then estimated.
phiA	The value of the dampening parameter in the model A. Used only for damped-trend models.
phiB	The value of the dampening parameter in the model B. Used only for damped-trend models.

`initialA` Either `"o"` - optimal or the vector of initials for the level and / or trend for the model A.

`initialB` Either `"o"` - optimal or the vector of initials for the level and / or trend for the model B.

`initialSeasonA` The vector of the initial seasonal components for the model A. If `NULL`, then it is estimated.

`initialSeasonB` The vector of the initial seasonal components for the model B. If `NULL`, then it is estimated.

`ic` Information criteria to use in case of model selection.

`h` Forecast horizon.

`holdout` If `TRUE`, holdout sample of size `h` is taken from the end of the data.

`interval` Type of interval to construct. This can be:

- `"none"`, aka `"n"` - do not produce prediction interval.
- `"parametric"`, `"p"` - use state-space structure of ETS. In case of mixed models this is done using simulations, which may take longer time than for the pure additive and pure multiplicative models. This type of interval relies on unbiased estimate of in-sample error variance, which divides the sume of squared errors by T-k rather than just T.
- `"likelihood"`, `"l"` - these are the same as `"p"`, but relies on the biased estimate of variance from the likelihood (division by T, not by T-k).
- `"semiparametric"`, `"sp"` - interval based on covariance matrix of 1 to h steps ahead errors and assumption of normal / log-normal distribution (depending on error type).
- `"nonparametric"`, `"np"` - interval based on values from a quantile regression on error matrix (see Taylor and Bunn, 1999). The model used in this process is e[j] = a j^b, where j=1,..,h.

The parameter also accepts `TRUE` and `FALSE`. The former means that parametric interval are constructed, while the latter is equivalent to `none`. If the forecasts of the models were combined, then the interval are combined quantile-wise (Lichtendahl et al., 2013).

`level` Confidence level. Defines width of prediction interval.

`bounds` What type of bounds to use in the model estimation. The first letter can be used instead of the whole word.

`silent` If `silent="none"`, then nothing is silent, everything is printed out and drawn. `silent="all"` means that nothing is produced or drawn (except for warnings). In case of `silent="graph"`, no graph is produced. If `silent="legend"`, then legend of the graph is skipped. And finally `silent="output"` means that nothing is printed out in the console, but the graph is produced. `silent` also accepts `TRUE` and `FALSE`. In this case `silent=TRUE` is equivalent to `silent="all"`, while `silent=FALSE` is equivalent to `silent="none"`. The parameter also accepts first letter of words ("n", "a", "g", "l", "o").

`xregA` The vector or the matrix of exogenous variables, explaining some parts of occurrence variable of the model A.

xregB	The vector or the matrix of exogenous variables, explaining some parts of occurrence variable of the model B.
initialXA	The vector of initial parameters for exogenous variables in the model A. Ignored if xregA is NULL.
initialXB	The vector of initial parameters for exogenous variables in the model B. Ignored if xregB is NULL.
xregDoA	Variable defines what to do with the provided xregA: "use" means that all of the data should be used, while "select" means that a selection using ic should be done.
xregDoB	Similar to the xregDoA, but for the part B of the model.
updateXA	If TRUE, transition matrix for exogenous variables is estimated, introducing non-linear interactions between parameters. Prerequisite - non-NULL xregA.
updateXB	If TRUE, transition matrix for exogenous variables is estimated, introducing non-linear interactions between parameters. Prerequisite - non-NULL xregB.
transitionXA	The transition matrix F_x for exogenous variables of the model A. Can be provided as a vector. Matrix will be formed using the default matrix(transition,nc,nc), where nc is number of components in state vector. If NULL, then estimated. Prerequisite - non-NULL xregA.
transitionXB	The transition matrix F_x for exogenous variables of the model B. Similar to the transitionXA.
persistenceXA	The persistence vector g_X, containing smoothing parameters for the exogenous variables of the model A. If NULL, then estimated. Prerequisite - non-NULL xregA.
persistenceXB	The persistence vector g_X, containing smoothing parameters for the exogenous variables of the model B. If NULL, then estimated. Prerequisite - non-NULL xregB.
...	The parameters passed to the optimiser, such as maxeval, xtol_rel, algorithm and print_level. The description of these is printed out by nloptr.print.options() function from the nloptr package. The default values in the oes function are maxeval=500, xtol_rel=1E-8, algorithm="NLOPT_LN_SBPLX" and print_level=0.

Details

The function estimates probability of demand occurrence, based on the iETS_G state-space model. It involves the estimation and modelling of the two simultaneous state space equations. Thus two parts for the model type, persistence, initials etc.

For the details about the model and its implementation, see the respective vignette: vignette("oes","smooth")

The model is based on:

$$o_t \sim Bernoulli(p_t)$$

$$p_t = \frac{a_t}{a_t + b_t}$$

,

where a_t and b_t are the parameters of the Beta distribution and are modelled using separate ETS models.

Value

The object of class "occurrence" is returned. It contains following list of values:

- `modelA` - the model A of the class oes, that contains the output similar to the one from the `oes()` function;
- `modelB` - the model B of the class oes, that contains the output similar to the one from the `oes()` function.
- `B` - the vector of all the estimated parameters.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

See Also

`es`, `oes`

Examples

```
y <- rpois(100,0.1)
oesg(y, modelA="MNN", modelB="ANN")
```

---

orders — *Functions that extract values from the fitted model*

Description

These functions allow extracting orders and lags for `ssarima()`, `gum()` and `sma()`, type of model from `es()` and `ces()` and name of model.

Usage

```
orders(object, ...)

lags(object, ...)

modelName(object, ...)

modelType(object, ...)
```

Arguments

`object`	Model estimated using one of the functions of smooth package.
`...`	Currently nothing is accepted via ellipsis.

Details

orders() and lags() are useful only for SSARIMA, GUM and SMA. They return NA for other functions. This can also be applied to arima(), Arima() and auto.arima() functions from stats and forecast packages. modelType() is useful only for ETS and CES. They return NA for other functions. This can also be applied to ets() function from forecast package. errorType extracts the type of error from the model (either additive or multiplicative). Finally, modelName returns the name of the fitted model. For example, "ARIMA(0,1,1)". This is purely descriptive and can also be applied to non-smooth classes, so that, for example, you can easily extract the name of the fitted AR model from ar() function from stats package.

Value

Either vector, scalar or list with values is returned. orders() in case of ssarima returns list of values:

- ar - AR orders.
- i - I orders.
- ma - MA orders.

lags() returns the vector of lags of the model. All the other functions return strings of character.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

See Also

forecast, ssarima

Examples

```
x <- rnorm(100,0,1)

# Just as example. orders and lags do not return anything for ces() and es(). But modelType() does.
ourModel <- ces(x, h=10)
orders(ourModel)
lags(ourModel)
modelType(ourModel)
modelName(ourModel)

# And as another example it does the opposite for gum() and ssarima()
ourModel <- gum(x, h=10, orders=c(1,1), lags=c(1,4))
orders(ourModel)
lags(ourModel)
modelType(ourModel)
modelName(ourModel)

# Finally these values can be used for simulate functions or original functions.
ourModel <- auto.ssarima(x)
ssarima(x, orders=orders(ourModel), lags=lags(ourModel), constant=ourModel$constant)
```

```
sim.ssarima(orders=orders(ourModel), lags=lags(ourModel), constant=ourModel$constant)

ourModel <- es(x)
es(x, model=modelType(ourModel))
sim.es(model=modelType(ourModel))
```

---

plot.smooth *Plots for the fit and states*

Description

The function produces plot actuals, fitted values and forecasts and states of the model

Usage

```
## S3 method for class 'smooth'
plot(x, which = c(1, 2, 4, 6), level = 0.95,
  legend = FALSE, ask = prod(par("mfcol")) < length(which) &&
  dev.interactive(), lowess = TRUE, ...)

## S3 method for class 'msdecompose'
plot(x, which = c(1, 2, 4, 6), level = 0.95,
  legend = FALSE, ask = prod(par("mfcol")) < length(which) &&
  dev.interactive(), lowess = TRUE, ...)
```

Arguments

x Time series model for which forecasts are required.

which Which of the plots to produce. The possible options (see details for explanations):

1. Actuals vs Fitted values;
2. Standardised residuals vs Fitted;
3. Studentised residuals vs Fitted;
4. Absolute residuals vs Fitted;
5. Squared residuals vs Fitted;
6. Q-Q plot with the specified distribution;
7. Fitted over time;
8. Standardised residuals vs Time;
9. Studentised residuals vs Time;
10. ACF of the residuals;
11. PACF of the residuals.
12. Plot of states of the model.

level Confidence level. Defines width of confidence interval. Used in plots (2), (3), (7), (8), (9), (10) and (11).

`legend`	If `TRUE`, then the legend is produced on plots (2), (3) and (7).
`ask`	Logical; if `TRUE`, the user is asked to press Enter before each plot.
`lowess`	Logical; if `TRUE`, LOWESS lines are drawn on scatterplots, see lowess.
`...`	The parameters passed to the plot functions. Recommended to use with separate plots.

Details

The list of produced plots includes:

1. Actuals vs Fitted values. Allows analysing, whether there are any issues in the fit. Does the variability of actuals increase with the increase of fitted values? Is the relation well captured? They grey line on the plot corresponds to the perfect fit of the model.
2. Standardised residuals vs Fitted. Plots the points and the confidence bounds (red lines) for the specified confidence `level`. Useful for the analysis of outliers;
3. Studentised residuals vs Fitted. This is similar to the previous plot, but with the residuals divided by the scales with the leave-one-out approach. Should be more sensitive to outliers;
4. Absolute residuals vs Fitted. Useful for the analysis of heteroscedasticity;
5. Squared residuals vs Fitted - similar to (3), but with squared values;
6. Q-Q plot with the specified distribution. Can be used in order to see if the residuals follow the assumed distribution. The type of distribution depends on the one used in the estimation (see `distribution` parameter in alm);
7. ACF of the residuals. Are the residuals autocorrelated? See acf for details;
8. Fitted over time. Plots actuals (black line), fitted values (purple line), point forecast (blue line) and prediction interval (grey lines). Can be used in order to make sure that the model did not miss any important events over time;
9. Standardised residuals vs Time. Useful if you want to see, if there is autocorrelation or if there is heteroscedasticity in time. This also shows, when the outliers happen;
10. Studentised residuals vs Time. Similar to previous, but with studentised residuals;
11. PACF of the residuals. No, really, are they autocorrelated? See pacf function from stats package for details;
12. Plot of the states of the model. It is not recommended to produce this plot together with the others, because there might be several states, which would cause the creation of a different canvas. In case of "msdecompose", this will produce the decomposition of the series into states on a different canvas.

Which of the plots to produce, is specified via the `which` parameter.

Value

The function produces the number of plots, specified in the parameter `which`.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

See Also

plot.greybox

Examples

```
ourModel <- es(c(rnorm(50,100,10),rnorm(50,120,10)), "ANN", h=10)
par(mfcol=c(3,4))
plot(ourModel, c(1:11))
plot(ourModel, 12)
```

---

pls — *Prediction Likelihood Score*

Description

Function estimates Prediction Likelihood Score for the provided model

Usage

```
pls(object, holdout = NULL, ...)

## S3 method for class 'smooth'
pls(object, holdout = NULL, ...)
```

Arguments

object	The model estimated using smooth functions. This thing also accepts other models (e.g. estimated using functions from forecast package), but may not always work properly with them.
holdout	The values for the holdout part of the sample. If the model was fitted on the data with the `holdout=TRUE`, then the parameter is not needed.
...	Parameters passed to multicov function. The function is called in order to get the covariance matrix of 1 to h steps ahead forecast errors.

Details

Prediction likelihood score (PLS) is based on either normal or log-normal distribution of errors. This is extracted from the provided model. The likelihood based on the distribution of 1 to h steps ahead forecast errors is used in the process.

Value

A value of the log-likelihood.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

References

distribution. IEEE Signal Processing Letters. 13 (5): 300-303. doi: 10.1109/LSP.2006.870353 - this is not yet used in the function.

