Package: fEGarch (via r-universe)

October 10, 2026

Type Package

Title SM/LM EGARCH & GARCH, VaR/ES Backtesting & Dual LM Extensions

Version 1.0.6

Maintainer Dominik Schulz <dominik.schulz@uni-paderborn.de>

Description Implement and fit a variety of short-memory (SM) and long-memory (LM) models from a very broad family of exponential generalized autoregressive conditional heteroskedasticity (EGARCH) models, such as a MEGARCH (modified EGARCH), FIEGARCH (fractionally integrated EGARCH), FIMLog-GARCH (fractionally integrated modulus Log-GARCH), and more. The FIMLog-GARCH as part of the EGARCH family is discussed in Feng et al. (2023) <https://econpapers.repec.org/paper/pdnciepap/156.htm>. For convenience and the purpose of comparison, a variety of other popular SM and LM GARCH-type models, like an APARCH model, a fractionally integrated APARCH (FIAPARCH) model, standard GARCH and fractionally integrated GARCH (FIGARCH) models, GJR-GARCH and FIGJR-GARCH models, TGARCH and FITGARCH models, are implemented as well as dual models with simultaneous modelling of the mean, including dual long-memory models with a fractionally integrated autoregressive moving average (FARIMA) model in the mean and a long-memory model in the variance, and semiparametric volatility model extensions. Parametric models and parametric model parts are fitted through quasi-maximum-likelihood estimation. Furthermore, common forecasting and backtesting functions for value-at-risk (VaR) and expected shortfall (ES) based on the package's models are provided.

License GPL-3

LinkingTo Rcpp, RcppArmadillo

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Collate 'AttachMessage.R' 'RcppExports.R' 'close_to_lreturn.R' 'lin_filters.R' 'hessCalc.R' 'arma-farima-wrappers.R' 'format_applier_ts.R' 'ts_split_train_and_test.R' 'snorm-distribution-functions.R' 'sstd-distribution-functions.R' 'sged-distribution-functions.R' 'sald-distribution-functions.R' 'base_sim_functions.R' 'density_selectors.R' 'generics.R' 'input_checkers_egarch_spec.R' 'input_checkers_mean_spec.R' 'input_checkers_nonpar_spec.R' 'class-mean_spec.R' 'class-locpol_spec.R' 'class-fEGarch_fit.R' 'class-egarch-spec.R' 'fitting-function.R' 'sim-functions.R' 'nonparametric-step.R' 'setup-estim.R' 'general_garch_fitting.R' 'aparchfit.R' 'gjrgarchfit.R' 'tgarchfit.R' 'garchfit.R' 'fiaparchfit.R' 'figjrgarchfit.R' 'fitgarchfit.R' 'figarchfit.R' 'garch_estim.R' 'varescalc.R' 'datasets.R' 'fEGarch-package.R' 'rugarch-wrappers.R' 'class-fEGarch_forecast.R' 'forecasting-functions.R' 'fEGarch_fit-plot.R' 'class-fEGarch_risk.R' 'ufRisk-functions.R' 'reexport-pipe.R' 'test-functions.R' 'class-fEGarch_distr_est.R' 'distr_est.R' 'popular-methods.R'

Depends R (>= 3.5), methods

Imports Rcpp (>= 1.0.9), Rsolnp, smoots, esemifar, zoo, stats, utils, rugarch, future, furrr, rlang, ggplot2, magrittr, cli, numDeriv

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation yes

Author Dominik Schulz [aut, cre] (Paderborn University, Germany), Yuanhua Feng [aut] (Paderborn University, Germany), Christian Peitz [aut] (Financial Intelligence Unit (German Government)), Oliver Kojo Ayensu [aut] (Paderborn University, Germany), Thomas Gries [ctb] (Paderborn University, Germany), Sikandar Siddiqui [ctb] (Deloitte Audit Analytics GmbH, Frankfurt, Germany), Shujie Li [ctb] (Paderborn University, Germany)

Repository https://cran.r-universe.dev

Date/Publication 2026-02-10 09:40:02 UTC

RemoteUrl https://github.com/cran/fEGarch

RemoteRef HEAD

RemoteSha 80deeb1d7034f90e0a3d83929e672a2918e0390b

Contents

fEGarch-package 4
accessor_methods 7
aparch 9

aparch_sim . . . 14
autoplot,fEGarch_fit-method . . . 15
autoplot,fEGarch_risk-method . . . 16
close_to_lreturn . . . 17
distr_est . . . 18
egarch_type_spec . . . 21
fEGarch . . . 22
fEGarch_sim . . . 28
fEGarch_spec . . . 29
fiaparch . . . 33
fiaparch_sim . . . 39
figarch . . . 40
figarch_sim . . . 45
figjrgarch . . . 47
figjrgarch_sim . . . 52
find_dist . . . 54
fit_test_suite,fEGarch_fit-method . . . 55
fitgarch . . . 56
fitgarch_sim . . . 62
fitted,fEGarch_fit-method . . . 63
fitted_object_generics . . . 64
garch . . . 65
garch_sim . . . 69
garchm_estim . . . 71
gjrgarch . . . 74
gjrgarch_sim . . . 79
goodn_of_fit_test,fEGarch_fit-method . . . 80
inf_criteria,fEGarch_distr_est-method . . . 81
ljung_box_test . . . 82
ljung_box_test,fEGarch_fit-method . . . 84
locpol_spec . . . 85
locpol_spec_methods . . . 87
loss_functions . . . 88
loss_functions,fEGarch_risk-method . . . 88
mean_spec . . . 90
mean_spec_methods . . . 91
measure_risk . . . 92
plot . . . 95
plot,fEGarch_fit,ANY-method . . . 95
plot,fEGarch_risk,ANY-method . . . 96
poly_order . . . 97
predict,fEGarch_fit-method . . . 98
predict_roll . . . 100
residuals,fEGarch_fit-method . . . 101
rugarch_wrappers . . . 102
show,fEGarch_fit_egarch-method . . . 105
sigma,fEGarch_fit-method . . . 106
sign_bias_test,fEGarch_fit-method . . . 107

sim_functions 108
SP500 109
spec_generics 109
spec_methods 110
submodel-specs 112
tgarch 114
tgarch_sim 119
trafflight_test 120
trafflight_test,fEGarch_risk-method 121
UKinflation 124
VaR_calc 124

Index **127**

`fEGarch-package` *Estimation of a Broad Family of EGARCH Models*

Description

A library of quasi maximum-likelihood estimation (QMLE) methods for fitting various short- and long-memory models from a broad family of exponential generalized autoregressive conditional heteroskedasticity (EGARCH) models. For the purpose of comparison, a FIAPARCH (fractionally integrated asymmetric power ARCH), a FIGARCH (fractionally integrated GARCH), a FITGARCH, a FIGJR-GARCH and their short-memory variants can be implemented as well.

Details

`fEGarch` is an R package for estimating a broad family of EGARCH models (Feng et al., 2025; Ayensu et al., 2025) including both short- and long-memory as well as a selection of varying transformations for the asymmetry and the magnitude term in such a model, for example in form of the FIMLog-GARCH (Feng et al., 2023). Log-GARCH specifications can be implemented as well as a special case of the broad EGARCH family. The six most common conditional distributions are supported, namely a normal distribution, a t- distribution, a generalized error distribution, as well as the skewed variants of these three distributions. Furthermore, as a novelty, an average Laplace (AL) distribution (see for example Feng et al., 2025) and its skewed version are provided as well. The main functions to implement these models are `fEGarch_spec` in combination with `fEGarch`. Further details on these models can also be found in the documentation of these two functions. For convenience, further specification functions for particular submodels are available as well: `egarch_spec`, `loggarch_spec`, `megarch_spec`, `mloggarch_spec`, `fiegarch_spec`, `filoggarch_spec`, `fimegarch_spec` and `fimloggarch_spec`.

As a popular alternative for the sake of comparison, a FIAPARCH model can be fitted as well using `fiaparch`. The corresponding documentation page also includes further information on the model. Similarly, `figarch` can be utilized for fitting FIGARCH models, `fitgarch` for fitting FITGARCH models and `figjrgarch` for fitting FIGJR-GARCH models. A general function for the estimation of additional GARCH-type models, including the aforementioned additional models as well as their short-memory variants, is `garchm_estim`.

In addition, the package provides functionalities in order to simultaneously model the conditional mean (using either autoregressive moving-average (ARMA) models or fractionally integrated ARMA

(FARIMA) models) alongside the conditional variance. For this purpose, the function mean_spec can be utilized and its result needs to be passed to fEGarch alongside the result of either fEGarch_spec or one of its wrappers.

Further options include the specification of semiparametric volatility models (see also Ayensu et al., 2025), where a smooth, nonparametric scale function is at first estimated and removed from an observed series, before estimating a parametric model. The scale estimation is currently done through automated local polynomial regression with designated bandwidth selection algorithms under short memory and long memory.

Main Functions

The main functions of the package are:

fEGarch_spec: setting the model specifications for a model from the broader EGARCH family,

mean_spec: setting the model specifications for the conditional mean,

fEGarch: fitting a model from the broad family of EGARCH models given a model specification and an observation series,

garchm_estim: fitting a GARCH-type model selectable from a standard GARCH, a GJR-GARCH, a TGARCH, an APARCH, a FIGARCH, a FIGJR-GARCH, a FITGARCH and a FIAPARCH,

fEGarch_sim: simulating from an EGARCH family model,

fiaparch_sim: simulating from a FIAPARCH model,

figarch_sim: simulating from a FIGARCH model,

figjrgarch_sim: simulating from a FIGJR-GARCH model,

fitgarch_sim: simulating from a FITGARCH model,

aparch_sim: simulating from an APARCH model,

garch_sim: simulating from a GARCH model,

gjrgarch_sim: simulating from a GJR-GARCH model,

tgarch_sim: simulating from a TGARCH model,

predict,fEGarch_fit-method: multistep point forecasts of the conditional mean and the conditional standard deviation,

predict_roll,fEGarch_fit-method: rolling point forecasts of the conditional mean and the conditional standard deviation over a test set.

measure_risk: value at risk and expected shortfall computation for various model specifications.

find_dist: fits all eight distributions considered in this package to a supposed iid series and selects the best fitted distribution following either BIC (the default) or AIC.

backtest_suite,fEGarch_risk-method: runs a selection of functions for backtesting VaR and ES.

