Package 'legion'

May 8, 2026

Type Package

Title Forecasting Using Multivariate Models

Version 0.2.1

Date 2025-02-03

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

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

Language en-GB

Description Functions implementing multivariate state space models for purposes of time series analysis and forecasting.
The focus of the package is on multivariate models, such as Vector Exponential Smoothing, Vector ETS (Error-Trend-Seasonal model) etc. It currently includes Vector Exponential Smoothing (VES, de Silva et al., 2010, <doi:10.1177/1471082X0901000401>), Vector ETS (Svetunkov et al., 2023, <doi:10.1016/j.ejor.2022.04.040>) and simulation function for VES.

License LGPL-2.1

Depends R (>= 3.5.0), greybox (>= 1.0.4), smooth (>= 3.1.0)

Imports Rcpp (>= 0.12.3), stats, generics (>= 0.1.2), graphics, grDevices, Matrix, nloptr, utils, zoo

LinkingTo Rcpp, RcppArmadillo (>= 0.8.100.0.0)

Suggests numDeriv, testthat, knitr, rmarkdown, doMC, doParallel, foreach

VignetteBuilder knitr

RoxygenNote 7.3.2

Encoding UTF-8

NeedsCompilation yes

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

Maintainer Ivan Svetunkov <ivan@svetunkov.com>

Repository CRAN

Date/Publication 2025-02-03 16:10:02 UTC

Contents

auto.vets 2
is.legion 7
legion 8
oves 9
plot.legion 11
sim.ves 13
ves 15

Index **21**

`auto.vets` *Vector ETS-PIC model*

Description

Function constructs vector ETS model based on VETS-PIC taxonomy and returns forecast, fitted values, errors and matrix of states along with other useful variables.

Usage

```
auto.vets(data, model = "PPP", lags = c(frequency(data)),
  loss = c("likelihood", "diagonal", "trace"), ic = c("AICc", "AIC", "BIC",
  "BICc"), h = 10, holdout = FALSE, occurrence = c("none", "fixed",
  "logistic"), bounds = c("admissible", "usual", "none"), silent = TRUE,
  parallel = FALSE, ...)

vets(data, model = "PPP", lags = c(frequency(data)),
  parameters = c("level", "trend", "seasonal", "damped"),
  initials = c("seasonal"), components = c("none"),
  loss = c("likelihood", "diagonal", "trace"), ic = c("AICc", "AIC", "BIC",
  "BICc"), h = 10, holdout = FALSE, occurrence = c("none", "fixed",
  "logistic"), bounds = c("admissible", "usual", "none"), silent = TRUE,
  ...)
```

Arguments

`data`	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. `PPP` means that the best pure model will be selected based on the chosen information criteria type. ATTENTION! ONLY PURE ADDITIVE AND

PURE MULTIPLICATIVE MODELS ARE AVAILABLE! Pure multiplicative models are done as additive model applied to log(y).

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

`lags` The lags of the model. Needed for seasonal models.

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

- `"likelihood"` - which implies the maximisation of likelihood of multivariate normal distribution (or log Normal if the multiplicative model is constructed);
- `"diagonal"` - similar to `"likelihood"`, but assumes that covariances between the error terms are zero.
- `"trace"` - the trace of the covariance matrix of errors. The sum of variances is minimised in this case.
- Provided by user as a custom function of `actual`, `fitted` and `B`. Note that internally function transposes the data, so that `actual` and `fitted` contain observations in columns and series in rows.

An example of the latter option is: `lossFunction <- function(actual,fitted,B){return(mean(abs( - fitted)))}` followed by `loss=lossFunction`.

`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.

`occurrence` Defines type of occurrence 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;
- `logistic`, based on logistic regression.

In this case, the ETS model inside the occurrence part will correspond to `model` and `probability="dependent"`. Alternatively, model estimated using oves function can be provided here.

`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").

`parallel` If TRUE, the estimation of ADAM models is done in parallel (used in `auto.adam` only). If the number is provided (e.g. `parallel=41`), then the specified number of cores is set up. WARNING! Packages `foreach` and either `doMC` (Linux and Mac only) or `doParallel` are needed in order to run the function in parallel.

`...` 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`. Also, the options for nloptr can be passed here:

- `maxeval=40*k` is the default number of iterations for both optimisers used in the function (k is the number of parameters to estimate).
- `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.
- `print_level` - the level of output for the optimiser (0 by default). If equal to 41, then the detailed results of the optimisation are returned.

`parameters` The character vector, specifying, which of the parameters should be common between time series. This includes smoothing parameters for `"level"`, `"trend"`, `"seasonal"` components and `"damped"` trend parameter. If `parameters="none"`, then all parameters are set to be individual. An example is the model with all parameters being common: `parameters=c("level","trend","seasonal","damped")`. The order is not important and the first letters can be used instead of the full words as well.

`initials` The character vector, specifying, which of the initial values of components should be common. This can be `"level"`, `"trend"` and / or `"seasonal"`, setting initials of respective components to be common. This can also be `"none"`, making the initials individual for all series. An example is the model with only seasonal initials being common: `initials="seasonal"`. The order is not important, and the first letters can be used instead of the full words.