Snyder, R. D., Ord, J. K., Beaumont, A., 2012. Forecasting the intermittent demand for slow-moving inventories: A modelling approach. International Journal of Forecasting 28 (2), 485-496.

- Kolassa, S., 2016. Evaluating predictive count data distributions in retail sales forecasting. International Journal of Forecasting 32 (3), 788-803..

Examples

```
# Generate data, apply es() with the holdout parameter and calculate PLS
x <- rnorm(100,0,1)
ourModel <- es(x, h=10, holdout=TRUE, interval=TRUE)
pls(ourModel, type="a")
pls(ourModel, type="e")
pls(ourModel, type="s", obs=100, nsim=100)
```

---

sim.ces — *Simulate Complex Exponential Smoothing*

Description

Function generates data using CES with Single Source of Error as a data generating process.

Usage

```
sim.ces(seasonality = c("none", "simple", "partial", "full"), obs = 10,
  nsim = 1, frequency = 1, a = NULL, b = NULL, initial = NULL,
  randomizer = c("rnorm", "rt", "rlaplace", "rs"), probability = 1, ...)
```

Arguments

`seasonality`	The type of seasonality used in CES. Can be: `none` - No seasonality; `simple` - Simple seasonality, using lagged CES (based on `t-m` observation, where `m` is the seasonality lag); `partial` - Partial seasonality with real seasonal components (equivalent to additive seasonality); `full` - Full seasonality with complex seasonal components (can do both multiplicative and additive seasonality, depending on the data). First letter can be used instead of full words. Any seasonal CES can only be constructed for time series vectors.
`obs`	Number of observations in each generated time series.

`nsim`	Number of series to generate (number of simulations to do).
`frequency`	Frequency of generated data. In cases of seasonal models must be greater than 1.
`a`	First complex smoothing parameter. Should be a complex number. NOTE! CES is very sensitive to a and b values so it is advised to use values from previously estimated model.
`b`	Second complex smoothing parameter. Can be real if `seasonality="partial"`. In case of `seasonality="full"` must be complex number.
`initial`	A matrix with initial values for CES. In case with `seasonality="partial"` and `seasonality="full"` first two columns should contain initial values for non-seasonal components, repeated `frequency` times.
`randomizer`	Type of random number generator function used for error term. Defaults are: `rnorm`, `rt`, `rlaplace` and `rs`. `rlnorm` should be used for multiplicative models (e.g. ETS(M,N,N)). But any function from Distributions will do the trick if the appropriate parameters are passed. For example `rpois` with `lambda=2` can be used as well, but might result in weird values.
`probability`	Probability of occurrence, used for intermittent data generation. This can be a vector, implying that probability varies in time (in TSB or Croston style).
`...`	Additional parameters passed to the chosen randomizer. All the parameters should be passed in the order they are used in chosen randomizer. For example, passing just `sd=0.5` to `rnorm` function will lead to the call `rnorm(obs,mean=0.5,sd=1)`.

Details

For the information about the function, see the vignette: `vignette("simulate","smooth")`

Value

List of the following values is returned:

- `model` - Name of CES model.
- `a` - Value of complex smoothing parameter a. If `nsim>1`, then this is a vector.
- `b` - Value of complex smoothing parameter b. If `seasonality="n"` or `seasonality="s"`, then this is equal to NULL. If `nsim>1`, then this is a vector.
- `initial` - Initial values of CES in a form of matrix. If `nsim>1`, then this is an array.
- `data` - Time series vector (or matrix if `nsim>1`) of the generated series.
- `states` - Matrix (or array if `nsim>1`) of states. States are in columns, time is in rows.
- `residuals` - Error terms used in the simulation. Either vector or matrix, depending on `nsim`.
- `occurrence` - Values of occurrence variable. Once again, can be either a vector or a matrix...
- `logLik` - Log-likelihood of the constructed model.

Author(s)

Ivan Svetunkov, `<ivan@svetunkov.ru>`

References

- Svetunkov, I., Kourentzes, N. (February 2015). Complex exponential smoothing. Working Paper of Department of Management Science, Lancaster University 2015:1, 1-31.
- Svetunkov I., Kourentzes N. (2017) Complex Exponential Smoothing for Time Series Forecasting. Not yet published.

See Also

sim.es, sim.ssarima, ces, Distributions

Examples

```
# Create 120 observations from CES(n). Generate 100 time series of this kind.
x <- sim.ces("n",obs=120,nsim=100)

# Generate similar thing for seasonal series of CES(s)_4
x <- sim.ces("s",frequency=4,obs=80,nsim=100)

# Estimate model and then generate 10 time series from it
ourModel <- ces(rnorm(100,100,5))
simulate(ourModel,nsim=10)
```

---

sim.es *Simulate Exponential Smoothing*

Description

Function generates data using ETS with Single Source of Error as a data generating process.

Usage

```
sim.es(model = "ANN", obs = 10, nsim = 1, frequency = 1,
  persistence = NULL, phi = 1, initial = NULL, initialSeason = NULL,
  bounds = c("usual", "admissible", "restricted"), randomizer = c("rnorm",
  "rlnorm", "rt", "rlaplace", "rs"), probability = 1, ...)
```

Arguments

model	Type of ETS model according to [Hyndman et. al., 2008] taxonomy. Can consist of 3 or 4 chars: ANN, AAN, AAdN, AAA, AAdA, MAdM etc.
obs	Number of observations in each generated time series.
nsim	Number of series to generate (number of simulations to do).
frequency	Frequency of generated data. In cases of seasonal models must be greater than 1.

persistence — Persistence vector, which includes all the smoothing parameters. Must correspond to the chosen model. The maximum length is 3: level, trend and seasonal smoothing parameters. If `NULL`, values are generated.

phi — Value of damping parameter. If trend is not chosen in the model, the parameter is ignored.

initial — Vector of initial states of level and trend. The maximum length is 2. If `NULL`, values are generated.

initialSeason — Vector of initial states for seasonal coefficients. Should have length equal to `frequency` parameter. If `NULL`, values are generated.

bounds — Type of bounds to use for persistence vector if values are generated. `"usual"` - bounds from p.156 by Hyndman et. al., 2008. `"restricted"` - similar to `"usual"` but with upper bound equal to 0.3. `"admissible"` - bounds from tables 10.1 and 10.2 of Hyndman et. al., 2008. Using first letter of the type of bounds also works. These bounds are also used for multiplicative models, but the models are much more restrictive, so weird results might be obtained. Be careful!

randomizer — Type of random number generator function used for error term. Defaults are: `rnorm`, `rt`, `rlaplace` and `rs`. `rlnorm` should be used for multiplicative models (e.g. ETS(M,N,N)). But any function from Distributions will do the trick if the appropriate parameters are passed. For example `rpois` with `lambda=2` can be used as well, but might result in weird values.

probability — Probability of occurrence, used for intermittent data generation. This can be a vector, implying that probability varies in time (in TSB or Croston style).

... — Additional parameters passed to the chosen randomizer. All the parameters should be passed in the order they are used in chosen randomizer. For example, passing just `sd=0.5` to `rnorm` function will lead to the call `rnorm(obs,mean=0.5,sd=1)`. ATTENTION! When generating the multiplicative errors some tuning might be needed to obtain meaningful data. `sd=0.1` is usually already a high value for such models. ALSO NOTE: In case of multiplicative error model, the randomizer will generate `1+e_t` error, not `e_t`. This means that the mean should typically be equal to 1, not zero.

Details

For the information about the function, see the vignette: `vignette("simulate","smooth")`

Value

List of the following values is returned:

- `model` - Name of ETS model.
- `data` - Time series vector (or matrix if `nsim>1`) of the generated series.
- `states` - Matrix (or array if `nsim>1`) of states. States are in columns, time is in rows.
- `persistence` - Vector (or matrix if `nsim>1`) of smoothing parameters used in the simulation.
- `phi` - Value of damping parameter used in time series generation.
- `initial` - Vector (or matrix) of initial values.

- `initialSeason` - Vector (or matrix) of initial seasonal coefficients.
- `probability` - vector of probabilities used in the simulation.
- `intermittent` - type of the intermittent model used.
- `residuals` - Error terms used in the simulation. Either vector or matrix, depending on `nsim`.
- `occurrence` - Values of occurrence variable. Once again, can be either a vector or a matrix...
- `logLik` - Log-likelihood of the constructed model.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

References

- Snyder, R. D., 1985. Recursive Estimation of Dynamic Linear Models. Journal of the Royal Statistical Society, Series B (Methodological) 47 (2), 272-276.
- Hyndman, R.J., Koehler, A.B., Ord, J.K., and Snyder, R.D. (2008) Forecasting with exponential smoothing: the state space approach, Springer-Verlag. http://dx.doi.org/10.1007/978-3-540-71918-2.

See Also

es, ets, forecast, ts, Distributions

Examples

```
# Create 40 observations of quarterly data using AAA model with errors from normal distribution
ETSAAA <- sim.es(model="AAA",frequency=4,obs=40,randomizer="rnorm",mean=0,sd=100)

# Create 50 series of quarterly data using AAA model
# with 40 observations each with errors from normal distribution
ETSAAA <- sim.es(model="AAA",frequency=4,obs=40,randomizer="rnorm",mean=0,sd=100,nsim=50)

# Create 50 series of quarterly data using AAdA model
# with 40 observations each with errors from normal distribution
# and smoothing parameters lying in the "admissible" range.
ETSAAA <- sim.es(model="AAA",phi=0.9,frequency=4,obs=40,bounds="admissible",
                  randomizer="rnorm",mean=0,sd=100,nsim=50)

# Create 60 observations of monthly data using ANN model
# with errors from beta distribution
ETSANN <- sim.es(model="ANN",persistence=c(1.5),frequency=12,obs=60,
                  randomizer="rbeta",shape1=1.5,shape2=1.5)
plot(ETSANN$states)

# Create 60 observations of monthly data using MAM model
# with errors from uniform distribution
ETSMAM <- sim.es(model="MAM",persistence=c(0.3,0.2,0.1),initial=c(2000,50),
           phi=0.8,frequency=12,obs=60,randomizer="runif",min=-0.5,max=0.5)
```

```
# Create 80 observations of quarterly data using MMM model
# with predefined initial values and errors from the normal distribution
ETSMMM <- sim.es(model="MMM",persistence=c(0.1,0.1,0.1),initial=c(2000,1),
           initialSeason=c(1.1,1.05,0.9,.95),frequency=4,obs=80,mean=0,sd=0.01)

# Generate intermittent data using AAdN
iETSAAdN <- sim.es("AAdN",obs=30,frequency=1,probability=0.1,initial=c(3,0),phi=0.8)

# Generate iETS(MNN) with TSB style probabilities
oETSMNN <- sim.oes("MNN",obs=50,occurrence="d",persistence=0.2,initial=1,
                   randomizer="rlnorm",meanlog=0,sdlog=0.3)
iETSMNN <- sim.es("MNN",obs=50,frequency=12,persistence=0.2,initial=4,
                  probability=oETSMNN$probability)
```

---

sim.gum *Simulate Generalised Exponential Smoothing*

Description

Function generates data using GUM with Single Source of Error as a data generating process.