Datasets

The package includes a few datasets. Follow the corresponding links to the documentation of the datasets to find additional information including the sources.

UKinflation: monthly inflation rate of the UK.

SP500: daily log-returns of the S&P 500 index.

License

The package is distributed under the General Public License v3 ([GPL-3](https://tldrlegal.com/license/gnu-general-public-license-v3-(gpl-3))).

Author(s)

- Dominik Schulz (Department of Economics, Paderborn University), Author and Package Creator
- Yuanhua Feng (Department of Economics, Paderborn University), Author
- Christian Peitz (Financial Intelligence Unit, German Government), Author
- Oliver Kojo Ayensu (Department of Economics, Paderborn University), Author

References

- Ayensu, O. K., Feng, Y., & Schulz, D. (2025). Recent Extensions of Exponential GARCH Models: Theory and Application. Forthcoming preprint, Paderborn University.
- Baillie, R., Bollerslev, T., & Mikkelsen, H. O. (1996). Fractionally integrated generalized autoregressive conditional heteroskedasticity. Journal of Econometrics, 74(1), 3-30. DOI: 10.1016/S0304-4076(95)01749-6.
- Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. Journal of Econometrics, 31(3): 307-327. DOI: 10.1016/0304-4076(86)90063-1.
- Bollerslev, T., & Mikkelsen, H. O. (1996). Modeling and pricing long memory in stock market volatility. Journal of Econometrics, 73(1), 151–184. DOI: 10.1016/0304-4076(95)01749-6.
- Conrad, C., & Haag, B. R. (2006). Inequality constraints in the fractionally integrated GARCH model. Journal of Financial Econometrics, 4(3): 413-449. DOI: 10.1093/jjfinec/nbj015.
- Conrad, C., & Karanasos, M. (2006). The impulse response function of the long memory GARCH process. Economics Letters, 90(1): 34-41. DOI: 10.1016/j.econlet.2005.07.001.
- Ding, Z., Granger, C. W. J., & Engle, R. F. (1993). A long memory property of stock market returns and a new model. Journal of Empirical Finance, 1(1): 83-106. DOI: 10.1016/0927-5398(93)90006-D.
- Engle, R. F. (1982). Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation. Econometrica, 50(4): 987-1007. DOI: 10.2307/1912773.
- Feng, Y., Beran, J., Ghosh, S., & Letmathe, S. (2020). Fractionally integrated Log-GARCH with application to value at risk and expected shortfall. Working Papers CIE No. 137, Paderborn University, Center for International Economics. URL: http://groups.uni-paderborn.de/wp-wiwi/RePEc/pdf/ciepap/WP137.pdf.
- Feng, Y., Gries, T., Letmathe, S., & Schulz, D. (2022). The smoots Package in R for Semiparametric Modeling of Trend Stationary Time Series. The R Journal, 14(1), 182-195. URL: https://journal.r-project.org/articles/RJ-2022-017/.
- Feng, Y., Gries, T., & Letmathe, S. (2023). FIEGARCH, modulus asymmetric FILog-GARCH and trend-stationary dual long memory time series. Working Papers CIE No. 156, Paderborn University. URL: https://econpapers.repec.org/paper/pdnciepap/156.htm.

- Feng, Y., Peitz, C., & Siddiqui, S. (2025). A few useful members of the EGARCH-family with short- or long-memory in volatility. Unpublished working paper at Paderborn University.
- Geweke, J. (1986). Modeling the persistence of conditional variances: A comment. Econometric Reviews, 5(1), 57-61. DOI: 10.1080/07474938608800088.
- Glosten, L. R., Jagannathan, R., & Runkle, D. E. (1993). On The Relation between The Expected Value and The Volatility of Nominal Excess Return on stocks. Journal of Finance 48(5), 1779-1801. DOI: 10.1111/j.1540-6261.1993.tb05128.x.
- Karanasos, M., Psaradakis, Z., & Sola, M. (2004). On the autocorrelation properties of long-memory GARCH processes. Journal of Time Series Analysis, 25(2): 265-281. DOI: 10.1046/j.0143-9782.2003.00349.x.
- Letmathe, S., Beran, J., & Feng, Y. (2023). An extended exponential SEMIFAR model with application in R. Communications in Statistics - Theory and Methods, 53(22), 7914–7926. DOI: 10.1080/03610926.2023.2276049.
- Milhoj, A. (1987). A Multiplicative Parameterization of ARCH Models. University of Copenhagen, Denmark.
- Missiakoulis, S. (1983). Sargan Densities: Which One?. Journal of Econometrics, 23(2): 223-233. DOI: 10.1016/0304-4076(93)90078-J
- Nelson, D. B. (1991). Conditional Heteroskedasticity in Asset Returns: A New Approach. Econometrica, 59(2), 347–370. DOI: 10.2307/2938260.
- Nielsen, M. O., & Noel, A. L. (2021). To infinity and beyond: Efficient computation of ARCH(∞) models. Journal of Time Series Analysis, 42(3), 338–354. DOI: 10.1111/jtsa.12570.
- Pantula, S. G. (1986). Modeling the persistence of conditional variances: A comment. Econometric Reviews, 5(1), 71-74. DOI: 10.1080/07474938608800089.
- Tse, Y. K. (1987). A Note On Sargan Densities. Journal of Econometrics, 34(3): 349-354. DOI: 10.1016/0304-4076(87)90017-0
- Tse, Y. K. (1998). The conditional heteroskedasticity of the yen-dollar exchange rate. Journal of Applied Econometrics, 13(1): 49-55. DOI: 10.1002/(SICI)1099-1255(199801/02)13:1<49::AID-JAE459>3.0.CO;2-O.
- Zakoian, J.-M. (1994). Threshold heteroskedastic models. Journal of Economic Dynamics and Control, 18(5): 931-955. DOI: 10.1016/0165-1889(94)90039-6.

`accessor_methods` *Methods for Accessing Model Estimation and Forecasting Output Elements*

Description

Accessors to access the elements of the same name in output objects returned by either `fEGarch`, `garchm_estim`, `predict` or `predict_roll`.

Usage

```
## S4 method for signature 'fEGarch_fit'
sigt(x)

## S4 method for signature 'fEGarch_fit'
cmeans(x)

## S4 method for signature 'fEGarch_fit'
etat(x)

## S4 method for signature 'fEGarch_fit'
inf_criteria(x)

## S4 method for signature 'fEGarch_fit'
llhood(x)

## S4 method for signature 'fEGarch_fit'
pars(x)

## S4 method for signature 'fEGarch_fit'
se(x)

## S4 method for signature 'fEGarch_fit'
vcov_mat(x)

## S4 method for signature 'fEGarch_forecast'
sigt(x)

## S4 method for signature 'fEGarch_forecast'
cmeans(x)
```

Arguments

x	an object returned by either fEGarch, garchm_estim, predict (for sigt and cmeans) or predict_roll (for sigt and cmeans).

Details

Convenience methods to access the elements of the same name that can otherwise be accessed via the operator @ within objects that inherit from class "fEGarch_fit", which covers objects returned by either fEGarch, garchm_estim, predict (for sigt and cmeans) or predict_roll (for sigt and cmeans).

As alternatives for sigt, cmeans and etat, see also sigma,fEGarch_fit-method, fitted,fEGarch_fit-method and residuals,fEGarch_fit-method.

Value

The element within the input object of the same name as the method is returned. Depending on the element that can be a numeric vector, an object of class "zoo" or a numeric matrix.

Examples

```
window.zoo <- get("window.zoo", envir = asNamespace("zoo"))
rt <- window.zoo(SP500, start = "2010-01-01", end = "2012-12-31")
est <- fEGarch(egarch_spec(), rt)

# Access estimated conditional standard deviations using
# the common operator "@" ...
sigt1 <- est@sigt

# ... or use the accessor method "sigt()"
sigt2 <- sigt(est)

zoo::plot.zoo(
 cbind("Approach 1" = sigt1, "Approach 2" = sigt2)
)

# Other methods
cmeans(est)
etat(est)
inf_criteria(est)
llhood(est)
pars(est)
se(est)
vcov_mat(est)
```

---

aparch *APARCH Model Fitting*

Description

Fit an APARCH model under the six most common and further conditional distributions to observed data using quasi maximum-likelihood estimation.

Usage

```
aparch(
  rt,
  orders = c(1, 1),
  cond_dist = c("norm", "std", "ged", "ald", "snorm", "sstd", "sged", "sald"),
  meanspec = mean_spec(),
  Drange = c(0, 1),
  nonparspec = locpol_spec(),
  use_nonpar = FALSE,
```

```
  n_test = 0,
  start_pars = NULL,
  LB = NULL,
  UB = NULL,
  control = list(),
  control_nonpar = list(),
  mean_after_nonpar = FALSE,
  parallel = TRUE,
  ncores = max(1, future::availableCores() - 1),
  trunc = "none",
  presample = 50,
  Prange = c(1, 5),
  fix_delta = c(NA, 1, 2),
  skip_vcov = FALSE
)
```

Arguments

rt	the observed series ordered from past to present; can be a numeric vector, a "zoo" class time series object, or a "ts" class time series object.
orders	a two-element numeric vector containing the two model orders p and q (see Details for more information); currently, only the default orders = c(1, 1) is supported; other specifications of a two-element numeric vector will lead to orders = c(1, 1) being run and a warning message being returned.
cond_dist	the conditional distribution to consider as a character object; the default is a conditional normal distribution "norm"; available are also, however, a t-distribution ("std"), a generalized error distribution ("ged"), an average Laplace distribution ("ald"), and their four skewed variants ("snorm", "sstd", "sged", "sald").
meanspec	an object of class "mean_spec"; indicates the specifications for the model in the conditional mean.
Drange	a two-element numeric vector that indicates the boundaries of the interval over which to search for the fractional differencing parameter D in a long-memory ARMA-type model in the conditional mean model part; by default, D being searched for on the interval from 0 to $0.5 - 1 \times 10^{-6}$; note that specific settings in the arguments LB and UB overwrite this argument.
nonparspec	an object of class "locpol_spec" returned by locpol_spec; defines the settings of the nonparametric smoothing technique for use_nonpar = TRUE.
use_nonpar	a logical indicating whether or not to implement a semiparametric extension of the volatility model defined through spec; see "Details" for more information.
n_test	a single numerical value indicating, how many observations at the end of rt not to include in the fitting process and to reserve for backtesting.
start_pars	the starting parameters for the numerical optimization routine; should be of the same length as the parameter output vector within the output object (also keeping the same order); for NULL, an internally saved default set of values is used; see "Details" for the order of elements; elements should be set with respect to a series rescaled to have sample variance one.