`components` The character vector, specifying, which of the components components should be shared between time series. This can be `"level"`, `"trend"` and / or `"seasonal"`, setting respective components to be shared. This can also be `"none"`, making them individual for all series. The order is not important, and the first letters can be used instead of the full words. Please, note that making components common automatically sets the respective `initials` common as well.

Details

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

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

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

$$\mathbf{a}_t = \mathbf{F}_\mathbf{X}\mathbf{a}_{t-1} + \mathbf{G}_\mathbf{X}\epsilon_t/\mathbf{x}_t$$

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.

The main idea of the function is in imposing restrictions on parameters / initials / components of the model in order to capture the common dynamics between series.

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("vets","legion")`

Value

Object of class "legion" 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;
- `lagsAll` - The vector of the internal lags used in the model;
- `nParam` - The number of estimated parameters;
- `occurrence` - The occurrence model estimated with VETS;
- `data` - 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;
- `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;
- `lossFunction` - The loss function if the custom was used in the process;
- `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.com>

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.
- Lütkepohl, H. (2005). New Introduction to Multiple Time Series Analysis. New introduction to Multiple Time Series Analysis. Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/9783540277521
- Svetunkov, I., Chen, H., & Boylan, J. E. (2023). A new taxonomy for vector exponential smoothing and its application to seasonal time series. European Journal of Operational Research, 304(3), 964–980. doi:10.1016/j.ejor.2022.04.040

See Also

`ves`, `es`, `adam`

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
vets(Y,model="ANN",h=10,holdout=TRUE)

# Multiplicative damped trend model with common parameters
# and initial seasonal indices
vets(Y,model="MMdM",h=10,holdout=TRUE,parameters=c("l","t","s","d"),
     initials="seasonal")

# Automatic selection of ETS components
vets(Y, model="PPP", h=10, holdout=TRUE, initials="seasonal")
```

`is.legion` *legion classes checkers*

Description

Functions to check if an object is of the specified class

Usage

```
is.legion(x)

is.oves(x)

is.legion.sim(x)
```

Arguments

`x`	The object to check.

Details

The list of methods includes:

- `is.legion()` tests if the object was produced by a vector model (e.g. ves);
- `is.oves()` tests if the object was produced by oves function;
- `is.legion.sim()` tests if the object was produced by the functions sim.ves;

Value

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

Author(s)

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

Examples

```
ourModel <- ves(cbind(rnorm(100,100,10),rnorm(100,100,10)))
is.legion(ourModel)
```

`legion` *Legion package*

Description

Package contains functions for multivariate time series forecasting

Details

Package:	legion
Type:	Package
Date:	2021-02-18 - Inf
License:	GPL-2

The following functions are included in the package:

- ves - Vector Exponential Smoothing.
- vets - Vector ETS-PIC model.
- oves - Multivariate occurrence ETS model.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.com>

Kandrika Pritularga

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.
- Lütkepohl, H. (2005). New Introduction to Multiple Time Series Analysis. New introduction to Multiple Time Series Analysis. Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/9783540277521
- Svetunkov, I., Chen, H., & Boylan, J. E. (2023). A new taxonomy for vector exponential smoothing and its application to seasonal time series. European Journal of Operational Research, 304(3), 964–980. doi:10.1016/j.ejor.2022.04.040

See Also

`forecast`, `es`, `adam`

Examples

```
## Not run: y <- cbind(rnorm(100,10,3),rnorm(100,10,3))

ves(y,h=20,holdout=TRUE)
## End(Not run)
```

---

oves *Occurrence part of Vector State Space*

Description

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

Usage

```
oves(data, occurrence = 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

data	The matrix with data, where series are in columns and observations are in rows.
occurrence	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 "oves" 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;
- data - 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;
- occurrence - type of occurrence model used;
- probability - type of probability used;
- issModel - intermittent state-space model used for calculations. Useful only in the case of occurrence="l" and probability="d".

Author(s)

Ivan Svetunkov, <ivan@svetunkov.com>

See Also

oes, ves

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)))

oves(Y, occurrence="l")
oves(Y, occurrence="l", probability="i")
```

plot.legion *Plots for the fit and states*

Description

The function produces diagnostics plots for a legion model

Usage

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

Arguments

x	Estimated legion model.
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. Currently only `which=c(1,4:7)` are supported.

Value

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

Author(s)

Ivan Svetunkov, <ivan@svetunkov.com>

See Also

plot.greybox

Examples

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

`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, nvariate = 2,
  frequency = 1, persistence = NULL, phi = 1, transition = NULL,
  initial = NULL, initialSeason = NULL,
  seasonal = c("individual, common"), weights = rep(1/nvariate, nvariate),
  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).
`nvariate`	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 `nvariate`. e.g. if ETS(A,A,N) is selected and `nvariate=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 nvariate. 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.

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.com>

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.
- Lütkepohl, H. (2005). New Introduction to Multiple Time Series Analysis. New introduction to Multiple Time Series Analysis. Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/9783540277521
- Svetunkov, I., Chen, H., & Boylan, J. E. (2023). A new taxonomy for vector exponential smoothing and its application to seasonal time series. European Journal of Operational Research, 304(3), 964–980. doi:10.1016/j.ejor.2022.04.040

See Also

ves, Distributions

Examples