Usage

```
sim.gum(orders = c(1), lags = c(1), obs = 10, nsim = 1,
  frequency = 1, measurement = NULL, transition = NULL,
  persistence = NULL, initial = NULL, randomizer = c("rnorm", "rt",
  "rlaplace", "rs"), probability = 1, ...)
```

Arguments

`orders`	Order of the model. Specified as vector of number of states with different lags. For example, `orders=c(1,1)` means that there are two states: one of the first lag type, the second of the second type.
`lags`	Defines lags for the corresponding orders. If, for example, `orders=c(1,1)` and lags are defined as `lags=c(1,12)`, then the model will have two states: the first will have lag 1 and the second will have lag 12. The length of `lags` must correspond to the length of `orders`.
`obs`	Number of observations in each generated time series.
`nsim`	Number of series to generate (number of simulations to do).
`frequency`	Frequency of generated data. In cases of seasonal models must be greater than 1.
`measurement`	Measurement vector w. If `NULL`, then estimated.
`transition`	Transition matrix F. Can be provided as a vector. Matrix will be formed using the default `matrix(transition,nc,nc)`, where `nc` is the number of components in state vector. If `NULL`, then estimated.

persistence	Persistence vector g, containing smoothing parameters. If NULL, then estimated.
initial	Vector of initial values for state matrix. If NULL, then generated using advanced, sophisticated technique - uniform distribution.
randomizer	Type of random number generator function used for error term. Defaults are: rnorm, rt, rlaplace and rs. rlnorm should be used for multiplicative models (e.g. ETS(M,N,N)). But any function from Distributions will do the trick if the appropriate parameters are passed. For example rpois with lambda=2 can be used as well, but might result in weird values.
probability	Probability of occurrence, used for intermittent data generation. This can be a vector, implying that probability varies in time (in TSB or Croston style).
...	Additional parameters passed to the chosen randomizer. All the parameters should be passed in the order they are used in chosen randomizer. For example, passing just sd=0.5 to rnorm function will lead to the call rnorm(obs,mean=0.5,sd=1).

Details

For the information about the function, see the vignette: vignette("simulate","smooth")

Value

List of the following values is returned:

- model - Name of GUM model.
- measurement - Matrix w.
- transition - Matrix F.
- persistence - Persistence vector. This is the place, where smoothing parameters live.
- initial - Initial values of GUM in a form of matrix. If nsim>1, then this is an array.
- data - Time series vector (or matrix if nsim>1) of the generated series.
- states - Matrix (or array if nsim>1) of states. States are in columns, time is in rows.
- residuals - Error terms used in the simulation. Either vector or matrix, depending on nsim.
- occurrence - Values of occurrence variable. Once again, can be either a vector or a matrix...
- logLik - Log-likelihood of the constructed model.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

References

- Svetunkov I. (2015 - Inf) "smooth" package for R - series of posts about the underlying models and how to use them: https://forecasting.svetunkov.ru/en/tag/smooth/.
- Svetunkov I. (2017). Statistical models underlying functions of 'smooth' package for R. Working Paper of Department of Management Science, Lancaster University 2017:1, 1-52.

See Also

sim.es, sim.ssarima, sim.ces, gum, Distributions

Examples

```
# Create 120 observations from GUM(1[1]). Generate 100 time series of this kind.
x <- sim.gum(orders=c(1),lags=c(1),obs=120,nsim=100)

# Generate similar thing for seasonal series of GUM(1[1],1[4]])
x <- sim.gum(orders=c(1,1),lags=c(1,4),frequency=4,obs=80,nsim=100,transition=c(1,0,0.9,0.9))

# Estimate model and then generate 10 time series from it
ourModel <- gum(rnorm(100,100,5))
simulate(ourModel,nsim=10)
```

---

sim.oes *Simulate Occurrence Part of ETS model*

Description

Function generates data using ETS with Single Source of Error as a data generating process for the demand occurrence. As the main output it produces probabilities of occurrence.

Usage

```
sim.oes(model = "MNN", obs = 10, nsim = 1, frequency = 1,
  occurrence = c("odds-ratio", "inverse-odds-ratio", "direct", "general"),
  bounds = c("usual", "admissible", "restricted"), randomizer = c("rlnorm",
  "rinvgauss", "rgamma", "rnorm"), persistence = NULL, phi = 1,
  initial = NULL, initialSeason = NULL, modelB = model,
  persistenceB = persistence, phiB = phi, initialB = initial,
  initialSeasonB = initialSeason, ...)
```

Arguments

model	Type of ETS model according to [Hyndman et. al., 2008] taxonomy. Can consist of 3 or 4 chars: ANN, AAN, AAdN, AAA, AAdA, MAdM etc. The conventional oETS model assumes that the error term is positive, so "MZZ" models are recommended for this. If you use additive error models, then the function will exponentiate the obtained values before transforming them and getting the probability. This is the type of model A.
obs	Number of observations in each generated time series.
nsim	Number of series to generate (number of simulations to do).
frequency	Frequency of generated data. In cases of seasonal models must be greater than 1.

occurrence
: Type of occurrence model. See `vignette("oes","smooth")` for details.

bounds
: Type of bounds to use for persistence vector if values are generated. `"usual"` - bounds from p.156 by Hyndman et. al., 2008. `"restricted"` - similar to `"usual"` but with upper bound equal to 0.3. `"admissible"` - bounds from tables 10.1 and 10.2 of Hyndman et. al., 2008. Using first letter of the type of bounds also works. These bounds are also used for multiplicative models, but the models are much more restrictive, so weird results might be obtained. Be careful!

randomizer
: Type of random number generator function used for error term. It is advised to use `rlnorm()` or `rinvgauss()` in case of multiplicative error models. If a randomiser is used, it is advised to specify the parameters in the ellipsis.

persistence
: Persistence vector, which includes all the smoothing parameters. Must correspond to the chosen model. The maximum length is 3: level, trend and seasonal smoothing parameters. If `NULL`, values are generated.

phi
: Value of damping parameter. If trend is not chosen in the model, the parameter is ignored.

initial
: Vector of initial states of level and trend. The maximum length is 2. If `NULL`, values are generated.

initialSeason
: Vector of initial states for seasonal coefficients. Should have length equal to `frequency` parameter. If `NULL`, values are generated.

modelB
: Type of model B. This is used in `occurrence="general"` and `occurrence="inverse-odds-ratio"`.

persistenceB
: The persistence vector for the model B.

phiB
: Value of damping parameter for the model B.

initialB
: Vector of initial states of level and trend for the model B.

initialSeasonB
: Vector of initial states for seasonal coefficients for the model B.

...
: Additional parameters passed to the chosen randomizer. All the parameters should be passed in the order they are used in chosen randomizer. Both model A and model B share the same parameters for the randomizer.

Details

For the information about the function, see the vignette: `vignette("simulate","smooth")`

Value

List of the following values is returned:

- `model` - Name of ETS model.
- `modelA` - Model A, generated using `sim.es()` function;
- `modelB` - Model B, generated using `sim.es()` function;
- `probability` - The value of probability generated by the model;
- `occurrence` - Type of occurrence used in the model;
- `logLik` - Log-likelihood of the constructed model.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

References

- Snyder, R. D., 1985. Recursive Estimation of Dynamic Linear Models. Journal of the Royal Statistical Society, Series B (Methodological) 47 (2), 272-276.
- Hyndman, R.J., Koehler, A.B., Ord, J.K., and Snyder, R.D. (2008) Forecasting with exponential smoothing: the state space approach, Springer-Verlag. http://dx.doi.org/10.1007/978-3-540-71918-2.

See Also

oes, sim.es, Distributions

Examples

```
# This example uses rinvgauss function from statmod package.
## Not run: oETSMNNIG <- sim.oes(model="MNN",frequency=12,obs=60,
                              randomizer="rinvgauss",mean=1,dispersion=0.5)
## End(Not run)

# A simpler example with log normal distribution
oETSMNNlogN <- sim.oes(model="MNN",frequency=12,obs=60,initial=1,
                       randomizer="rlnorm",meanlog=0,sdlog=0.1)
```

---

sim.sma *Simulate Simple Moving Average*

Description

Function generates data using SMA in a Single Source of Error state space model as a data generating process.

Usage

```
sim.sma(order = NULL, obs = 10, nsim = 1, frequency = 1,
  initial = NULL, randomizer = c("rnorm", "rt", "rlaplace", "rs"),
  probability = 1, ...)
```

Arguments

order	Order of the modelled series. If omitted, then a random order from 1 to 100 is selected.
obs	Number of observations in each generated time series.
nsim	Number of series to generate (number of simulations to do).
frequency	Frequency of generated data. In cases of seasonal models must be greater than 1.
initial	Vector of initial states for the model. If `NULL`, values are generated.
randomizer	Type of random number generator function used for error term. Defaults are: `rnorm`, `rt`, `rlaplace` and `rs`. `rlnorm` should be used for multiplicative models (e.g. ETS(M,N,N)). But any function from Distributions will do the trick if the appropriate parameters are passed. For example `rpois` with `lambda=2` can be used as well, but might result in weird values.
probability	Probability of occurrence, used for intermittent data generation. This can be a vector, implying that probability varies in time (in TSB or Croston style).
...	Additional parameters passed to the chosen randomizer. All the parameters should be passed in the order they are used in chosen randomizer. For example, passing just `sd=0.5` to `rnorm` function will lead to the call `rnorm(obs,mean=0.5,sd=1)`.

Details

For the information about the function, see the vignette: `vignette("simulate","smooth")`

Value

List of the following values is returned:

- `model` - Name of SMA model.
- `data` - Time series vector (or matrix if `nsim>1`) of the generated series.
- `states` - Matrix (or array if `nsim>1`) of states. States are in columns, time is in rows.
- `initial` - Vector (or matrix) of initial values.
- `probability` - vector of probabilities used in the simulation.
- `intermittent` - type of the intermittent model used.
- `residuals` - Error terms used in the simulation. Either vector or matrix, depending on `nsim`.
- `occurrence` - Values of occurrence variable. Once again, can be either a vector or a matrix...
- `logLik` - Log-likelihood of the constructed model.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

References

- Snyder, R. D., 1985. Recursive Estimation of Dynamic Linear Models. Journal of the Royal Statistical Society, Series B (Methodological) 47 (2), 272-276.
- Hyndman, R.J., Koehler, A.B., Ord, J.K., and Snyder, R.D. (2008) Forecasting with exponential smoothing: the state space approach, Springer-Verlag. http://dx.doi.org/10.1007/978-3-540-71918-2.

See Also

es, ets, forecast, ts, Distributions

Examples

```
# Create 40 observations of quarterly data using AAA model with errors from normal distribution
sma10 <- sim.sma(order=10,frequency=4,obs=40,randomizer="rnorm",mean=0,sd=100)
```

---

sim.ssarima *Simulate SSARIMA*

Description

Function generates data using SSARIMA with Single Source of Error as a data generating process.

Usage

```
sim.ssarima(orders = list(ar = 0, i = 1, ma = 1), lags = 1, obs = 10,
  nsim = 1, frequency = 1, AR = NULL, MA = NULL, constant = FALSE,
  initial = NULL, bounds = c("admissible", "none"),
  randomizer = c("rnorm", "rt", "rlaplace", "rs"), probability = 1, ...)
```

Arguments

orders	List of orders, containing vector variables `ar`, `i` and `ma`. Example: `orders=list(ar=c(1,2),i=c(1),ma=` If a variable is not provided in the list, then it is assumed to be equal to zero. At least one variable should have the same length as `lags`.
lags	Defines lags for the corresponding orders (see examples above). The length of `lags` must correspond to the length of `orders`. There is no restrictions on the length of `lags` vector. It is recommended to order `lags` ascending.
obs	Number of observations in each generated time series.
nsim	Number of series to generate (number of simulations to do).
frequency	Frequency of generated data. In cases of seasonal models must be greater than 1.

AR	Vector or matrix of AR parameters. The order of parameters should be lag-wise. This means that first all the AR parameters of the firs lag should be passed, then for the second etc. AR of another ssarima can be passed here.
MA	Vector or matrix of MA parameters. The order of parameters should be lag-wise. This means that first all the MA parameters of the firs lag should be passed, then for the second etc. MA of another ssarima can be passed here.
constant	If TRUE, constant term is included in the model. Can also be a number (constant value).
initial	Vector of initial values for state matrix. If NULL, then generated using advanced, sophisticated technique - uniform distribution.
bounds	Type of bounds to use for AR and MA if values are generated. "admissible" - bounds guaranteeing stability and stationarity of SSARIMA. "none" - we generate something, but do not guarantee stationarity and stability. Using first letter of the type of bounds also works.
randomizer	Type of random number generator function used for error term. Defaults are: rnorm, rt, rlaplace and rs. rlnorm should be used for multiplicative models (e.g. ETS(M,N,N)). But any function from Distributions will do the trick if the appropriate parameters are passed. For example rpois with lambda=2 can be used as well, but might result in weird values.
probability	Probability of occurrence, used for intermittent data generation. This can be a vector, implying that probability varies in time (in TSB or Croston style).
...	Additional parameters passed to the chosen randomizer. All the parameters should be passed in the order they are used in chosen randomizer. For example, passing just sd=0.5 to rnorm function will lead to the call rnorm(obs,mean=0.5,sd=1).