LB	the lower boundaries of the parameters in the numerical optimization routine; should be of the same length as the parameter output vector within the output object (also keeping the same order); for NULL, an internally saved default set of values is used; see "Details" for the order of elements; elements should be set with respect to a series rescaled to have sample variance one.
UB	the upper boundaries of the parameters in the numerical optimization routine; should be of the same length as the parameter output vector within the output object (also keeping the same order); for NULL, an internally saved default set of values is used; see "Details" for the order of elements; elements should be set with respect to a series rescaled to have sample variance one.
control	a list that is passed to control of the function solnp of the package Rsolnp.
control_nonpar	a list containing changes to the arguments for the hyperparameter estimation algorithm in the nonparametric scale function estimation for use_nonpar = TRUE; see "Details" for more information.
mean_after_nonpar	only for use_nonpar = TRUE; considers the unconditional mean of the parametric model part in the QMLE step in a semiparametric model; by default, a zero-mean model is considered for the parametric part in a semiparametric model.
parallel	only relevant for a (skewed) average Laplace (AL) distribution, i.e. if cond_dist in spec is set to cond_dist = "ald" or cond_dist = "sald"; parallel is a logical value indicating whether or not the slices for the positive integer-valued parameter of the SM distribution should be fitted in parallel for a speed boost.
ncores	only relevant for a (skewed) average Laplace (AL) distribution, i.e. if cond_dist in spec is set to cond_dist = "ald" or cond_dist = "sald", and if simultaneously parallel = TRUE; ncores is a single numeric value indicating the number of cores to use for parallel computations.
trunc	a positive integer indicating the finite truncation length of the infinite-order polynomials of the infinite-order representations of the long-memory model parts; the character "none" is an optional input that specifies that truncation should always be applied back to the first (presample) observation time point, i.e. that maximum length filters should be applied at all times.
presample	the presample length for initialization (for extended EGARCH- / Log-GARCH-type models only relevant for the FARIMA-part, as series in log-transformed conditional variance are initialized by zero).
Prange	a two-element vector that indicates the search boundaries for the parameter P in a (skewed) average Laplace distribution.
fix_delta	let the parameter δ be either free (fix_delta = NA), fix it to 1 (fix_delta = 1) or to 2 (fix_delta = 2); the latter two are specific submodels of a FIAPARCH.
skip_vcov	a logical indicating whether or not to skip the computation of the variance-covariance matrix of the parameter estimators and therefore also standard error computation.

Details

Let $\{r_t\}$, with $t \in \mathbb{Z}$ as the time index, be a theoretical time series that follows

$$r_t = \mu + \varepsilon_t \text{ with } \varepsilon_t = \sigma_t \eta_t \text{ and } \eta_t \sim \text{IID}(0,1), \text{ where}$$

$$\sigma_t^\delta = \omega + \sum_{i=1}^{p} \phi_i \left(|\varepsilon_{t-i}| - \gamma_i \varepsilon_{t-i} \right)^\delta + \sum_{j=1}^{j} \beta_j \sigma_{t-j}^\delta .$$

Here, $\eta_t \sim \text{IID}(0,1)$ means that the innovations η_t are independent and identically distributed (iid) with mean zero and variance one, whereas $\sigma_t > 0$ are the conditional standard deviations in r_t. Moreover, β_j, $j = 1, 2, \ldots, q$, ϕ_i, $i = 1, 2, \ldots, p$, are real-valued coefficients. p and q are the model orders definable through the argument `orders`, where p is the first element and q is the second element in the argument. In addition, we have $\mu = E(r_t)$ as a real-valued parameter, and $\gamma \in (-1, 1)$. $\omega > 0$ is the intercept. It is assumed that all β_j and ϕ_i are non-negative.

See also the reference section for sources on the APARCH (Ding et al., 1993) and the FIAPARCH (Tse, 1998) models.

In the current package version, standard errors of parameter estimates are computed from the Hessian at the optimum of the log-likelihood using `hessian`. To ensure numerical stability and applicability to a huge variety of differently scaled data, parametric models are first fitted to data that is scaled to have sample variance 1. Parameter estimates and other quantities are then either retransformed or recalculated afterwards for the original data.

For a conditional average Laplace distribution, an optimal model for each distribution parameter P from 1 to 5 is estimated (assuming that P is then fixed to the corresponding value). Afterwards, P is then estimated by selecting the estimated model among the five fitted models that has the largest log-likelihood. The five models are, by default, fitted simultaneously using parallel programming techniques (see also the arguments `parallel` and `ncores`, which are only relevant for a conditional average Laplace distribution). After the optimal model (including the estimate of P called $\hat{P}$) has been determined, $P = \hat{P}$ is seen as fixed to obtain the standard errors via the Hessian matrix for the estimates of the continuous parameters. A standard error for $\hat{P}$ is therefore not obtained and the ones obtained for the remaining estimates do not account for $\hat{P}$.

An ARMA-APARCH or a FARIMA-APARCH can be fitted by adjusting the argument `meanspec` correspondingly.

As an alternative, a semiparametric extension of the pure models in the conditional variance can be implemented. If `use_nonpar = TRUE`, `meanspec` is omitted and before fitting a model in the conditional volatility following the remaining function arguments, a smooth scale function, i.e. a function representing the unconditional standard deviation over time, is being estimated following the specifications in `nonparspec` and `control_nonpar`. This preliminary step stabilizes the input series `rt`, as long-term changes in the unconditional variance are being estimated and removed before the parametric step using `tsmooth`. `control_nonpar` can be adjusted following to make changes to the arguments of `tsmooth` for short-memory specifications. These arguments specify settings for the automated bandwidth selection algorithms implemented by this function. By default, we use the settings `InfR = "Nai"`, `bStart = 0.15`, `cb = 0.05`, and `method = "lpr"` for `tsmooth`. `locpol_spec` passed to `nonparspec` handles more direct settings of the local polynomial smoother itself. See the documentation for these functions to get a detailed overview of these settings. Assume $\{r_t\}$ to be the observed series, where $t = 1, 2, \ldots, n$, then $r_t^* = r_t - \bar{r}$, with $\bar{r}$ being the arithmetic mean over the observed r_t, is computed and subsequently $y_t = \ln\left[(r_t^*)^2\right]$. The subtraction of $\bar{r}$ is necessary so that r_t^* are all different from zero almost surely. Once y_t are available, its trend $m(x_t)$, with x_t as the rescaled time on the interval $[0, 1]$, is being estimated using `tsmooth` and denoted here by $\hat{m}(x_t)$. Then from $\hat{\xi}_t = y_t - \hat{m}(x_t)$ obtain $\hat{C} = -\ln\left\{\sum_{t=1}^{n} \exp\left(\hat{\xi}_t\right)\right\}$, and obtain the estimated scale function as $\hat{s}(x_t) = \exp\left[\left(\hat{\mu}(x_t) - \hat{C}\right)/2\right]$. The stabilized / standardized version of

the series $\{r_t\}$ is then $\tilde{r}_t = r_t^*/\hat{s}(x_t)$, to which a purely parametric volatility model following the remaining function arguments is then fitted. The estimated volatility at a given time point is then the product of the estimate of the corresponding scale function value and of the estimated conditional standard deviation (following the parametric model part) for that same time point. See for example Feng et al. (2022) or Letmathe et al. (2023) for more information on the semiparametric extension of volatility models.

The order for manual settings of `start_pars`, `LB` and `UB` is crucial. The correct order is: μ, $\text{ar}_1, \ldots, \text{ar}_{p^*}$, $\text{ma}_1, \ldots, \text{ma}_{q^*}$,$D$,$\omega$, $\phi_1, \ldots, \phi_p$, $\beta_1, \ldots, \beta_q$, $\gamma_1, \ldots, \gamma_p$, δ, shape parameter, skewness parameter. Depending on the exact model specification, parameters irrelevant for the specification at hand should be dropped in `start_pars`, `LB` and `UB`.

Value

An object of S4-class `"fEGarch_fit_aparch"` is returned. It contains the following elements.

`pars`: a named numeric vector with the parameter estimates.

`se`: a named numeric vector with the obtained standard errors in accordance with the parameter estimates.

`vcov_mat`: the variance-covariance matrix of the parameter estimates with named columns and rows.

`rt`: the input object `rt` (or at least the training data, if `n_test` is greater than zero); if `rt` was a `"zoo"` or `"ts"` object, the formatting is kept.

`cmeans`: the estimated conditional means; if `rt` was a `"zoo"` or `"ts"` object, the formatting is also applied to `cmeans`.

`sigt`: the estimated conditional standard deviations (or for `use_nonpar = TRUE` the estimated total volatilities, i.e. scale function value times conditional standard deviation); if `rt` was a `"zoo"` or `"ts"` object, the formatting is also applied to `sigt`.

`etat`: the obtained residuals; if `rt` was a `"zoo"` or `"ts"` object, the formatting is also applied to `etat`.

`orders`: a two-element numeric vector stating the considered model orders.

`cond_dist`: a character value stating the conditional distribution considered in the model fitting.

`long_memo`: a logical value stating whether or not long memory was considered in the model fitting.

`llhood`: the log-likelihood value obtained at the optimal parameter combination.

`inf_criteria`: a named two-element numeric vector with the corresponding AIC (first element) and BIC (second element) of the fitted parametric model part; for purely parametric models, these criteria are valid for the entire model; for semiparametric models, they are only valid for the parametric step and are not valid for the entire model.