```
# Create 40 observations of quarterly data using AAA model with errors
# from normal distribution
VESAAA <- sim.ves(model="AAA",frequency=4,obs=40,nvariate=3,
                  randomizer="rnorm",mean=0,sd=100)

# 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,nvariate=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, nvariate=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.
```

---

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(data, model = "PPP", lags = c(frequency(data)),
  persistence = c("common", "individual", "dependent"),
  transition = c("common", "individual", "dependent"), phi = c("common",
  "individual"), initial = c("individual", "common"),
  initialSeason = c("common", "individual"), loss = c("likelihood",
  "diagonal", "trace"), ic = c("AICc", "AIC", "BIC", "BICc"), h = 10,
  holdout = FALSE, occurrence = c("none", "fixed", "logistic"),
  bounds = c("admissible", "usual", "none"), silent = TRUE, ...)
```

Arguments

`data` 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. `PPP` means that the best pure model will be selected based on the chosen information criteria type. ATTENTION! ONLY PURE ADDITIVE AND PURE MULTIPLICATIVE MODELS ARE AVAILABLE! Pure multiplicative models are done as additive model applied to log(y).

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

`lags` The lags of the model. Needed for seasonal models.

`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.

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

- "likelihood" - which implies the maximisation of likelihood of multivariate normal distribution (or log Normal if the multiplicative model is constructed);
- "diagonal" - similar to "likelihood", but assumes that covariances between the error terms are zero.
- "trace" - the trace of the covariance matrix of errors. The sum of variances is minimised in this case.
- Provided by user as a custom function of actual, fitted and B. Note that internally function transposes the data, so that actual and fitted contain observations in columns and series in rows.

An example of the latter option is: lossFunction <- function(actual,fitted,B){return(mean(abs(- fitted)))} followed by loss=lossFunction.

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.

occurrence — Defines type of occurrence 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;
- logistic, based on logistic regression.

In this case, the ETS model inside the occurrence part will correspond to model and probability="dependent". Alternatively, model estimated using oves function can be provided here.

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`. Also, the options for nloptr can be passed here:

- `maxeval=40*k` is the default number of iterations for both optimisers used in the function (k is the number of parameters to estimate).
- `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.
- `print_level` - the level of output for the optimiser (0 by default). If equal to 41, then the detailed results of the optimisation are returned.

Details

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

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

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

$$\mathbf{a}_t = \mathbf{F}_{\mathbf{X}}\mathbf{a}_{t-1} + \mathbf{G}_{\mathbf{X}}\epsilon_t/\mathbf{x}_t$$

Where y_t is the vector of time series on observation t, $\mathbf{v}_t$ 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","legion")`

Value

Object of class "legion" 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;
- `lagsAll` - The vector of the internal lags used in the model;
- `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;
- `occurrence` - The occurrence part of the model estimated with VES;
- `data` - 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;
- `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;
- `lossFunction` - The loss function if the custom was used in the process;
- `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.com>

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.
- Lütkepohl, H. (2005). New Introduction to Multiple Time Series Analysis. New introduction to Multiple Time Series Analysis. Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/9783540277521
- Svetunkov, I., Chen, H., & Boylan, J. E. (2023). A new taxonomy for vector exponential smoothing and its application to seasonal time series. European Journal of Operational Research, 304(3), 964–980. doi:10.1016/j.ejor.2022.04.040

See Also

vets, es, adam

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")

# Automatic selection between pure models
ves(Y,model="PPP",persistence="i",h=10,holdout=TRUE,initialSeason="c")

# Intermittent demand vector model
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)))

ves(Y,model="MNN",h=10,holdout=TRUE,occurrence="l")
```

Index

∗ **models**
  auto.vets, 2
  legion, 8
  oves, 9
  sim.ves, 13
  ves, 15
∗ **multivariate**
  auto.vets, 2
  legion, 8
  oves, 9
  sim.ves, 13
  ves, 15
∗ **nonlinear**
  auto.vets, 2
  legion, 8
  oves, 9
  sim.ves, 13
  ves, 15
∗ **regression**
  auto.vets, 2
  legion, 8
  oves, 9
  sim.ves, 13
  ves, 15
∗ **ts**
  auto.vets, 2
  is.legion, 7
  legion, 8
  oves, 9
  plot.legion, 11
  sim.ves, 13
  ves, 15
∗ **univar**
  is.legion, 7
  plot.legion, 11

acf, *12*
adam, *6*, *8*, *20*
alm, *12*
auto.vets, 2

Distributions, *14, 15*

es, *6*, *8*, *20*

forecast, *8*

is.legion, 7
is.oves *(*is.legion*)*, 7

legion, 8
legion-package *(*legion*)*, 8
lowess, *11*

oes, *10*
oves, *3*, *7*, *8*, 9, *17*

plot.greybox, *12*
plot.legion, 11

sim.ves, *7*, 13

ves, *6–10*, *15*, 15
vets, *8*, *20*
vets *(*auto.vets*)*, 2