Details

For the information about the function, see the vignette: vignette("simulate","smooth")

Value

List of the following values is returned:

- model - Name of SSARIMA model.
- AR - Value of AR parameters. If nsim>1, then this is a matrix.
- MA - Value of MA parameters. If nsim>1, then this is a matrix.
- constant - Value of constant term. If nsim>1, then this is a vector.
- initial - Initial values of SSARIMA. If nsim>1, then this is a matrix.
- data - Time series vector (or matrix if nsim>1) of the generated series.
- states - Matrix (or array if nsim>1) of states. States are in columns, time is in rows.
- residuals - Error terms used in the simulation. Either vector or matrix, depending on nsim.
- occurrence - Values of occurrence variable. Once again, can be either a vector or a matrix...
- logLik - Log-likelihood of the constructed model.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

References

- Snyder, R. D., 1985. Recursive Estimation of Dynamic Linear Models. Journal of the Royal Statistical Society, Series B (Methodological) 47 (2), 272-276.
- Hyndman, R.J., Koehler, A.B., Ord, J.K., and Snyder, R.D. (2008) Forecasting with exponential smoothing: the state space approach, Springer-Verlag. http://dx.doi.org/10.1007/978-3-540-71918-2.
- Svetunkov, I., & Boylan, J. E. (2019). State-space ARIMA for supply-chain forecasting. International Journal of Production Research, 0(0), 1–10. https://doi.org/10.1080/00207543.2019.1600764

See Also

sim.es, ssarima, Distributions, orders

Examples

```
# Create 120 observations from ARIMA(1,1,1) with drift. Generate 100 time series of this kind.
x <- sim.ssarima(ar.orders=1,i.orders=1,ma.orders=1,obs=120,nsim=100,constant=TRUE)

# Generate similar thing for seasonal series of SARIMA(1,1,1)(0,0,2)_4
x <- sim.ssarima(ar.orders=c(1,0),i.orders=c(1,0),ma.orders=c(1,2),lags=c(1,4),
                 frequency=4,obs=80,nsim=100,constant=FALSE)

# Generate 10 series of high frequency data from SARIMA(1,0,2)_1(0,1,1)_7(1,0,1)_30
x <- sim.ssarima(ar.orders=c(1,0,1),i.orders=c(0,1,0),ma.orders=c(2,1,1),lags=c(1,7,30),
                 obs=360,nsim=10)
```

---

sim.ves *Simulate Vector Exponential Smoothing*

Description

Function generates data using VES model as a data generating process.

Usage

```
sim.ves(model = "ANN", obs = 10, nsim = 1, nSeries = 2,
  frequency = 1, persistence = NULL, phi = 1, transition = NULL,
  initial = NULL, initialSeason = NULL,
  seasonal = c("individual, common"), weights = rep(1/nSeries, nSeries),
  bounds = c("usual", "admissible", "restricted"), randomizer = c("rnorm",
  "rt", "rlaplace", "rs"), ...)
```

Arguments

model	Type of ETS model. This can consist of 3 or 4 chars: ANN, AAN, AAdN, AAA, AAdA etc. Only pure additive models are supported. If you want to have multiplicative one, then just take exponent of the generated data.
obs	Number of observations in each generated time series.
nsim	Number of series to generate (number of simulations to do).
nSeries	Number of series in each generated group of series.
frequency	Frequency of generated data. In cases of seasonal models must be greater than 1.
persistence	Matrix of smoothing parameters for all the components of all the generated time series.
phi	Value of damping parameter. If trend is not chosen in the model, the parameter is ignored. If vector is provided, then several parameters are used for different series.
transition	Transition matrix. This should have the size appropriate to the selected model and nSeries. e.g. if ETS(A,A,N) is selected and nSeries=3, then the transition matrix should be 6 x 6. In case of damped trend, the phi parameter should be placed in the matrix manually. if NULL, then the default transition matrix for the selected type of model is used. If both phi and transition are provided, then the value of phi is ignored.
initial	Vector of initial states of level and trend. The minimum length is one (in case of ETS(A,N,N), the initial is used for all the series), the maximum length is 2 x nSeries. If NULL, values are generated for each series.
initialSeason	Vector or matrix of initial states for seasonal coefficients. Should have number of rows equal to frequency parameter. If NULL, values are generated for each series.
seasonal	The type of seasonal component across the series. Can be "individual", so that each series has its own component or "common", so that the component is shared across the series.
weights	The weights for the errors between the series with the common seasonal component. Ignored if seasonal="individual".
bounds	Type of bounds to use for persistence vector if values are generated. "usual" - bounds from p.156 by Hyndman et. al., 2008. "restricted" - similar to "usual" but with upper bound equal to 0.3. "admissible" - bounds from tables 10.1 and 10.2 of Hyndman et. al., 2008. Using first letter of the type of bounds also works.
randomizer	Type of random number generator function used for error term. Defaults are: rnorm, rt, rlaplace, rs. But any function from Distributions will do the trick if the appropriate parameters are passed. mvrnorm from MASS package can also be used.
...	Additional parameters passed to the chosen randomizer. All the parameters should be passed in the order they are used in chosen randomizer. For example, passing just sd=0.5 to rnorm function will lead to the call rnorm(obs,mean=0.5,sd=1). ATTENTION! When generating the multiplicative errors some tuning might be

needed to obtain meaningful data. sd=0.1 is usually already a high value for such models.

Details

For the information about the function, see the vignette: vignette("simulate","smooth")

Value

List of the following values is returned:

- model - Name of ETS model.
- data - The matrix (or an array if nsim>1) of the generated series.
- states - The matrix (or array if nsim>1) of states. States are in columns, time is in rows.
- persistence - The matrix (or array if nsim>1) of smoothing parameters used in the simulation.
- transition - The transition matrix (or array if nsim>1).
- initial - Vector (or matrix) of initial values.
- initialSeason - Vector (or matrix) of initial seasonal coefficients.
- residuals - Error terms used in the simulation. Either matrix or array, depending on nsim.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

References

- de Silva A,, Hyndman R.J. and Snyder, R.D. (2010). The vector innovations structural time series framework: a simple approach to multivariate forecasting. Statistical Modelling, 10 (4), pp.353-374
- Hyndman, R.J., Koehler, A.B., Ord, J.K., and Snyder, R.D. (2008) Forecasting with exponential smoothing: the state space approach, Springer-Verlag. http://www.exponentialsmoothing.net.
- Lütkepohl, H. (2005). New Introduction to Multiple Time Series Analysis. New introduction to Multiple Time Series Analysis. Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-27752-1

See Also

es, ets, forecast, ts, Distributions

Examples

```
# Create 40 observations of quarterly data using AAA model with errors
# from normal distribution
## Not run: VESAAA <- sim.ves(model="AAA",frequency=4,obs=40,nSeries=3,
                  randomizer="rnorm",mean=0,sd=100)
```

```
## End(Not run)

# You can also use mvrnorm function from MASS package as randomizer,
# but you need to provide mu and Sigma explicitly
## Not run: VESANN <- sim.ves(model="ANN",frequency=4,obs=40,nSeries=2,
                   randomizer="mvrnorm",mu=c(100,50),Sigma=matrix(c(40,20,20,30),2,2))
## End(Not run)

# When generating the data with multiplicative model a more diligent definitiion
# of parameters is needed. Here's an example with MMM model:

VESMMM <- sim.ves("AAA", obs=120, nSeries=2, frequency=12, initial=c(10,0),
          initialSeason=runif(12,-1,1), persistence=c(0.06,0.05,0.2), mean=0, sd=0.03)
VESMMM$data <- exp(VESMMM$data)

# Note that smoothing parameters should be low and the standard diviation should
# definitely be less than 0.1. Otherwise you might face the explosions.
```

---

sma *Simple Moving Average*

Description

Function constructs state space simple moving average of predefined order

Usage

```
sma(y, order = NULL, ic = c("AICc", "AIC", "BIC", "BICc"), h = 10,
  holdout = FALSE, cumulative = FALSE, interval = c("none", "parametric",
  "likelihood", "semiparametric", "nonparametric"), level = 0.95,
  silent = c("all", "graph", "legend", "output", "none"), ...)
```

Arguments

`y`	Vector or ts object, containing data needed to be forecasted.
`order`	Order of simple moving average. If `NULL`, then it is selected automatically using information criteria.
`ic`	The information criterion used in the model selection procedure.
`h`	Length of forecasting horizon.
`holdout`	If `TRUE`, holdout sample of size `h` is taken from the end of the data.
`cumulative`	If `TRUE`, then the cumulative forecast and prediction interval are produced instead of the normal ones. This is useful for inventory control systems.
`interval`	Type of interval to construct. This can be: • `"none"`, aka `"n"` - do not produce prediction interval.

- "parametric", "p" - use state-space structure of ETS. In case of mixed models this is done using simulations, which may take longer time than for the pure additive and pure multiplicative models. This type of interval relies on unbiased estimate of in-sample error variance, which divides the sume of squared errors by T-k rather than just T.
- "likelihood", "l" - these are the same as "p", but relies on the biased estimate of variance from the likelihood (division by T, not by T-k).
- "semiparametric", "sp" - interval based on covariance matrix of 1 to h steps ahead errors and assumption of normal / log-normal distribution (depending on error type).
- "nonparametric", "np" - interval based on values from a quantile regression on error matrix (see Taylor and Bunn, 1999). The model used in this process is e[j] = a j^b, where j=1,..,h.

The parameter also accepts TRUE and FALSE. The former means that parametric interval are constructed, while the latter is equivalent to none. If the forecasts of the models were combined, then the interval are combined quantile-wise (Lichtendahl et al., 2013).

level Confidence level. Defines width of prediction interval.

silent If silent="none", then nothing is silent, everything is printed out and drawn. silent="all" means that nothing is produced or drawn (except for warnings). In case of silent="graph", no graph is produced. If silent="legend", then legend of the graph is skipped. And finally silent="output" means that nothing is printed out in the console, but the graph is produced. silent also accepts TRUE and FALSE. In this case silent=TRUE is equivalent to silent="all", while silent=FALSE is equivalent to silent="none". The parameter also accepts first letter of words ("n", "a", "g", "l", "o").

... Other non-documented parameters. For example parameter model can accept a previously estimated SMA model and use its parameters.

Details

The function constructs AR model in the Single Source of Error state space form based on the idea that:

$y_t = \frac{1}{n} \sum_{j=1}^n y_{t-j}$

which is AR(n) process, that can be modelled using:

$y_t = w' v_{t-1} + \epsilon_t$

$v_t = F v_{t-1} + g \epsilon_t$

Where v_t is a state vector.

For some more information about the model and its implementation, see the vignette: vignette("sma","smooth")

Value

Object of class "smooth" is returned. It contains the list of the following values:

- model - the name of the estimated model.