`meanspec`: the settings for the model in the conditional mean; is an object of class `"mean_spec"` that is identical to the object passed to the input argument `meanspec`.

`test_obs`: the observations at the end up the input `rt` reserved for testing following `n_test`.

`scale_fun`: the estimated scale function values, if `use_nonpar = TRUE`, otherwise `NULL`; formatting of `rt` is reused.

`nonpar_model`: the estimation object returned by `tsmooth` for `use_nonpar = TRUE`.

`trunc`: the input argument `trunc`.

References

- Ding, Z., Granger, C. W. J., & Engle, R. F. (1993). A long memory property of stock market returns and a new model. Journal of Empirical Finance, 1(1): 83-106. DOI: 10.1016/0927-5398(93)90006-D.
- Feng, Y., Gries, T., Letmathe, S., & Schulz, D. (2022). The smoots Package in R for Semi-parametric Modeling of Trend Stationary Time Series. The R Journal, 14(1), 182-195. URL: https://journal.r-project.org/articles/RJ-2022-017/.
- Letmathe, S., Beran, J., & Feng, Y. (2023). An extended exponential SEMIFAR model with application in R. Communications in Statistics - Theory and Methods, 53(22), 7914–7926. DOI: 10.1080/03610926.2023.2276049.
- Tse, Y. K. (1998). The conditional heteroskedasticity of the yen-dollar exchange rate. Journal of Applied Econometrics, 13(1): 49-55. DOI: 10.1002/(SICI)1099-1255(199801/02)13:1<49::AID-JAE459>3.0.CO;2-O.

Examples

```
window.zoo <- get("window.zoo", envir = asNamespace("zoo"))
rt <- window.zoo(SP500, end = "2002-12-31")
model <- aparch(rt)
model
```

`aparch_sim` *Simulate From APARCH Models*

Description

A streamlined simulation function to simulate from asymmetric power autoregressive conditional heteroskedasticity (APARCH) models.

Usage

```
aparch_sim(
 pars = list(mu = 0, ar = numeric(0), ma = numeric(0), D = 0, omega = 4e-04, phi = 0.05,
    beta = 0.8, gamma = 0.1, delta = 2, df = 10, shape = 2, P = 3, skew = 1),
  cond_dist = c("norm", "std", "ged", "ald", "snorm", "sstd", "sged", "sald"),
  n = 1000,
  nstart = 5000,
  trunc = "none"
)
```

Arguments

`pars`	a named list with the parameter specifications; the user can provide a named list with only the settings they would like to adjust relative to the default settings.

cond_dist	a one-element character vector specifying the conditional distribution to consider.
n	the number of observations to return.
nstart	the number of burn-in observations to simulate before the final n values to keep; the first nstart values are not returned; if a dual model, i.e. with model in the conditional mean and in the conditional variance, is considered, two times nstart is considered in the first simulation step in the conditional variance, so that n + nstart values can be fed into the second simulation step for the conditional mean.
trunc	a truncation for the finite-order coefficient series in long-memory models; can either be the character "none" for truncation back to the very first observation at each time point, or to any positive integer for setting the corresponding truncation length of the infinite-order representation polynomial.

Details

See the documentation on aparch for information on the APARCH model. This function provides an easy way to simulate from these models.

Value

A list with four elements is returned: rt are the simulated observations, etat are the underlying innovations, sigt are the correspondingly simulated conditional standard deviations, and cmeans are the simulated conditional means. These four elements are formatted as "ts" class time series objects.

Examples

```
sim <- aparch_sim(n = 1000)
mat <- do.call(cbind, sim)
plot(mat, main = "")
```

---

autoplot,fEGarch_fit-method

Plot Method for Fitting Step Results in the Style of ggplot2

Description

This is method for producing various plots of the decomposition results returned by this package.

Usage

```
## S4 method for signature 'fEGarch_fit'
autoplot(object, which = NULL, ...)
```

Arguments

object	an object returned by the fitting functions of this package, for example by fEGarch.
which	various plots can be selected either via a keyword or a number; enter "returns" or 1 to show a plot of the input training series; enter "means" or 2 to show the fitted conditional means; enter "stand_deviations" or 3 to show the fitted conditional standard deviations; use "residuals" or 4 to show the standardized residuals following the fitted model; the default is which = NULL which then lets you select a plot interactively in the R console.
...	no purpose and only implemented for compatibility.

Details

Create predefined standard plots of the estimation objects returned by the fEGarch package. Plots are created in the ggplot2 plot style. The type of plot can be chosen either interactively from the console, or the argument which can be used to directly select the kind of plot to create (see also the description of the argument which) within the function call.

Value

A ggplot2-graphic object is returned, i.e. an object of classes "gg" and "ggplot".

Author(s)

- Dominik Schulz (Research Assistant) (Department of Economics, Paderborn University), Author and Package Creator

Examples

```
window.zoo <- get("window.zoo", envir = asNamespace("zoo"))
rt <- window.zoo(SP500, end = "2002-12-31")
# Pure conditional volatility model
spec <- fEGarch_spec()
model <- fEGarch(spec, rt)
autoplot(model, which = 3)
```

---

autoplot,fEGarch_risk-method

Plotting of Risk Measure Results (ggplot2)

Description

Plot risk measure results returned by measure_risk as a points-over-threshold plot in style of ggplot2.

Usage

```
## S4 method for signature 'fEGarch_risk'
autoplot(object, which = NULL, ...)
```

Arguments

object	an object returned by measure_risk.
which	one of the levels of VaR and ES saved in object, usually either 0.975 or 0.99 by default.
...	without use.

Value

Returns a ggplot2 plot object.

Examples

```
window.zoo <- get("window.zoo", envir = asNamespace("zoo"))
rt <- window.zoo(SP500, end = "2003-12-31")

egarch_spec() %>%
 fEGarch(rt = rt, n_test = 250) %>%
 predict_roll() %>%
 measure_risk() %>%
 autoplot(which = 0.99)
```

---

close_to_lreturn *Log-Return Calculation From Closing Prices*

Description

Makes log-returns available from an input closing price series.

Usage

```
close_to_lreturn(close)
```

Arguments

close	a closing price series as a numeric vector or some time series object like "ts" or "zoo" ordered chronologically.

Details

Let P_t, $t = 1, \ldots, n$, be an observed closing price series. The function returns

$$r_t = \ln P_t - \ln P_{t-1}$$

, t = 2,. . . ,n.

Value

Returns the log-return series following the input `close`. The output object has one observation less than `close`, but keeps potential time series formatting.

Examples

```
# Assume SP500 + 100 was a closing price series,
# which it is not
close <- SP500 + 100
close_to_lreturn(close)
```

`distr_est` *MLE for Distribution Fitting*

Description

Given a vector of values assumed to stem from independent and identically distributed (iid) random variables, fit a selection of distributions, from the normal distribution, the t-distribution, the generalized error distribution (GED), the average Laplace distribution (ALD), and their skewed variants, to the data using maximum-likelihood estimation (MLE).

Usage

```
distr_est(
  x,
  dist = c("norm", "std", "ged", "ald", "snorm", "sstd", "sged", "sald"),
  fix_mean = NULL,
  fix_sdev = NULL,
  Prange = c(1, 5),
  skip_vcov = FALSE
)

norm_est(x, fix_mean = NULL, fix_sdev = NULL, skip_vcov = FALSE)

std_est(x, fix_mean = NULL, fix_sdev = NULL, skip_vcov = FALSE)

ged_est(x, fix_mean = NULL, fix_sdev = NULL, skip_vcov = FALSE)

ald_est(
```

```
  x,
  fix_mean = NULL,
  fix_sdev = NULL,
  Prange = c(1, 5),
  skip_vcov = FALSE
)

snorm_est(x, fix_mean = NULL, fix_sdev = NULL, skip_vcov = FALSE)

sstd_est(x, fix_mean = NULL, fix_sdev = NULL, skip_vcov = FALSE)

sged_est(x, fix_mean = NULL, fix_sdev = NULL, skip_vcov = FALSE)

sald_est(
  x,
  fix_mean = NULL,
  fix_sdev = NULL,
  Prange = c(1, 5),
  skip_vcov = FALSE
)
```

Arguments

x	a numeric vector with the data.
dist	a character value that specifies the distribution to consider; available are a normal distribution ("norm"), a t-distribution ("std"), a GED ("ged"), an ALD ("ald"), and their skewed variants ("snorm", "sstd", "sged", "sald").
fix_mean	optional; for the default NULL, a location parameter representing the (unconditional) mean of the distribution is also being estimated; for any numerical value, however, the mean will be fixed to the corresponding value and therefore excluded from the estimation itself.
fix_sdev	optional; for the default NULL, a scale parameter representing the (unconditional) standard deviation of the distribution is also being estimated; for any numerical value, however, the standard deviation will be fixed to the corresponding value and therefore excluded from the estimation itself.
Prange	a two-element numeric vector, giving the boundaries of the search space for the shape parameter P in an ALD or its skewed variant.
skip_vcov	a logical indicating whether or not the computation of the variance-covariance matrix should be skipped.

Details

Let x be an individual observation. Let μ a real-valued location parameter, representing the unconditional mean of the distribution, and σ a real-valued scale parameter, representing the unconditional standard deviation. Generally, let θ be a vector with all parameters of the underlying distribution.

The likelihood of x is given through

$$L_x^{\text{norm}}(\theta) = \frac{\sigma^{-1}}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)$$

for a normal distribution,

$$L_x^{\text{std}}(\theta) = \frac{\sigma^{-1}\Gamma\left(\frac{\nu+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)\sqrt{\pi(\nu-2)}}\left[1+\frac{1}{\nu-2}\left(\frac{x-\mu}{\sigma}\right)^2\right]^{-\frac{\nu+1}{2}}$$

for a t-distribution with ν as the degrees of freedom and Γ as the gamma function,

$$L_x^{\text{ged}}(\theta) = \frac{\sigma^{-1}\beta}{2}\sqrt{\frac{C_{\Gamma,3}}{C_{\Gamma,1}^3}} \exp\left\{-\left|\frac{x-\mu}{\sigma}\right|^\beta \left(\frac{C_{\Gamma,3}}{C_{\Gamma,1}}\right)^{\frac{\beta}{2}}\right\}$$

for a GED with β as its real-valued shape and with $C_{\Gamma,i} = \Gamma\left(\frac{i}{\beta}\right)$, $i \in \{1,3\}$, and in

$$L_x^{\text{ald}}(\theta) = \frac{\sigma^{-1}sB}{2} \exp\left(-s\left|\frac{x-\mu}{\sigma}\right|\right) \sum_{j=0}^{P} c_j \left(s\left|\frac{x-\mu}{\sigma}\right|\right)^j$$

for an ALD with P as its discrete shape, where $s = \sqrt{2(P+1)}$,

$$B = 2^{-2P}\binom{2P}{P}, \quad P \geq 0,$$

and

$$c_j = \frac{2(P-j+1)}{j(2P-j+1)}c_{j-1}, \quad j = 2,3,\ldots,P,$$

with $c_0 = c_1 = 1$. The individual-observation likelihoods for the skewed variants are derived analogously from the idea by Fernandez and Steel (1998). The log-likelihoods to maximize over are then just the sum of the log-transformed likelihoods for each observation.

`distr_est` is a general purpose distribution fitting function, where the distribution can be selected through the argument `dist`. `norm_est`, `std_est`, `ged_est`, `ald_est`, `snorm_est`, `sstd_est`, `sged_est`, and `sald_est` are wrappers around `distr_est` in order to directly provide fitting functions for the different distributions available in this package.

Value

Returns a list with the following elements.

References

- Fernandez, C., & Steel, M. F. J. (1998). Bayesian Modeling of Fat Tails and Skewness. Journal of the American Statistical Association, 93(441), 359–371. DOI: 10.1080/01621459.1998.10474117.

Examples

```
# Draw obs. from GED and implement standard deviation 1.2
# and mean 3.1
x <- rged_s(4000, shape = 1.5) * 1.2 + 3.1
# Fit GED
ged_est(x)
# Fit GED differently using distr_est()
distr_est(x, dist = "ged")
# Fit GED while fixing mean and standard deviation
ged_est(x, fix_mean = 3.1, fix_sdev = 1.2)
# Fit another distribution
sstd_est(x)
```

`egarch_type_spec` *Subspecification of EGARCH Family Models*

Description

Two common subspecifications of the broad EGARCH family, namely for a EGARCH-type and a Log-GARCH-type model.

Usage

```
egarch_type_spec(
  orders = c(1, 1),
  long_memo = TRUE,
  cond_dist = c("norm", "std", "ged", "ald", "snorm", "sstd", "sged", "sald"),
  powers = c(1, 1),
  modulus = c(FALSE, FALSE)
)

loggarch_type_spec(
  orders = c(1, 1),
  long_memo = TRUE,
  cond_dist = c("norm", "std", "ged", "ald", "snorm", "sstd", "sged", "sald")
)
```

Arguments

`orders`	a two-element numeric vector with the model orders; the first element is the order p for the term based on $\ln\left(\sigma_t^2\right)$, i.e. the log-transformed conditional variance, while the second element is the order q for the innovation-based term (see Details below for more information).
`long_memo`	a logical value that indicates whether the long-memory version of the model should be considered or not.

cond_dist	a character value stating the underlying conditional distribution to consider; available are a normal distribution ("norm"), a t-distribution ("std"), a generalized error distribution ("ged"), an average Laplace distribution ("ald") and the skewed versions of them ("snorm", "sstd", "sged", "sald").
powers	a two-element numeric vector that states the exponents in the power-transformations of the asymmetry and the magnitude terms in that order (see Details for more information).
modulus	a two-element logical vector indicating if the innovations in the asymmetry and the magnitude terms (in that order) should use a modulus transformation (see Details for more information).

Details

These are wrappers for fEGarch_spec. egarch_type_spec() is when setting model_type in fEGarch_spec to eGARCH. loggarch_type_spec() is a shortcut when setting model_type = "loggarch". See fEGarch_spec for further details.

Value

An object of class "egarch_type_spec" or "loggarch_type_spec" is returned.

---

fEGarch *Fitting Function for Models of the Broader EGARCH Family*

Description

Use quasi-maximum-likelihood estimation to fit a model from the broader EGARCH family to some observed time series.

Usage

```
fEGarch(
  spec = egarch_spec(),
  rt,
  drange = c(0, 1),
  meanspec = mean_spec(),
  Drange = c(0, 1),
  nonparspec = locpol_spec(),
  use_nonpar = FALSE,
  n_test = 0,
  start_pars = NULL,
  LB = NULL,
  UB = NULL,
  control = list(),
  control_nonpar = list(),
  mean_after_nonpar = FALSE,
  parallel = TRUE,
```

```
  ncores = max(1, future::availableCores() - 1),
  trunc = "none",
  presample = 50,
  Prange = c(1, 5),
  skip_vcov = FALSE
)
```

Arguments

spec	an S4 object of class "egarch_type_spec" or "loggarch_type_spec" as returned by the various spec functions of this package, for example fEGarch_spec or its wrappers like megarch_spec or mloggarch_spec, among others.
rt	the input time series to fit the model to ordered from past to present; can also be a "zoo" class object or a "ts" class object.
drange	a two-element numeric vector that indicates the boundaries of the interval over which to search for the fractional differencing parameter d in a long-memory GARCH-type model in the conditional volatility model part; by default, d being searched for on the interval from 0 to 1; note that specific settings in the arguments LB and UB overwrite this argument.
meanspec	an object of class "mean_spec"; indicates the specifications for the model in the conditional mean.
Drange	a two-element numeric vector that indicates the boundaries of the interval over which to search for the fractional differencing parameter d in a long-memory ARMA-type model in the conditional mean model part; by default, D being searched for on the interval from 0 to 1; note that specific settings in the arguments LB and UB overwrite this argument.
nonparspec	an object of class "locpol_spec" returned by locpol_spec; defines the settings of the nonparametric smoothing technique for use_nonpar = TRUE.
use_nonpar	a logical indicating whether or not to implement a semiparametric extension of the volatility model defined through spec; see "Details" for more information.
n_test	a single numerical value indicating, how many observations at the end of rt not to include in the fitting process and to reserve for backtesting.
start_pars	a vector with starting parameters for the optimization; must be of the same length as the output vector of parameters; the default NULL uses internally saved default sets of starting values; see "Details" for the order of elements.
LB	a vector with lower boundaries for parameters; must be of the same length as the output vector of parameters; the default NULL uses internally saved default sets of lower boundary values; see "Details" for the order of elements.
UB	a vector with upper boundaries for parameters; must be of the same length as the output vector of parameters; the default NULL uses internally saved default sets of upper boundary values; see "Details" for the order of elements.
control	a list that is passed to control of the function solnp of the package Rsolnp.
control_nonpar	a list containing changes to the arguments for the hyperparameter estimation algorithm in the nonparametric scale function estimation for use_nonpar = TRUE; see "Details" for more information.

`mean_after_nonpar`
: only for `use_nonpar = TRUE`; considers the unconditional mean of the parametric model part in the QMLE step in a semiparametric model; by default, a zero-mean model is considered for the parametric part in a semiparametric model.

`parallel`
: only relevant for a (skewed) average Laplace (AL) distribution, i.e. if `cond_dist` in `spec` is set to `cond_dist = "ald"` or `cond_dist = "sald"`; `parallel` is a logical value indicating whether or not the slices for the positive integer-valued parameter of the SM distribution should be fitted in parallel for a speed boost.

`ncores`
: only relevant for a (skewed) average Laplace (AL) distribution, i.e. if `cond_dist` in `spec` is set to `cond_dist = "ald"` or `cond_dist = "sald"`, and if simultaneously `parallel = TRUE`; `ncores` is a single numeric value indicating the number of cores to use for parallel computations.

`trunc`
: a positive integer indicating the finite truncation length of the infinite-order polynomials of the infinite-order representations of the long-memory model parts; the character `"none"` is an optional input that specifies that truncation should always be applied back to the first (presample) observation time point, i.e. that maximum length filters should be applied at all times.

`presample`
: the presample length for initialization (for extended EGARCH- / Log-GARCH-type models only relevant for the FARIMA-part, as series in log-transformed conditional variance are initialized by zero).

`Prange`
: a two-element vector that indicates the search boundaries for the parameter P in a (skewed) average Laplace distribution.

`skip_vcov`
: a logical indicating whether or not to skip the computation of the variance-covariance matrix of the parameter estimators and therefore also standard error computation.

Details

For details on the models in the conditional variance, see fEGarch_spec. For details on the models in the conditional mean, see mean_spec. The combined model defined through mean_spec and fEGarch_spec is the specified model. It can be thought of as the model described in mean_spec with $\{r_t\}$ therein being governed by a model from the EGARCH family (see for example Feng et al., 2025) as described in fEGarch_spec, however with mean of $\{r_t\}$ fixed to zero.

The specified model is then fitted using quasi maximum likelihood estimation, where pre-sample values of $g(\eta_{t-1})$ are filled in through its theoretical expectation of zero, which is analogous to not setting pre-sample values in the long-memory case. In addition, in short-memory models, pre-sample values of $ln(\sigma_t^2)$ are roughly approximated through $\ln\left(\hat{\sigma}_t^2\right)$, where $\hat{\sigma}_t^2$ is the sample variance of the observations.

See the references section for sources on the EGARCH (Nelson, 1991), FIEGARCH (Bollerslev and Mikkelsen, 1996), Log-GARCH (Geweke, 1986; Pantula, 1986; Milhoj, 1987) and FILog-GARCH (Feng et al., 2020) models. For information on the FIMLog-GARCH, see Feng et al. (2023).

In the current package version, standard errors of parameter estimates are computed from the Hessian at the optimum of the log-likelihood using `hessian`. To ensure numerical stability and applicability to a huge variety of differently scaled data, parametric models are first fitted to data that is scaled to have sample variance 1. Parameter estimates and other quantities are then either retransformed or recalculated afterwards for the original data.