- timeElapsed - time elapsed for the construction of the model.
- states - the matrix of the fuzzy components of ssarima, where rows correspond to time and cols to states.
- transition - matrix F.
- persistence - the persistence vector. This is the place, where smoothing parameters live.
- measurement - measurement vector of the model.
- order - order of moving average.
- initial - Initial state vector values.
- initialType - Type of initial values used.
- nParam - table with the number of estimated / provided parameters. If a previous model was reused, then its initials are reused and the number of provided parameters will take this into account.
- fitted - the fitted values.
- forecast - the point forecast.
- lower - the lower bound of prediction interval. When interval=FALSE then NA is returned.
- upper - the higher bound of prediction interval. When interval=FALSE then NA is returned.
- residuals - the residuals of the estimated model.
- errors - The matrix of 1 to h steps ahead errors.
- s2 - variance of the residuals (taking degrees of freedom into account).
- interval - type of interval asked by user.
- level - confidence level for interval.
- cumulative - whether the produced forecast was cumulative or not.
- y - the original data.
- holdout - the holdout part of the original data.
- ICs - values of information criteria of the model. Includes AIC, AICc, BIC and BICc.
- logLik - log-likelihood of the function.
- lossValue - Cost function value.
- loss - Type of loss function used in the estimation.
- accuracy - vector of accuracy measures for the holdout sample. Includes: MPE, MAPE, SMAPE, MASE, sMAE, RelMAE, sMSE and Bias coefficient (based on complex numbers). This is available only when holdout=TRUE.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

References

- Svetunkov I. (2015 - Inf) "smooth" package for R - series of posts about the underlying models and how to use them: https://forecasting.svetunkov.ru/en/tag/smooth/.
- Svetunkov I. (2017). Statistical models underlying functions of 'smooth' package for R. Working Paper of Department of Management Science, Lancaster University 2017:1, 1-52.
- Svetunkov, I., & Petropoulos, F. (2017). Old dog, new tricks: a modelling view of simple moving averages. International Journal of Production Research, 7543(January), 1-14. https://doi.org/10.1080/00207543.2017.1380326

See Also

ma, es, ssarima

Examples

```
# SMA of specific order
ourModel <- sma(rnorm(118,100,3),order=12,h=18,holdout=TRUE,interval="p")

# SMA of arbitrary order
ourModel <- sma(rnorm(118,100,3),h=18,holdout=TRUE,interval="sp")

summary(ourModel)
forecast(ourModel)
plot(forecast(ourModel))
```

---

smooth *Smooth package*

Description

Package contains functions implementing Single Source of Error state space models for purposes of time series analysis and forecasting.

Details

Package:	smooth
Type:	Package
Date:	2016-01-27 - Inf
License:	GPL-2

The following functions are included in the package:

- es - Exponential Smoothing in Single Source of Errors State Space form.

- ces - Complex Exponential Smoothing.
- gum - Generalised Exponential Smoothing.
- ssarima - SARIMA in state space framework.
- auto.ces - Automatic selection between seasonal and non-seasonal CES.
- auto.ssarima - Automatic selection of ARIMA orders.
- sma - Simple Moving Average in state space form.
- smoothCombine - the function that combines forecasts from es(), ces(), gum(), ssarima() and sma() functions.
- cma - Centered Moving Average. This is for smoothing time series, not for forecasting.
- ves - Vector Exponential Smoothing.
- sim.es - simulate time series using ETS as a model.
- sim.ces - simulate time series using CES as a model.
- sim.ssarima - simulate time series using SARIMA as a model.
- sim.gum - simulate time series using GUM as a model.
- sim.sma - simulate time series using SMA.
- iss - intermittent data state space model. This function models the part with data occurrences using one of three methods.
- viss - Does the same as iss, but for the multivariate models.

There are also several methods implemented in the package for the classes "smooth" and "smooth.sim":

- orders - extracts orders of the fitted model.
- lags - extracts lags of the fitted model.
- modelType - extracts type of the fitted model.
- forecast - produces forecast using provided model.
- multicov - returns covariance matrix of multiple steps ahead forecast errors.
- pls - returns Prediction Likelihood Score.
- nparam - returns number of the estimated parameters.
- fitted - extracts fitted values from provided model.
- getResponse - returns actual values from the provided model.
- residuals - extracts residuals of provided model.
- plot - plots either states of the model or produced forecast (depending on what object is passed).
- simulate - uses sim functions in order to simulate data using the provided object.
- summary - provides summary of the object.
- AICc, BICc - return, guess what...

Author(s)

Ivan Svetunkov

Maintainer: Ivan Svetunkov <ivan@svetunkov.ru>

References

- Snyder, R. D., 1985. Recursive Estimation of Dynamic Linear Models. Journal of the Royal Statistical Society, Series B (Methodological) 47 (2), 272-276.
- Hyndman, R.J., Koehler, A.B., Ord, J.K., and Snyder, R.D. (2008) Forecasting with exponential smoothing: the state space approach, Springer-Verlag. http://dx.doi.org/10.1007/978-3-540-71918-2.
- Svetunkov Ivan and Boylan John E. (2017). Multiplicative State-Space Models for Intermittent Time Series. Working Paper of Department of Management Science, Lancaster University, 2017:4 , 1-43.
- Teunter R., Syntetos A., Babai Z. (2011). Intermittent demand: Linking forecasting to inventory obsolescence. European Journal of Operational Research, 214, 606-615.
- Croston, J. (1972) Forecasting and stock control for intermittent demands. Operational Research Quarterly, 23(3), 289-303.
- Syntetos, A., Boylan J. (2005) The accuracy of intermittent demand estimates. International Journal of Forecasting, 21(2), 303-314.
- Svetunkov, I., Kourentzes, N. (February 2015). Complex exponential smoothing. Working Paper of Department of Management Science, Lancaster University 2015:1, 1-31.
- Svetunkov I., Kourentzes N. (2017) Complex Exponential Smoothing for Time Series Forecasting. Not yet published.
- Svetunkov I. (2015 - Inf) "smooth" package for R - series of posts about the underlying models and how to use them: https://forecasting.svetunkov.ru/en/tag/smooth/.
- Svetunkov I. (2017). Statistical models underlying functions of 'smooth' package for R. Working Paper of Department of Management Science, Lancaster University 2017:1, 1-52.
- Kolassa, S. (2011) Combining exponential smoothing forecasts using Akaike weights. International Journal of Forecasting, 27, pp 238 - 251.
- Taylor, J.W. and Bunn, D.W. (1999) A Quantile Regression Approach to Generating Prediction Intervals. Management Science, Vol 45, No 2, pp 225-237.
- Lichtendahl Kenneth C., Jr., Grushka-Cockayne Yael, Winkler Robert L., (2013) Is It Better to Average Probabilities or Quantiles? Management Science 59(7):1594-1611. DOI: [10.1287/mnsc.1120.1667](https://doi.org/10.1287/mnsc.1120.1667)

See Also

forecast, es, ssarima, ces, gum

Examples

```
## Not run: y <- ts(rnorm(100,10,3),frequency=12)

es(y,h=20,holdout=TRUE)
gum(y,h=20,holdout=TRUE)
auto.ces(y,h=20,holdout=TRUE)
```

```
auto.ssarima(y,h=20,holdout=TRUE)
## End(Not run)
```

`smoothCombine` *Combination of forecasts of state space models*

Description

Function constructs ETS, SSARIMA, CES, GUM and SMA and combines their forecasts using IC weights.

Usage

```
smoothCombine(y, models = NULL, initial = c("optimal", "backcasting"),
  ic = c("AICc", "AIC", "BIC", "BICc"), loss = c("MSE", "MAE", "HAM",
  "MSEh", "TMSE", "GTMSE", "MSCE"), h = 10, holdout = FALSE,
  cumulative = FALSE, interval = c("none", "parametric", "likelihood",
  "semiparametric", "nonparametric"), level = 0.95, bins = 200,
  intervalCombine = c("quantile", "probability"), occurrence = c("none",
  "auto", "fixed", "general", "odds-ratio", "inverse-odds-ratio",
  "probability"), oesmodel = "MNN", bounds = c("admissible", "none"),
  silent = c("all", "graph", "legend", "output", "none"), xreg = NULL,
  xregDo = c("use", "select"), initialX = NULL, updateX = FALSE,
  persistenceX = NULL, transitionX = NULL, ...)
```

Arguments

`y`	Vector or ts object, containing data needed to be forecasted.
`models`	List of the estimated smooth models to use in the combination. If `NULL`, then all the models are estimated in the function.
`initial`	Can be `"optimal"`, meaning that the initial states are optimised, or `"backcasting"`, meaning that the initials are produced using backcasting procedure.
`ic`	The information criterion used in the model selection procedure.
`loss`	The type of Loss Function used in optimization. `loss` can be: `MSE` (Mean Squared Error), `MAE` (Mean Absolute Error), `HAM` (Half Absolute Moment), `TMSE` - Trace Mean Squared Error, `GTMSE` - Geometric Trace Mean Squared Error, `MSEh` - optimisation using only h-steps ahead error, `MSCE` - Mean Squared Cumulative Error. If `loss!="MSE"`, then likelihood and model selection is done based on equivalent `MSE`. Model selection in this cases becomes not optimal. There are also available analytical approximations for multistep functions: `aMSEh`, `aTMSE` and `aGTMSE`. These can be useful in cases of small samples. Finally, just for fun the absolute and half analogues of multistep estimators are available: `MAEh`, `TMAE`, `GTMAE`, `MACE`, `TMAE`, `HAMh`, `THAM`, `GTHAM`, `CHAM`.
`h`	Length of forecasting horizon.

holdout
: If `TRUE`, holdout sample of size `h` is taken from the end of the data.

cumulative
: If `TRUE`, then the cumulative forecast and prediction interval are produced instead of the normal ones. This is useful for inventory control systems.

interval
: Type of interval to construct. This can be:

- `"none"`, aka `"n"` - do not produce prediction interval.
- `"parametric"`, `"p"` - use state-space structure of ETS. In case of mixed models this is done using simulations, which may take longer time than for the pure additive and pure multiplicative models. This type of interval relies on unbiased estimate of in-sample error variance, which divides the sume of squared errors by T-k rather than just T.
- `"likelihood"`, `"l"` - these are the same as `"p"`, but relies on the biased estimate of variance from the likelihood (division by T, not by T-k).
- `"semiparametric"`, `"sp"` - interval based on covariance matrix of 1 to h steps ahead errors and assumption of normal / log-normal distribution (depending on error type).
- `"nonparametric"`, `"np"` - interval based on values from a quantile regression on error matrix (see Taylor and Bunn, 1999). The model used in this process is e[j] = a j^b, where j=1,..,h.

The parameter also accepts `TRUE` and `FALSE`. The former means that parametric interval are constructed, while the latter is equivalent to `none`. If the forecasts of the models were combined, then the interval are combined quantile-wise (Lichtendahl et al., 2013).

level
: Confidence level. Defines width of prediction interval.

bins
: The number of bins for the prediction interval. The lower value means faster work of the function, but less precise estimates of the quantiles. This needs to be an even number.

intervalCombine
: How to average the prediction interval: quantile-wise (`"quantile"`) or probability-wise (`"probability"`).

occurrence
: The type of model used in probability estimation. Can be `"none"` - none, `"fixed"` - constant probability, `"general"` - the general Beta model with two parameters, `"odds-ratio"` - the Odds-ratio model with b=1 in Beta distribution, `"inverse-odds-ratio"` - the model with a=1 in Beta distribution, `"direct"` - the TSB-like (Teunter et al., 2011) probability update mechanism a+b=1, `"auto"` - the automatically selected type of occurrence model.

oesmodel
: The type of ETS model used for the modelling of the time varying probability. Object of the class "oes" can be provided here, and its parameters would be used in iETS model.

bounds
: What type of bounds to use in the model estimation. The first letter can be used instead of the whole word.

silent
: If `silent="none"`, then nothing is silent, everything is printed out and drawn. `silent="all"` means that nothing is produced or drawn (except for warnings). In case of `silent="graph"`, no graph is produced. If `silent="legend"`, then legend of the graph is skipped. And finally `silent="output"` means that nothing is printed out in the console, but the graph is produced. `silent` also accepts `TRUE` and `FALSE`. In this case `silent=TRUE` is equivalent to `silent="all"`,

while silent=FALSE is equivalent to silent="none". The parameter also accepts first letter of words ("n", "a", "g", "l", "o").

xreg
: The vector (either numeric or time series) or the matrix (or data.frame) of exogenous variables that should be included in the model. If matrix included than columns should contain variables and rows - observations. Note that xreg should have number of observations equal either to in-sample or to the whole series. If the number of observations in xreg is equal to in-sample, then values for the holdout sample are produced using es function.

xregDo
: The variable defines what to do with the provided xreg: "use" means that all of the data should be used, while "select" means that a selection using ic should be done. "combine" will be available at some point in future...

initialX
: The vector of initial parameters for exogenous variables. Ignored if xreg is NULL.

updateX
: If TRUE, transition matrix for exogenous variables is estimated, introducing non-linear interactions between parameters. Prerequisite - non-NULL xreg.

persistenceX
: The persistence vector g_X, containing smoothing parameters for exogenous variables. If NULL, then estimated. Prerequisite - non-NULL xreg.

transitionX
: The transition matrix F_x for exogenous variables. Can be provided as a vector. Matrix will be formed using the default matrix(transition,nc,nc), where nc is number of components in state vector. If NULL, then estimated. Prerequisite - non-NULL xreg.