For a conditional average Laplace distribution, an optimal model for each distribution parameter P from 1 to 5 is estimated (assuming that P is then fixed to the corresponding value). Afterwards, P is then estimated by selecting the estimated model among the five fitted models that has the largest log-likelihood. The five models are, by default, fitted simultaneously using parallel programming techniques (see also the arguments `parallel` and `ncores`, which are only relevant for a conditional average Laplace distribution). After the optimal model (including the estimate of P called $\hat{P}$) has been determined, $P = \hat{P}$ is seen as fixed to obtain the standard errors via the Hessian matrix for the estimates of the continuous parameters. A standard error for $\hat{P}$ is therefore not obtained and the ones obtained for the remaining estimates do not account for $\hat{P}$.

As an alternative, a semiparametric extension of the pure models in the conditional variance can be implemented. If `use_nonpar = TRUE`, `meanspec` is omitted and before fitting a zero-mean model in the conditional volatility following `spec`, a smooth scale function, i.e. a function representing the unconditional standard deviation over time, is being estimated following the specifications in `nonparspec` and `control_nonpar`. This preliminary step stabilizes the input series `rt`, as long-term changes in the unconditional variance are being estimated and removed before the parametric step using either `tsmooth` or `tsmoothlm` depending on whether `spec` specifies a model with short memory or with long memory. `control_nonpar` can be adjusted following the arguments of `tsmooth` for short-memory specifications of `spec`, on the other hand changes to arguments of `tsmoothlm` can be passed to it for long-memory specifications. These arguments specify settings for the automated bandwidth selection algorithms implemented by these two functions. By default, we use the settings `Mcf = "NP"`, `InfR = "Opt"`, `bStart = 0.15`, `bvc = "Y"`, `cb = 0.05`, and `method = "lpr"` within the call to `tsmooth`, as well as `pmin = 0`, `pmax = 1`, `qmin = 0`, `qmax = 1`, `InfR = "Opt"`, `bStart = 0.15`, `cb = 0.05`, and `method = "lpr"` for `tsmoothlm`. `locpol_spec` passed to `nonparspec` handles more direct settings of the local polynomial smoother itself. See the documentation for these functions to get a detailed overview of these settings. Assume $\{r_t\}$ to be the observed series, where $t = 1, 2, \ldots, n$, then $r_t^* = r_t - \bar{r}$, with $\bar{r}$ being the arithmetic mean over the observed r_t, is computed and subsequently $y_t = \ln\left[(r_t^*)^2\right]$. The subtraction of $\bar{r}$ is necessary so that r_t^* are all different from zero almost surely. Once y_t are available, its trend $m(x_t)$, with x_t as the rescaled time on the interval $[0, 1]$, is being estimated using either `tsmooth` or `tsmoothlm` and denoted here by $\hat{m}(x_t)$. Then from $\hat{\xi}_t = y_t - \hat{m}(x_t)$ obtain $\hat{C} = -\ln\left\{\sum_{t=1}^{n} \exp\left(\hat{\xi}_t\right)\right\}$, and obtain the estimated scale function as $\hat{s}(x_t) = \exp\left[\left(\hat{\mu}(x_t) - \hat{C}\right)/2\right]$. The stabilized / standardized version of the series $\{r_t\}$ is then $\tilde{r}_t = r_t^*/\hat{s}(x_t)$, to which a purely parametric volatility model following `spec` is then fitted. The estimated volatility at a given time point is then the product of the estimate of the corresponding scale function value and of the estimated conditional standard deviation (following the parametric model part) for that same time point. See for example Feng et al. (2022) or Letmathe et al. (2023) for more information on the semiparametric extension of volatility models. Moreover, if `bwidth` in the object passed to `nonparspec` is not at its default `NULL` but instead a numeric value between `0` and `0.5`, the automated bandwidth selection is skipped and the provided bandwidth in `bwidth` is used.

The order for manual settings of `start_pars`, `LB` and `UB` is crucial. The correct order is: μ, $\text{ar}_1, \ldots, \text{ar}_{p^*}$, $\text{ma}_1, \ldots, \text{ma}_{q^*}$,$D$,$\omega_\sigma$, $\phi_1, \ldots, \phi_p$, $\psi_1, \ldots, \psi_{q-1}$, κ, γ, d, shape parameter, skewness parameter for Type I models (see `fEGarch_spec`). For Type II models, we have μ, $\text{ar}_1, \ldots, \text{ar}_{p^*}$, $\text{ma}_1, \ldots, \text{ma}_{q^*}$,$D$,$\omega_\sigma$, $\phi_1, \ldots, \phi_p$, $\psi_1, \ldots, \psi_q$, d, shape parameter, skewness parameter. Depending on the exact model specification, parameters irrelevant for the specification at hand should be dropped in `start_pars`, `LB` and `UB`.

Value

An object of S4-class "fEGarch_fit_egarch" "fEGarch_fit_loggarch" is returned depending on the selected model type in the model specification. It contains the following elements.

pars: a named numeric vector with the parameter estimates.

se: a named numeric vector with the obtained standard errors in accordance with the parameter estimates.

vcov_mat: the variance-covariance matrix of the parameter estimates with named columns and rows.

rt: the input object rt (or at least the training data, if n_test is greater than zero); if rt was a "zoo" or "ts" object, the formatting is kept.

sigt: the estimated conditional standard deviations (or for use_nonpar = TRUE the estimated total volatilities, i.e. scale function value times conditional standard deviation); if rt was a "zoo" or "ts" object, the formatting is also applied to sigt.

cmeans: the estimated conditional means; if rt was a "zoo" or "ts" object, the formatting is also applied to cmeans.

etat: the obtained residuals; if rt was a "zoo" or "ts" object, the formatting is also applied to etat.

orders: a two-element numeric vector stating the considered model orders.

cond_dist: a character value stating the conditional distribution considered in the model fitting.

long_memo: a logical value stating whether or not long memory was considered in the model fitting.

llhood: the log-likelihood value obtained at the optimal parameter combination.

inf_criteria: a named two-element numeric vector with the corresponding AIC (first element) and BIC (second element) of the fitted parametric model part; for purely parametric models, these criteria are valid for the entire model; for semiparametric models, they are only valid for the parametric step and are not valid for the entire model.

powers: a two-element numeric vector stating the powers considered for the asymmetry term (first element) and the magnitude term (second element); only exists, if a type I model was fitted.

modulus: a two-element logical vector stating the modulus transformations (TRUE, otherwise FALSE) considered for the asymmetry term (first element) and the magnitude term (second element); only exists, if a type I model was fitted.

meanspec: the settings for the model in the conditional mean; is an object of class "mean_spec" that is identical to the object passed to the input argument meanspec.

test_obs: the observations at the end up the input rt reserved for testing following n_test.

scale_fun: the estimated scale function values, if use_nonpar = TRUE, otherwise NULL; formatting of rt is reused.

nonpar_model: the estimation object returned by either tsmooth or tsmoothlm for use_nonpar = TRUE.

trunc: the input argument trunc.

References

- Bollerslev, T., & Mikkelsen, H. O. (1996). Modeling and pricing long memory in stock market volatility. Journal of Econometrics, 73(1), 151–184. DOI: 10.1016/0304-4076(95)01749-6.
- Feng, Y., Beran, J., Ghosh, S., & Letmathe, S. (2020). Fractionally integrated Log-GARCH with application to value at risk and expected shortfall. Working Papers CIE No. 137, Paderborn University, Center for International Economics. URL: http://groups.uni-paderborn.de/wp-wiwi/RePEc/pdf/ciepap/WP137.pdf.
- Feng, Y., Gries, T., Letmathe, S., & Schulz, D. (2022). The smoots Package in R for Semi-parametric Modeling of Trend Stationary Time Series. The R Journal, 14(1), 182-195. URL: https://journal.r-project.org/articles/RJ-2022-017/.
- Feng, Y., Gries, T., & Letmathe, S. (2023). FIEGARCH, modulus asymmetric FILog-GARCH and trend-stationary dual long memory time series. Working Papers CIE No. 156, Paderborn University. URL: https://econpapers.repec.org/paper/pdnciepap/156.htm.
- Feng, Y., Peitz, C., & Siddiqui, S. (2025). A few useful members of the EGARCH-family with short- or long-memory in volatility. Unpublished working paper at Paderborn University.
- Geweke, J. (1986). Modeling the persistence of conditional variances: A comment. Econometric Reviews, 5(1), 57-61. DOI: 10.1080/07474938608800088.
- Letmathe, S., Beran, J., & Feng, Y. (2023). An extended exponential SEMIFAR model with application in R. Communications in Statistics - Theory and Methods, 53(22), 7914–7926. DOI: 10.1080/03610926.2023.2276049.
- Milhoj, A. (1987). A Multiplicative Parameterization of ARCH Models. University of Copenhagen, Denmark.
- Nelson, D. B. (1991). Conditional Heteroskedasticity in Asset Returns: A New Approach. Econometrica, 59(2), 347–370. DOI: 10.2307/2938260.
- Pantula, S. G. (1986). Modeling the persistence of conditional variances: A comment. Econometric Reviews, 5(1), 71-74. DOI: 10.1080/07474938608800089.

Examples

```
window.zoo <- get("window.zoo", envir = asNamespace("zoo"))
rt <- window.zoo(SP500, end = "2002-12-31")
# Pure conditional volatility model
spec <- fEGarch_spec()
model <- fEGarch(spec, rt)
model
# Simultaneously model conditional mean
spec <- egarch_spec()
model2 <- suppressWarnings(
 fEGarch(spec, rt, meanspec = mean_spec(orders = c(1, 1)))
)
model2
```

fEGarch_sim *Simulate From Models of the Broader EGARCH Family*

Description

A streamlined simulation function to simulate from models that are part of the broader EGARCH family specifiable through fEGarch_spec.

Usage

```
fEGarch_sim(
  spec = egarch_spec(),
 pars = list(mu = 0, ar = numeric(0), ma = numeric(0), D = 0, omega_sig = -9, phi = 0.8,
   psi = numeric(0), kappa = -0.2, gamma = 0.3, d = 0, df = 10, shape = 2, P = 3, skew =
     1),
  n = 1000,
  nstart = 5000,
  trunc = "none"
)
```

Arguments

spec	an object of class "egarch_type_spec" or "loggarch_type_spec" returned by fEGarch_spec or related wrapper functions; note that the model orders in the conditional mean and in the conditional variance as well as the long-memory settings are not controlled through the element orders and long_memo of spec but solely through the parameter settings in pars.
pars	a named list with the parameter specifications; the user can provide a named list with only the settings they would like to adjust relative to the default settings.
n	the number of observations to return.
nstart	the number of burn-in observations to simulate before the final n values to keep; the first nstart values are not returned; if a dual model, i.e. with model in the conditional mean and in the conditional variance, is considered, two times nstart is considered in the first simulation step in the conditional variance, so that n + nstart values can be fed into the second simulation step for the conditional mean.
trunc	a truncation for the finite-order coefficient series in long-memory models; can either be the character "none" for truncation back to the very first observation at each time point, or to any positive integer for setting the corresponding truncation length of the infinite-order representation polynomial.

Details

See the documentation on fEGarch_spec for information on the models of the broader EGARCH family. This function provides an easy way to simulate from these models.

Value

A list with four elements is returned: `rt` are the simulated observations, `etat` are the underlying innovations, `sigt` are the correspondingly simulated conditional standard deviations, and `cmeans` are the simulated conditional means. These four elements are formatted as `"ts"` class time series objects.

Examples

```
spec <- megarch_spec()
sim <- fEGarch_sim(spec = spec)
mat <- do.call(cbind, sim)
plot(mat, main = "")
```

---

fEGarch_spec *General EGARCH Family Model Specification*

Description

Create an object with specifications for a model from the broader EGARCH family.

Usage

```
fEGarch_spec(
  model_type = c("egarch", "loggarch"),
  orders = c(1, 1),
  long_memo = FALSE,
  cond_dist = c("norm", "std", "ged", "ald", "snorm", "sstd", "sged", "sald"),
  powers = c(1, 1),
  modulus = c(FALSE, FALSE)
)
```

Arguments

`model_type`	a character value (either `"egarch"` or `"loggarch"`) that indicates the type of model to implement (see Details for more information).
`orders`	a two-element numeric vector with the model orders; the first element is the order p for the term based on $\ln\left(\sigma_t^2\right)$, i.e. the log-transformed conditional variance, while the second element is the order q for the innovation-based term (see Details below for more information).
`long_memo`	a logical value that indicates whether the long-memory version of the model should be considered or not.
`cond_dist`	a character value stating the underlying conditional distribution to consider; available are a normal distribution (`"norm"`), a t-distribution (`"std"`), a generalized error distribution (`"ged"`), an average Laplace distribution (`"ald"`) and the skewed versions of them (`"snorm"`, `"sstd"`, `"sged"`, `"sald"`).

`powers`	a two-element numeric vector that states the exponents in the power-transformations of the asymmetry and the magnitude terms in that order (see Details for more information).
`modulus`	a two-element logical vector indicating if the innovations in the asymmetry and the magnitude terms (in that order) should use a modulus transformation (see Details for more information).

Details

Let $\{r_t\}$, with $t \in \mathbb{Z}$ as the time index, be a theoretical time series that follows

$$r_t = \mu + \sigma_t \eta_t \text{ with } \eta_t \sim \text{IID}(0,1), \text{ where}$$

$$\ln\left(\sigma_t^2\right) = \omega_\sigma + \theta(B) g(\eta_{t-1}).$$

Here, $\eta_t \sim \text{IID}(0,1)$ means that the innovations η_t are independent and identically distributed (iid) with mean zero and variance one, whereas $\sigma_t > 0$ are the conditional standard deviations in r_t. Note that $\ln(\cdot)$ denotes the natural logarithm. Moreover, B is the backshift operator and $\theta(B) = 1 + \sum_{i=1}^{\infty} \theta_i B^i$, where θ_i, $i = 1, 2, \ldots$, are real-valued coefficients. $g(\eta_{t-1})$ is a suitable function in η_{t-1}. Generally, $\{g(\eta_t)\}$ should be an iid zero-mean sequence with finite variance. We have $\mu = E(r_t)$ as a real-valued parameter. The real-valued parameter ω_σ is in fact $\omega_\sigma = E\left[\ln\left(\sigma_t^2\right)\right]$. This previous set of equations defines the broader family of EGARCH models (Feng et al., 2025; Ayensu et al., 2025), from which subtypes are described in the following that depend on the choice of g.

Type I :

We have $\theta(B) = \phi^{-1}(B)(1-B)^{-d}\psi(B)$, where

$$\phi(B) = 1 - \sum_{i=1}^{p} \phi_i B^i \text{ and}$$

$$\psi(B) = 1 + \sum_{j=1}^{q-1} \psi_j B^j,$$

are characteristic polynomials with real coefficients $\phi_0, \ldots, \phi_p, \psi_0, \ldots, \psi_{q-1}$, by fixing $\phi_0 = \psi_0 = 1$, and without common roots. Furthermore, the fractional differencing parameter is $d \in [0, 1]$.

$g(\cdot)$ can be defined in different ways. Following a type I specification (`model_type = "egarch"`), we have

$$g(\eta_t) = \kappa\{g_a(\eta_t) - E[g_a(\eta_t)]\} + \gamma\{g_m(\eta_t) - E[g_m(\eta_t)]\}$$

with $g_a(\eta_t)$ and $g_m(\eta_t)$ being suitable transformations of η_t and where κ and γ are two additional real-valued parameters. In case of a simple (FI)EGARCH, we have $g_a(\eta_t) = \eta_t$ (and therefore with $E[g_a(\eta_t)] = 0$) and $g_m(\eta_t) = |\eta_t|$.

Generally, we consider two cases:

$$g_{1,a}(\eta_t) = \text{sgn}(\eta_t)\,|\eta_t|^{p_a}/p_a \text{ and}$$

$$g_{2,a}(\eta_t) = \text{sgn}(\eta_t)\,[(|\eta_t|+1)^{p_a} - 1]/p_a$$

whereas

$$g_{1,m}(\eta_t) = |\eta_t|^{p_m}/p_m \text{ and}$$

$$g_{2,m}(\eta_t) = \left[(|\eta_t| + 1)^{p_m} - 1\right]/p_m.$$

Note that $\text{sgn}(\eta_t)$ denotes the sign of η_t. $g_{1,\cdot}$ incorporates a power transformation and $g_{2,\cdot}$ a modulus transformation together with a power transformation. The choices $g_{1,a}$ and $g_{2,a}$ correspond to setting the first element in `modulus` to `FALSE` or `TRUE`, respectively, under `model_type = "egarch"`, where p_a can be selected via the first element in `powers`. As a special case, for $p_a = 0$, a log-transformation is employed and the division through p_a is dropped, i.e. $g_{1,a}(\eta_t) = \text{sgn}(\eta_t)\ln(|\eta_t|)$ and $g_{2,a}(\eta_t) = \text{sgn}(\eta_t)\ln(|\eta_t| + 1)$ are employed for $p_a = 0$. Completely analogous thoughts hold for $g_{1,m}$ and $g_{2,m}$ and the second elements in the arguments `modulus` and `powers`. The aforementioned model family is a type I model selectable through `model_type = "egarch"`. Simple (FI)EGARCH models are given through selection of $g_{1,a}(\cdot)$ and $g_{1,m}(\cdot)$ in combination with $p_a = p_m = 1$. Another of such type I models is the FIMLog-GARCH (Feng et al., 2023), where instead $g_{2,a}(\cdot)$ and $g_{2,m}(\cdot)$ are selected with $p_a = p_m = 0$.

Type II :

As additional specifications of a Log-GARCH and a FILog-GARCH, which belong to the broader EGARCH family, we redefine

$$\psi(B) = 1 + \sum_{j=1}^{q} \psi_j B^j$$

now as a polynomial of order q and

$$\ln\left(\sigma_t^2\right) = \omega_\sigma + \left[\phi^{-1}(B)(1-B)^{-d}\psi(B) - 1\right]\xi_t,$$

where

$$\xi_t = \ln\left(\eta_t^2\right) - E\left[\ln\left(\eta_t^2\right)\right].$$

Everything else is defined as before. Since

$$\phi^{-1}(B)(1-B)^{-d}\psi(B) - 1 = \sum_{i=1}^{\infty}\gamma_i B^i = \gamma_1 B\left[\sum_{i=1}^{\infty}(\gamma_i/\gamma_1)B^{i-1}\right] = \gamma_1 B\left[1 + \sum_{i=1}^{\infty}\theta_i B^i\right] = \theta(B)\gamma_1 B,$$

where $\theta_i = \gamma_{i+1}/\gamma_1$, $i = 0, 1, \ldots$, and by defining

$$g(\eta_{t-1}) = \gamma_1\left\{\ln\left(\eta_{t-1}^2\right) - E\left[\ln\left(\eta_{t-1}^2\right)\right]\right\} = 2\gamma_1\left\{\ln(|\eta_{t-1}|) - E\left[\ln(|\eta_{t-1}|)\right]\right\},$$

the equation of $\ln\left(\sigma_t^2\right)$ can be stated to be

$$\ln\left(\sigma_t^2\right) = \omega_\sigma + \theta(B)g(\eta_{t-1})$$

as in the broad EGARCH family at the very beginning. Therefore, Log-GARCH and FI-Log-GARCH models are equivalent to the type I models, where $\kappa = 0$ with usage of $g_{1,m}$ with $p_m = 0$ and where $\gamma = 2\gamma_1$. Nonetheless, in this package, the type II models make use of the more common parameterization of $\ln\left(\sigma_t^2\right)$ stated at the beginning of the type II model description.

This describes the established Log-GARCH models as part of the broad EGARCH family (type II models; `model_type = "loggarch"`).

General information :

While the arguments `powers` and `modulus` are only relevant under a type I model, i.e. for `model_type = "egarch"`, the arguments `orders`, `long_memo` and `cond_dist` are meaningful for both `model_type = "egarch"` and `model_type = "loggarch"`, i.e. both under type I and II models. The first element

of the two-element vector orders is the order p, while the second element is the order q. Furthermore, for `long_memo = TRUE`, the mentioned models are kept as they are, while for `long_memo = FALSE` the parameter d is set to zero. `cond_dist` controls the conditional distribution. The unconditional mean μ is controlled via the function `mean_spec`; for `include_mean = FALSE` therein, μ is not being estimated and fixed to zero; its default is however `include_mean = TRUE`.