...
: This currently determines nothing.

- timeElapsed - time elapsed for the construction of the model.
- initialType - type of the initial values used.
- fitted - fitted values of ETS.
- quantiles - the 3D array of produced quantiles if interval!="none" with the dimensions: (number of models) x (bins) x (h).
- forecast - point forecast of ETS.
- lower - lower bound of prediction interval. When interval="none" then NA is returned.
- upper - higher bound of prediction interval. When interval="none" then NA is returned.
- residuals - residuals of the estimated model.
- s2 - variance of the residuals (taking degrees of freedom into account).
- interval - type of interval asked by user.
- level - confidence level for interval.
- cumulative - whether the produced forecast was cumulative or not.
- y - original data.
- holdout - holdout part of the original data.
- occurrence - model of the class "oes" if the occurrence model was estimated. If the model is non-intermittent, then occurrence is NULL.
- xreg - provided vector or matrix of exogenous variables. If xregDo="s", then this value will contain only selected exogenous variables.

- updateX - boolean, defining, if the states of exogenous variables were estimated as well.
- ICs - values of information criteria of the model. Includes AIC, AICc, BIC and BICc.
- accuracy - vector of accuracy measures for the holdout sample. In case of non-intermittent data includes: MPE, MAPE, SMAPE, MASE, sMAE, RelMAE, sMSE and Bias coefficient (based on complex numbers). In case of intermittent data the set of errors will be: sMSE, sPIS, sCE (scaled cumulative error) and Bias coefficient.

Details

The combination of these models using information criteria weights is possible because they are all formulated in Single Source of Error framework. Due to the the complexity of some of the models, the estimation process may take some time. So be patient.

The prediction interval are combined either probability-wise or quantile-wise (Lichtendahl et al., 2013), which may take extra time, because we need to produce all the distributions for all the models. This can be sped up with the smaller value for bins parameter, but the resulting interval may be imprecise.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

References

- Snyder, R. D., 1985. Recursive Estimation of Dynamic Linear Models. Journal of the Royal Statistical Society, Series B (Methodological) 47 (2), 272-276.
- Hyndman, R.J., Koehler, A.B., Ord, J.K., and Snyder, R.D. (2008) Forecasting with exponential smoothing: the state space approach, Springer-Verlag. http://dx.doi.org/10.1007/978-3-540-71918-2.
- Kolassa, S. (2011) Combining exponential smoothing forecasts using Akaike weights. International Journal of Forecasting, 27, pp 238 - 251.
- Taylor, J.W. and Bunn, D.W. (1999) A Quantile Regression Approach to Generating Prediction Intervals. Management Science, Vol 45, No 2, pp 225-237.
- Lichtendahl Kenneth C., Jr., Grushka-Cockayne Yael, Winkler Robert L., (2013) Is It Better to Average Probabilities or Quantiles? Management Science 59(7):1594-1611. DOI: [10.1287/mnsc.1120.1667](https://doi.org/10.1287/mnsc.1120.1667)

See Also

es, auto.ssarima, auto.ces, auto.gum, sma

Examples

```
library(Mcomp)

ourModel <- smoothCombine(M3[[578]],interval="p")
plot(ourModel)

# models parameter accepts either previously estimated smoothCombine
# or a manually formed list of smooth models estimated in sample:
smoothCombine(M3[[578]],models=ourModel)

## Not run: models <- list(es(M3[[578]]), sma(M3[[578]]))
smoothCombine(M3[[578]],models=models)

## End(Not run)
```

`sowhat` *Function returns the ultimate answer to any question*

Description

You need description? So what?

Usage

```
sowhat(...)
```

Arguments

`...`	Any number of variables or string with a question.

Details

You need details? So what?

Value

It doesn't return any value, only messages. So what?

Author(s)

Ivan Svetunkov, `<ivan@svetunkov.ru>`

References

- Sowhat?
- Sowhat?
- 42

See Also

Nowwhat (to be implemented),

Examples

```
x <- rnorm(10000,0,1);
sowhat(x);

sowhat("What's the meaning of life?")

sowhat("I don't have a girlfriend.")
```

`ssarima` *State Space ARIMA*

Description

Function constructs State Space ARIMA, estimating AR, MA terms and initial states.

Usage

```
ssarima(y, orders = list(ar = c(0), i = c(1), ma = c(1)), lags = c(1),
  constant = FALSE, AR = NULL, MA = NULL, initial = c("backcasting",
  "optimal"), ic = c("AICc", "AIC", "BIC", "BICc"), loss = c("MSE", "MAE",
  "HAM", "MSEh", "TMSE", "GTMSE", "MSCE"), h = 10, holdout = FALSE,
  cumulative = FALSE, interval = c("none", "parametric", "likelihood",
  "semiparametric", "nonparametric"), level = 0.95, occurrence = c("none",
  "auto", "fixed", "general", "odds-ratio", "inverse-odds-ratio", "direct"),
  oesmodel = "MNN", bounds = c("admissible", "none"), silent = c("all",
  "graph", "legend", "output", "none"), xreg = NULL, xregDo = c("use",
  "select"), initialX = NULL, updateX = FALSE, persistenceX = NULL,
  transitionX = NULL, ...)
```

Arguments

`y`	Vector or ts object, containing data needed to be forecasted.
`orders`	List of orders, containing vector variables `ar`, `i` and `ma`. Example: `orders=list(ar=c(1,2),i=c(1),ma=` If a variable is not provided in the list, then it is assumed to be equal to zero. At least one variable should have the same length as `lags`. Another option is to specify orders as a vector of a form `orders=c(p,d,q)`. The non-seasonal ARIMA(p,d,q) is constructed in this case.
`lags`	Defines lags for the corresponding orders (see examples above). The length of `lags` must correspond to the length of either `ar`, `i` or `ma` in `orders` variable. There is no restrictions on the length of `lags` vector. It is recommended to order `lags` ascending. The orders are set by a user. If you want the automatic order selection, then use auto.ssarima function instead.

constant
: If TRUE, constant term is included in the model. Can also be a number (constant value).

AR
: Vector or matrix of AR parameters. The order of parameters should be lag-wise. This means that first all the AR parameters of the firs lag should be passed, then for the second etc. AR of another ssarima can be passed here.

MA
: Vector or matrix of MA parameters. The order of parameters should be lag-wise. This means that first all the MA parameters of the firs lag should be passed, then for the second etc. MA of another ssarima can be passed here.

initial
: Can be either character or a vector of initial states. If it is character, then it can be "optimal", meaning that the initial states are optimised, or "backcasting", meaning that the initials are produced using backcasting procedure.

ic
: The information criterion used in the model selection procedure.

loss
: The type of Loss Function used in optimization. loss can be: MSE (Mean Squared Error), MAE (Mean Absolute Error), HAM (Half Absolute Moment), TMSE - Trace Mean Squared Error, GTMSE - Geometric Trace Mean Squared Error, MSEh - optimisation using only h-steps ahead error, MSCE - Mean Squared Cumulative Error. If loss!="MSE", then likelihood and model selection is done based on equivalent MSE. Model selection in this cases becomes not optimal.

  There are also available analytical approximations for multistep functions: aMSEh, aTMSE and aGTMSE. These can be useful in cases of small samples.

  Finally, just for fun the absolute and half analogues of multistep estimators are available: MAEh, TMAE, GTMAE, MACE, TMAE, HAMh, THAM, GTHAM, CHAM.

h
: Length of forecasting horizon.

holdout
: If TRUE, holdout sample of size h is taken from the end of the data.

cumulative
: If TRUE, then the cumulative forecast and prediction interval are produced instead of the normal ones. This is useful for inventory control systems.

interval
: Type of interval to construct. This can be:

  - "none", aka "n" - do not produce prediction interval.
  - "parametric", "p" - use state-space structure of ETS. In case of mixed models this is done using simulations, which may take longer time than for the pure additive and pure multiplicative models. This type of interval relies on unbiased estimate of in-sample error variance, which divides the sume of squared errors by T-k rather than just T.
  - "likelihood", "l" - these are the same as "p", but relies on the biased estimate of variance from the likelihood (division by T, not by T-k).
  - "semiparametric", "sp" - interval based on covariance matrix of 1 to h steps ahead errors and assumption of normal / log-normal distribution (depending on error type).
  - "nonparametric", "np" - interval based on values from a quantile regression on error matrix (see Taylor and Bunn, 1999). The model used in this process is e[j] = a j^b, where j=1,..,h.

  The parameter also accepts TRUE and FALSE. The former means that parametric interval are constructed, while the latter is equivalent to none. If the forecasts of the models were combined, then the interval are combined quantile-wise (Lichtendahl et al., 2013).

`level`	Confidence level. Defines width of prediction interval.
`occurrence`	The type of model used in probability estimation. Can be `"none"` - none, `"fixed"` - constant probability, `"general"` - the general Beta model with two parameters, `"odds-ratio"` - the Odds-ratio model with b=1 in Beta distribution, `"inverse-odds-ratio"` - the model with a=1 in Beta distribution, `"direct"` - the TSB-like (Teunter et al., 2011) probability update mechanism a+b=1, `"auto"` - the automatically selected type of occurrence model.
`oesmodel`	The type of ETS model used for the modelling of the time varying probability. Object of the class "oes" can be provided here, and its parameters would be used in iETS model.
`bounds`	What type of bounds to use in the model estimation. The first letter can be used instead of the whole word.
`silent`	If `silent="none"`, then nothing is silent, everything is printed out and drawn. `silent="all"` means that nothing is produced or drawn (except for warnings). In case of `silent="graph"`, no graph is produced. If `silent="legend"`, then legend of the graph is skipped. And finally `silent="output"` means that nothing is printed out in the console, but the graph is produced. `silent` also accepts `TRUE` and `FALSE`. In this case `silent=TRUE` is equivalent to `silent="all"`, while `silent=FALSE` is equivalent to `silent="none"`. The parameter also accepts first letter of words ("n", "a", "g", "l", "o").
`xreg`	The vector (either numeric or time series) or the matrix (or data.frame) of exogenous variables that should be included in the model. If matrix included than columns should contain variables and rows - observations. Note that `xreg` should have number of observations equal either to in-sample or to the whole series. If the number of observations in `xreg` is equal to in-sample, then values for the holdout sample are produced using es function.
`xregDo`	The variable defines what to do with the provided xreg: `"use"` means that all of the data should be used, while `"select"` means that a selection using `ic` should be done. `"combine"` will be available at some point in future...
`initialX`	The vector of initial parameters for exogenous variables. Ignored if `xreg` is NULL.
`updateX`	If `TRUE`, transition matrix for exogenous variables is estimated, introducing non-linear interactions between parameters. Prerequisite - non-NULL `xreg`.
`persistenceX`	The persistence vector g_X, containing smoothing parameters for exogenous variables. If `NULL`, then estimated. Prerequisite - non-NULL `xreg`.
`transitionX`	The transition matrix F_x for exogenous variables. Can be provided as a vector. Matrix will be formed using the default `matrix(transition,nc,nc)`, where `nc` is number of components in state vector. If `NULL`, then estimated. Prerequisite - non-NULL `xreg`.
`...`	Other non-documented parameters. Parameter `model` can accept a previously estimated SSARIMA model and use all its parameters. `FI=TRUE` will make the function produce Fisher Information matrix, which then can be used to calculated variances of parameters of the model.

Details

The model, implemented in this function, is discussed in Svetunkov & Boylan (2019).

The basic ARIMA(p,d,q) used in the function has the following form:

$(1-B)^d(1-a_1B-a_2B^2-...-a_pB^p)y_[t]=(1+b_1B+b_2B^2+...+b_qB^q)\epsilon_[t]+c$

where $y_[t]$ is the actual values, $\epsilon_[t]$ is the error term, a_i, b_j are the parameters for AR and MA respectively and c is the constant. In case of non-zero differences c acts as drift.

This model is then transformed into ARIMA in the Single Source of Error State space form (proposed in Snyder, 1985):

$y_t = o_t(w'v_{t-l} + x_t a_{t-1} + \epsilon_t)$

$v_t = F v_{t-l} + g\epsilon_t$

$a_t = F_X a_{t-1} + g_X \epsilon_t / x_t$

Where o_t is the Bernoulli distributed random variable (in case of normal data equal to 1), v_t is the state vector (defined based on `orders`) and l is the vector of `lags`, x_t is the vector of exogenous parameters. w is the `measurement` vector, F is the `transition` matrix, g is the `persistence` vector, a_t is the vector of parameters for exogenous variables, F_X is the `transitionX` matrix and g_X is the `persistenceX` matrix.

Due to the flexibility of the model, multiple seasonalities can be used. For example, something crazy like this can be constructed: SARIMA(1,1,1)(0,1,1)[24](2,0,1)[24*7](0,0,1)[24*30], but the estimation may take some finite time... If you plan estimating a model with more than one seasonality, it is recommended to consider doing it using msarima.

The model selection for SSARIMA is done by the auto.ssarima function.

For some more information about the model and its implementation, see the vignette: `vignette("ssarima","smooth")`

Value

Object of class "smooth" is returned. It contains the list of the following values:

- `model` - the name of the estimated model.
- `timeElapsed` - time elapsed for the construction of the model.
- `states` - the matrix of the fuzzy components of ssarima, where `rows` correspond to time and `cols` to states.
- `transition` - matrix F.
- `persistence` - the persistence vector. This is the place, where smoothing parameters live.
- `measurement` - measurement vector of the model.
- `AR` - the matrix of coefficients of AR terms.
- `I` - the matrix of coefficients of I terms.
- `MA` - the matrix of coefficients of MA terms.
- `constant` - the value of the constant term.
- `initialType` - Type of the initial values used.
- `initial` - the initial values of the state vector (extracted from `states`).

- nParam - table with the number of estimated / provided parameters. If a previous model was reused, then its initials are reused and the number of provided parameters will take this into account.
- fitted - the fitted values.
- forecast - the point forecast.
- lower - the lower bound of prediction interval. When interval="none" then NA is returned.
- upper - the higher bound of prediction interval. When interval="none" then NA is returned.
- residuals - the residuals of the estimated model.
- errors - The matrix of 1 to h steps ahead errors.
- s2 - variance of the residuals (taking degrees of freedom into account).
- interval - type of interval asked by user.
- level - confidence level for interval.
- cumulative - whether the produced forecast was cumulative or not.
- y - the original data.
- holdout - the holdout part of the original data.
- occurrence - model of the class "oes" if the occurrence model was estimated. If the model is non-intermittent, then occurrence is NULL.
- xreg - provided vector or matrix of exogenous variables. If xregDo="s", then this value will contain only selected exogenous variables.
- updateX - boolean, defining, if the states of exogenous variables were estimated as well.
- initialX - initial values for parameters of exogenous variables.
- persistenceX - persistence vector g for exogenous variables.
- transitionX - transition matrix F for exogenous variables.
- ICs - values of information criteria of the model. Includes AIC, AICc, BIC and BICc.
- logLik - log-likelihood of the function.
- lossValue - Cost function value.
- loss - Type of loss function used in the estimation.
- FI - Fisher Information. Equal to NULL if FI=FALSE or when FI is not provided at all.
- accuracy - vector of accuracy measures for the holdout sample. In case of non-intermittent data includes: MPE, MAPE, SMAPE, MASE, sMAE, RelMAE, sMSE and Bias coefficient (based on complex numbers). In case of intermittent data the set of errors will be: sMSE, sPIS, sCE (scaled cumulative error) and Bias coefficient. This is available only when holdout=TRUE.
- B - the vector of all the estimated parameters.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

References

- Taylor, J.W. and Bunn, D.W. (1999) A Quantile Regression Approach to Generating Prediction Intervals. Management Science, Vol 45, No 2, pp 225-237.
- Lichtendahl Kenneth C., Jr., Grushka-Cockayne Yael, Winkler Robert L., (2013) Is It Better to Average Probabilities or Quantiles? Management Science 59(7):1594-1611. DOI: [10.1287/mnsc.1120.1667](https://doi.org/10.1287/mnsc.1120.1667)
- Snyder, R. D., 1985. Recursive Estimation of Dynamic Linear Models. Journal of the Royal Statistical Society, Series B (Methodological) 47 (2), 272-276.
- Hyndman, R.J., Koehler, A.B., Ord, J.K., and Snyder, R.D. (2008) Forecasting with exponential smoothing: the state space approach, Springer-Verlag. http://dx.doi.org/10.1007/978-3-540-71918-2.
- Svetunkov, I., & Boylan, J. E. (2019). State-space ARIMA for supply-chain forecasting. International Journal of Production Research, 0(0), 1–10. https://doi.org/10.1080/00207543.2019.1600764

See Also

auto.ssarima, orders, msarima, auto.msarima, sim.ssarima, auto.arima

Examples

```
# ARIMA(1,1,1) fitted to some data
ourModel <- ssarima(rnorm(118,100,3),orders=list(ar=c(1),i=c(1),ma=c(1)),lags=c(1),h=18,
                             holdout=TRUE,interval="p")

# The previous one is equivalent to:
## Not run: ourModel <- ssarima(rnorm(118,100,3),ar.orders=c(1),i.orders=c(1),ma.orders=c(1),lags=c(1),h=18,
                    holdout=TRUE,interval="p")
## End(Not run)

# Model with the same lags and orders, applied to a different data
ssarima(rnorm(118,100,3),orders=orders(ourModel),lags=lags(ourModel),h=18,holdout=TRUE)

# The same model applied to a different data
ssarima(rnorm(118,100,3),model=ourModel,h=18,holdout=TRUE)

# Example of SARIMA(2,0,0)(1,0,0)[4]
## Not run: ssarima(rnorm(118,100,3),orders=list(ar=c(2,1)),lags=c(1,4),h=18,holdout=TRUE)

# SARIMA(1,1,1)(0,0,1)[4] with different initialisations
## Not run: ssarima(rnorm(118,100,3),orders=list(ar=c(1),i=c(1),ma=c(1,1)),
        lags=c(1,4),h=18,holdout=TRUE)
ssarima(rnorm(118,100,3),orders=list(ar=c(1),i=c(1),ma=c(1,1)),
        lags=c(1,4),h=18,holdout=TRUE,initial="o")
## End(Not run)

# SARIMA of a peculiar order on AirPassengers data
```

```
## Not run: ssarima(AirPassengers,orders=list(ar=c(1,0,3),i=c(1,0,1),ma=c(0,1,2)),lags=c(1,6,12),
        h=10,holdout=TRUE)
## End(Not run)

# ARIMA(1,1,1) with Mean Squared Trace Forecast Error
## Not run: ssarima(rnorm(118,100,3),orders=list(ar=1,i=1,ma=1),lags=1,h=18,holdout=TRUE,loss="TMSE")
ssarima(rnorm(118,100,3),orders=list(ar=1,i=1,ma=1),lags=1,h=18,holdout=TRUE,loss="aTMSE")
## End(Not run)

# SARIMA(0,1,1) with exogenous variables
ssarima(rnorm(118,100,3),orders=list(i=1,ma=1),h=18,holdout=TRUE,xreg=c(1:118))

# SARIMA(0,1,1) with exogenous variables with crazy estimation of xreg
## Not run: ourModel <- ssarima(rnorm(118,100,3),orders=list(i=1,ma=1),h=18,holdout=TRUE,
                    xreg=c(1:118),updateX=TRUE)
## End(Not run)

summary(ourModel)
forecast(ourModel)
plot(forecast(ourModel))
```

---

ves *Vector Exponential Smoothing in SSOE state space model*

Description

Function constructs vector ETS model and returns forecast, fitted values, errors and matrix of states along with other useful variables.

Usage

```
ves(y, model = "ANN", persistence = c("common", "individual", "dependent",
  "seasonal-common"), transition = c("common", "individual", "dependent"),
  phi = c("common", "individual"), initial = c("individual", "common"),
  initialSeason = c("common", "individual"), seasonal = c("individual",
  "common"), weights = rep(1/ncol(y), ncol(y)), loss = c("likelihood",
  "diagonal", "trace"), ic = c("AICc", "AIC", "BIC", "BICc"), h = 10,
  holdout = FALSE, interval = c("none", "conditional", "unconditional",
  "individual", "likelihood"), level = 0.95, cumulative = FALSE,
  intermittent = c("none", "fixed", "logistic"), imodel = "ANN",
  iprobability = c("dependent", "independent"), bounds = c("admissible",
  "usual", "none"), silent = c("all", "graph", "output", "none"), ...)
```

Arguments

y	The matrix with the data, where series are in columns and observations are in rows.

`model` The type of ETS model. Can consist of 3 or 4 chars: `ANN`, `AAN`, `AAdN`, `AAA`, `AAdA`, `MMdM` etc. `ZZZ` means that the model will be selected based on the chosen information criteria type. ATTENTION! ONLY PURE ADDITIVE AND PURE MULTIPLICATIVE MODELS ARE CURRENTLY AVAILABLE + NO MODEL SELECTION IS AVAILABLE AT THIS STAGE! Pure multiplicative models are done as additive model applied to log(data).

Also `model` can accept a previously estimated VES model and use all its parameters.

Keep in mind that model selection with "Z" components uses Branch and Bound algorithm and may skip some models that could have slightly smaller information criteria.

`persistence` Persistence matrix G, containing smoothing parameters. Can be:

- `"independent"` - each series has its own smoothing parameters and no interactions are modelled (all the other values in the matrix are set to zero);
- `"dependent"` - each series has its own smoothing parameters, but interactions between the series are modelled (the whole matrix is estimated);
- `"group"` each series has the same smoothing parameters for respective components (the values of smoothing parameters are repeated, all the other values in the matrix are set to zero).
- `"seasonal"` - each component has its own smoothing parameter, except for the seasonal one, which is common across the time series.
- provided by user as a vector or as a matrix. The value is used by the model.

You can also use the first letter instead of writing the full word.

`transition` Transition matrix F. Can be:

- `"independent"` - each series has its own preset transition matrix and no interactions are modelled (all the other values in the matrix are set to zero);
- `"dependent"` - each series has its own transition matrix, but interactions between the series are modelled (the whole matrix is estimated). The estimated model behaves similar to VAR in this case;
- `"group"` each series has the same transition matrix for respective components (the values are repeated, all the other values in the matrix are set to zero).
- provided by user as a vector or as a matrix. The value is used by the model.

You can also use the first letter instead of writing the full word.

`phi` In cases of damped trend this parameter defines whether the phi should be estimated separately for each series (`"individual"`) or for the whole set (`"common"`). If vector or a value is provided here, then it is used by the model.

`initial` Can be either character or a vector / matrix of initial states. If it is character, then it can be `"individual"`, individual values of the initial non-seasonal components are used, or `"common"`, meaning that the initials for all the time series are set to be equal to the same value. If vector of states is provided, then it is automatically transformed into a matrix, assuming that these values are provided for the whole group.

`initialSeason` Can be either character or a vector / matrix of initial states. Treated the same way as `initial`. This means that different time series may share the same initial seasonal component.

`seasonal` The type of seasonal component across the series. Can be `"individual"`, so that each series has its own component or `"common"`, so that the component is shared across the series.

`weights` The weights for the errors between the series with the common seasonal component. Ignored if `seasonal="individual"`.

`loss` Type of Loss Function used in optimization. `loss` can be:

- `likelihood` - which assumes the minimisation of the determinant of the covariance matrix of errors between the series. This implies that the series could be correlated;
- `diagonal` - the covariance matrix is assumed to be diagonal with zeros off the diagonal. The determinant of this matrix is just a product of variances. This thing is minimised in this situation in logs.
- `trace` - the trace of the covariance matrix. The sum of variances is minimised in this case.

`ic` The information criterion used in the model selection procedure.

`h` Length of forecasting horizon.

`holdout` If `TRUE`, holdout sample of size `h` is taken from the end of the data.

`interval` Type of interval to construct.
This can be:

- `"none"`, aka `"n"` - do not produce prediction interval.
- `"conditional"`, `"c"` - produces multidimensional elliptic interval for each step ahead forecast. NOT AVAILABLE YET!
- `"unconditional"`, `"u"` - produces separate bounds for each series based on ellipses for each step ahead. These bounds correspond to min and max values of the ellipse assuming that all the other series but one take values in the centre of the ellipse. This leads to less accurate estimates of bounds (wider interval than needed), but these could still be useful. NOT AVAILABLE YET!
- `"independent"`, `"i"` - produces interval based on variances of each separate series. This does not take vector structure into account. In the calculation of covariance matrix, the division is done by T-k rather than T.
- `"likelihood"`, `"l"` - produces `"individual"` interval with the variance matrix estimated from the likelihood, which is a biased estimate of the true matrix. This means that the division of sum of squares is done by T rather than T-k.

The parameter also accepts `TRUE` and `FALSE`. The former means that the independent interval are constructed, while the latter is equivalent to `none`. You can also use the first letter instead of writing the full word.

`level` Confidence level. Defines width of prediction interval.

`cumulative` If `TRUE`, then the cumulative forecast and prediction interval are produced instead of the normal ones. This is useful for inventory control systems.

`intermittent` Defines type of intermittent model used. Can be:

- `none`, meaning that the data should be considered as non-intermittent;

- `fixed`, taking into account constant Bernoulli distribution of demand occurrences;
- `tsb`, based on Teunter et al., 2011 method.
- `auto` - automatic selection of intermittency type based on information criteria. The first letter can be used instead.

`imodel` Either character specifying what type of VES / ETS model should be used for probability modelling, or a model estimated using viss function.

`iprobability` Type of multivariate probability used in the model. Can be either `"independent"` or `"dependent"`. In the former case it is assumed that non-zeroes occur in each series independently. In the latter case each possible outcome is treated separately.

`bounds` What type of bounds to use in the model estimation. The first letter can be used instead of the whole word. `"admissible"` means that the model stability is ensured, while `"usual"` means that the all the parameters are restricted by the (0, 1) region.

`silent` If `silent="none"`, then nothing is silent, everything is printed out and drawn. `silent="all"` means that nothing is produced or drawn (except for warnings). In case of `silent="graph"`, no graph is produced. If `silent="legend"`, then legend of the graph is skipped. And finally `silent="output"` means that nothing is printed out in the console, but the graph is produced. `silent` also accepts `TRUE` and `FALSE`. In this case `silent=TRUE` is equivalent to `silent="all"`, while `silent=FALSE` is equivalent to `silent="none"`. The parameter also accepts first letter of words ("n", "a", "g", "l", "o").

`...` Other non-documented parameters. For example `FI=TRUE` will make the function also produce Fisher Information matrix, which then can be used to calculated variances of smoothing parameters and initial states of the model. The vector of initial parameter for the optimiser can be provided here as the variable `B`. The upper bound for the optimiser is provided via `ub`, while the lower one is `lb`. `maxeval=1000` is the default number of iterations for both optimisers used in the function. `algorithm1="NLOPT_LN_BOBYQA"` is the algorithm used in the first optimiser, while `algorithm2="NLOPT_LN_NELDERMEAD"` is the second one. `xtol_rel1=1e-8` is the relative tolerance in the first optimiser, while `xtol_rel2=1e-6` is for the second one. All of this can be amended and passed in ellipsis for finer tuning.

Details

Function estimates vector ETS in a form of the Single Source of Error state space model of the following type:

$$\mathbf{y}_t = \mathbf{o}_t(\mathbf{W}\mathbf{v}_{t-l} + \mathbf{x}_t\mathbf{a}_{t-1} + \epsilon_t)$$

$$\mathbf{v}_t = \mathbf{F}\mathbf{v}_{t-l} + \mathbf{G}\epsilon_t$$

$$\mathbf{a}_t = \mathbf{F_X}\mathbf{a}_{t-1} + \mathbf{G_X}\epsilon_t/\mathbf{x}_t$$

Where y_t is the vector of time series on observation t, o_t is the vector of Bernoulli distributed random variable (in case of normal data it becomes unit vector for all observations), $\mathbf{v}_t$ is the matrix of states and l is the matrix of lags, $\mathbf{x}_t$ is the vector of exogenous variables. $\mathbf{W}$ is the measurement matrix, $\mathbf{F}$ is the transition matrix and $\mathbf{G}$ is the persistence matrix. Finally, ϵ_t is the vector of error terms.

Conventionally we formulate values as:

$$\mathbf{y}'_t = (y_{1,t}, y_{2,t}, \dots, y_{m,t})$$

where m is the number of series in the group.

$$\mathbf{v}'_t = (v_{1,t}, v_{2,t}, \dots, v_{m,t})$$

where $v_{i,t}$ is vector of components for i-th time series.

$$\mathbf{W}' = (w_1, \dots, 0; \vdots, \ddots, \vdots; 0, \vdots, w_m)$$

is matrix of measurement vectors.

For the details on the additive model see Hyndman et al. (2008), chapter 17.

In case of multiplicative model, instead of the vector y_t we use its logarithms. As a result the multiplicative model is much easier to work with.

For some more information about the model and its implementation, see the vignette: `vignette("ves","smooth")`

Value

Object of class "vsmooth" is returned. It contains the following list of values:

- `model` - The name of the fitted model;
- `timeElapsed` - The time elapsed for the construction of the model;
- `states` - The matrix of states with components in columns and time in rows;
- `persistence` - The persistence matrix;
- `transition` - The transition matrix;
- `measurement` - The measurement matrix;
- `phi` - The damping parameter value;
- `B` - The vector of all the estimated coefficients;
- `initial` - The initial values of the non-seasonal components;
- `initialSeason` - The initial values of the seasonal components;
- `nParam` - The number of estimated parameters;
- `imodel` - The intermittent model estimated with VES;
- `y` - The matrix with the original data;
- `fitted` - The matrix of the fitted values;
- `holdout` - The matrix with the holdout values (if `holdout=TRUE` in the estimation);
- `residuals` - The matrix of the residuals of the model;

- `Sigma` - The covariance matrix of the errors (estimated with the correction for the number of degrees of freedom);
- `forecast` - The matrix of point forecasts;
- `PI` - The bounds of the prediction interval;
- `interval` - The type of the constructed prediction interval;
- `level` - The level of the confidence for the prediction interval;
- `ICs` - The values of the information criteria;
- `logLik` - The log-likelihood function;
- `lossValue` - The value of the loss function;
- `loss` - The type of the used loss function;
- `accuracy` - the values of the error measures. Currently not available.
- `FI` - Fisher information if user asked for it using `FI=TRUE`.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

References

- de Silva A,, Hyndman R.J. and Snyder, R.D. (2010). The vector innovations structural time series framework: a simple approach to multivariate forecasting. Statistical Modelling, 10 (4), pp.353-374
- Hyndman, R.J., Koehler, A.B., Ord, J.K., and Snyder, R.D. (2008) Forecasting with exponential smoothing: the state space approach, Springer-Verlag. http://www.exponentialsmoothing.net.
- Lütkepohl, H. (2005). New Introduction to Multiple Time Series Analysis. New introduction to Multiple Time Series Analysis. Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-27752-1

See Also

es, ets

Examples

```
Y <- ts(cbind(rnorm(100,100,10),rnorm(100,75,8)),frequency=12)

# The simplest model applied to the data with the default values
ves(Y,model="ANN",h=10,holdout=TRUE)

# Damped trend model with the dependent persistence
ves(Y,model="AAdN",persistence="d",h=10,holdout=TRUE)

# Multiplicative damped trend model with individual phi
ves(Y,model="MMdM",persistence="i",h=10,holdout=TRUE,initialSeason="c")
```

```
Y <- cbind(c(rpois(25,0.1),rpois(25,0.5),rpois(25,1),rpois(25,5)),
           c(rpois(25,0.1),rpois(25,0.5),rpois(25,1),rpois(25,5)))

# Intermittent VES with logistic probability
ves(Y,model="MNN",h=10,holdout=TRUE,intermittent="l")
```

---

viss *Vector Intermittent State Space*

Description

Function calculates the probability for vector intermittent state space model. This is needed in order to forecast intermittent demand using other functions.

Usage

```
viss(y, intermittent = c("logistic", "none", "fixed"), ic = c("AICc",
  "AIC", "BIC", "BICc"), h = 10, holdout = FALSE,
  probability = c("dependent", "independent"), model = "ANN",
  persistence = NULL, transition = NULL, phi = NULL, initial = NULL,
  initialSeason = NULL, xreg = NULL, ...)
```

Arguments

y	The matrix with data, where series are in columns and observations are in rows.
intermittent	Type of method used in probability estimation. Can be "none" - none, "fixed" - constant probability or "logistic" - probability based on logit model.
ic	Information criteria to use in case of model selection.
h	Forecast horizon.
holdout	If TRUE, holdout sample of size h is taken from the end of the data.
probability	Type of probability assumed in the model. If "dependent", then it is assumed that occurrence of one variable is connected with the occurrence with another one. In case of "independent" the occurrence of the variables is assumed to happen independent of each other.
model	Type of ETS model used for the estimation. Normally this should be either "ANN" or "MNN". If you assume that there are some tendencies in occurrence, then you can use more complicated models. Model selection is not yet available.
persistence	Persistence matrix type. If NULL, then it is estimated. See ves for the details.
transition	Transition matrix type. If NULL, then it is estimated. See ves for the details.
phi	Damping parameter type. If NULL, then it is estimated. See ves for the details.
initial	Initial vector type. If NULL, then it is estimated. See ves for the details.
initialSeason	Type of the initial vector of seasonal components. If NULL, then it is estimated. See ves for the details.
xreg	Vector of matrix of exogenous variables, explaining some parts of occurrence variable (probability).
...	Other parameters. This is not needed for now.

Details

The function estimates probability of demand occurrence, using one of the VES state-space models.

Value

The object of class "iss" is returned. It contains following list of values:

- `model` - the type of the estimated ETS model;
- `fitted` - fitted values of the constructed model;
- `forecast` - forecast for `h` observations ahead;
- `states` - values of states (currently level only);
- `variance` - conditional variance of the forecast;
- `logLik` - likelihood value for the model
- `nParam` - number of parameters used in the model;
- `residuals` - residuals of the model;
- `y` - actual values of probabilities (zeros and ones).
- `persistence` - the vector of smoothing parameters;
- `initial` - initial values of the state vector;
- `initialSeason` - the matrix of initials seasonal states;
- `intermittent` - type of intermittent model used;
- `probability` - type of probability used;
- `issModel` - intermittent state-space model used for calculations. Useful only in the case of `intermittent="l"` and `probability="d"`.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.ru>

References

- Svetunkov Ivan and Boylan John E. (2017). Multiplicative State-Space Models for Intermittent Time Series. Working Paper of Department of Management Science, Lancaster University, 2017:4 , 1-43.
- Teunter R., Syntetos A., Babai Z. (2011). Intermittent demand: Linking forecasting to inventory obsolescence. European Journal of Operational Research, 214, 606-615.
- Croston, J. (1972) Forecasting and stock control for intermittent demands. Operational Research Quarterly, 23(3), 289-303.
- Syntetos, A., Boylan J. (2005) The accuracy of intermittent demand estimates. International Journal of Forecasting, 21(2), 303-314.

See Also

`ets`, `forecast`, `es`

Examples

```
Y <- cbind(c(rpois(25,0.1),rpois(25,0.5),rpois(25,1),rpois(25,5)),
           c(rpois(25,0.1),rpois(25,0.5),rpois(25,1),rpois(25,5)))

viss(Y, intermittent="l")
viss(Y, intermittent="l", probability="i")
```

Index