See also the closely related spec-functions that immediately create specifications of specific submodels of the broad EGARCH family. These functions are `egarch_spec()`, `fiegarch_spec()`, `loggarch_spec()`,`filoggarch_spec()`, `megarch_spec()`, `mloggarch_spec()` and `mafiloggarch_spec()`, which are all wrappers for `fEGarch_spec()`.

See the references section for sources on the EGARCH (Nelson, 1991), FIEGARCH (Bollerslev and Mikkelsen, 1996), Log-GARCH (Geweke, 1986; Pantula, 1986; Milhoj, 1987) and FILog-GARCH (Feng et al., 2020) models.

Value

An object of either class `"egarch_type_spec"` or `"loggarch_type_spec"` is returned, depending on the choice for the input argument `model_type`.

References

- Ayensu, O. K., Feng, Y., & Schulz, D. (2025). Recent Extensions of Exponential GARCH Models: Theory and Application. Forthcoming preprint, Paderborn University.
- Bollerslev, T., & Mikkelsen, H. O. (1996). Modeling and pricing long memory in stock market volatility. Journal of Econometrics, 73(1), 151–184. DOI: 10.1016/0304-4076(95)01749-6.
- Feng, Y., Beran, J., Ghosh, S., & Letmathe, S. (2020). Fractionally integrated Log-GARCH with application to value at risk and expected shortfall. Working Papers CIE No. 137, Paderborn University, Center for International Economics. URL: http://groups.uni-paderborn.de/wp-wiwi/RePEc/pdf/ciepap/WP137.pdf.
- Feng, Y., Gries, T., & Letmathe, S. (2023). FIEGARCH, modulus asymmetric FILog-GARCH and trend-stationary dual long memory time series. Working Papers CIE No. 156, Paderborn University. URL: https://econpapers.repec.org/paper/pdnciepap/156.htm.
- Feng, Y., Peitz, C., & Siddiqui, S. (2025). A few useful members of the EGARCH-family with short- or long-memory in volatility. Unpublished working paper at Paderborn University.
- Geweke, J. (1986). Modeling the persistence of conditional variances: A comment. Econometric Reviews, 5(1), 57-61. DOI: 10.1080/07474938608800088.
- Milhoj, A. (1987). A Multiplicative Parameterization of ARCH Models. University of Copenhagen, Denmark.
- Nelson, D. B. (1991). Conditional Heteroskedasticity in Asset Returns: A New Approach. Econometrica, 59(2), 347–370. DOI: 10.2307/2938260.
- Pantula, S. G. (1986). Modeling the persistence of conditional variances: A comment. Econometric Reviews, 5(1), 71-74. DOI: 10.1080/07474938608800089.

Examples

```
# EGARCH(1, 1) with cond. normal distribution
spec1 <- fEGarch_spec()
```

```
# EGARCH(2, 1) with cond. t-distribution
spec2 <- fEGarch_spec(orders = c(2, 1), cond_dist = "std")
# FIEGARCH(1, 1) with cond. normal distribution
spec3 <- fEGarch_spec(long_memo = TRUE)
# MEGARCH(1, 1) with cond. generalized error distribution
spec4 <- fEGarch_spec(modulus = c(TRUE, FALSE), powers = c(0, 1))
# Some unnamed specification
spec5 <- fEGarch_spec(
 model_type = "egarch",
 orders = c(1, 1),
 long_memo = TRUE,
 cond_dist = "std",
 powers = c(0.25, 0.75),
 modulus = c(TRUE, FALSE)
)
```

`fiaparch` *FIAPARCH Model Fitting*

Description

Fit a fractionally integrated APARCH model under the six most common and further conditional distributions to observed data using quasi maximum-likelihood estimation.

Usage

```
fiaparch(
  rt,
  orders = c(1, 1),
  cond_dist = c("norm", "std", "ged", "ald", "snorm", "sstd", "sged", "sald"),
  drange = c(0, 1),
  meanspec = mean_spec(),
  Drange = c(0, 1),
  nonparspec = locpol_spec(),
  use_nonpar = FALSE,
  n_test = 0,
  start_pars = NULL,
  LB = NULL,
  UB = NULL,
  control = list(),
  control_nonpar = list(),
  mean_after_nonpar = FALSE,
  parallel = TRUE,
  ncores = max(1, future::availableCores() - 1),
  trunc = "none",
  presample = 50,
  Prange = c(1, 5),
```

```
  fix_delta = c(NA, 1, 2),
  skip_vcov = FALSE
)
```

Arguments

rt	the observed series ordered from past to present; can be a numeric vector, a "zoo" class time series object, or a "ts" class time series object.
orders	a two-element numeric vector containing the two model orders p and q (see Details for more information); currently, only the default orders = c(1, 1) is supported; other specifications of a two-element numeric vector will lead to orders = c(1, 1) being run and a warning message being returned.
cond_dist	the conditional distribution to consider as a character object; the default is a conditional normal distribution "norm"; available are also, however, a t-distribution ("std"), a generalized error distribution ("ged"), an average Laplace distribution ("ald"), and their four skewed variants ("snorm", "sstd", "sged", "sald").
drange	a two-element numeric vector that gives the boundaries of the search interval for the fractional differencing parameter d in the conditional volatility model part; is overwritten by the settings of the arguments LB and UB.
meanspec	an object of class "mean_spec"; indicates the specifications for the model in the conditional mean.
Drange	a two-element numeric vector that indicates the boundaries of the interval over which to search for the fractional differencing parameter D in a long-memory ARMA-type model in the conditional mean model part; by default, D being searched for on the interval from 0 to $0.5 - 1 \times 10^{-6}$; note that specific settings in the arguments LB and UB overwrite this argument.
nonparspec	an object of class "locpol_spec" returned by locpol_spec; defines the settings of the nonparametric smoothing technique for use_nonpar = TRUE.
use_nonpar	a logical indicating whether or not to implement a semiparametric extension of the volatility model defined through spec; see "Details" for more information.
n_test	a single numerical value indicating, how many observations at the end of rt not to include in the fitting process and to reserve for backtesting.
start_pars	the starting parameters for the numerical optimization routine; should be of the same length as the parameter output vector within the output object (also keeping the same order); for NULL, an internally saved default set of values is used; see "Details" for the order of elements; elements should be set with respect to a series rescaled to have sample variance one.
LB	the lower boundaries of the parameters in the numerical optimization routine; should be of the same length as the parameter output vector within the output object (also keeping the same order); for NULL, an internally saved default set of values is used; see "Details" for the order of elements; elements should be set with respect to a series rescaled to have sample variance one.
UB	the upper boundaries of the parameters in the numerical optimization routine; should be of the same length as the parameter output vector within the output object (also keeping the same order); for NULL, an internally saved default set of

values is used; see "Details" for the order of elements; elements should be set with respect to a series rescaled to have sample variance one.

`control` a list that is passed to `control` of the function `solnp` of the package `Rsolnp`.

`control_nonpar` a list containing changes to the arguments for the hyperparameter estimation algorithm in the nonparametric scale function estimation for `use_nonpar = TRUE`; see "Details" for more information.

`mean_after_nonpar`
only for `use_nonpar = TRUE`; considers the unconditional mean of the parametric model part in the QMLE step in a semiparametric model; by default, a zero-mean model is considered for the parametric part in a semiparametric model.

`parallel` only relevant for a (skewed) average Laplace (AL) distribution, i.e. if `cond_dist` in `spec` is set to `cond_dist = "ald"` or `cond_dist = "sald"`; `parallel` is a logical value indicating whether or not the slices for the positive integer-valued parameter of the SM distribution should be fitted in parallel for a speed boost.

`ncores` only relevant for a (skewed) average Laplace (AL) distribution, i.e. if `cond_dist` in `spec` is set to `cond_dist = "ald"` or `cond_dist = "sald"`, and if simultaneously `parallel = TRUE`; `ncores` is a single numeric value indicating the number of cores to use for parallel computations.

`trunc` a positive integer indicating the finite truncation length of the infinite-order polynomials of the infinite-order representations of the long-memory model parts; the character `"none"` is an optional input that specifies that truncation should always be applied back to the first (presample) observation time point, i.e. that maximum length filters should be applied at all times.

`presample` the presample length for initialization (for extended EGARCH- / Log-GARCH-type models only relevant for the FARIMA-part, as series in log-transformed conditional variance are initialized by zero).

`Prange` a two-element vector that indicates the search boundaries for the parameter P in a (skewed) average Laplace distribution.

`fix_delta` let the parameter δ be either free (`fix_delta = NA`), fix it to 1 (`fix_delta = 1`) or to 2 (`fix_delta = 2`); the latter two are specific submodels of a FIAPARCH.

`skip_vcov` a logical indicating whether or not to skip the computation of the variance-covariance matrix of the parameter estimators and therefore also standard error computation.

Details

Consider a FIAPARCH(p, d, q) with constant asymmetry term γ regardless of the lag. Let $\{r_t\}$, with $t \in \mathbb{Z}$ as the time index, be a theoretical time series that follows

$$r_t = \mu + \varepsilon_t \text{ with } \varepsilon_t = \sigma_t \eta_t \text{ and } \eta_t \sim \text{IID}(0, 1), \text{ where}$$

$$\sigma_t^\delta = \omega + \left[1 - \beta^{-1}(B)\phi(B)(1 - B)^d\right] \left(|\varepsilon_t| - \gamma\varepsilon_t\right)^\delta .$$

Here, $\eta_t \sim \text{IID}(0, 1)$ means that the innovations η_t are independent and identically distributed (iid) with mean zero and variance one, whereas $\sigma_t > 0$ are the conditional standard deviations in r_t. Moreover, B is the backshift operator and $\beta(B) = 1 - \sum_{j=1}^{q} \beta_j B^j$, where β_j, $j = 1, 2, \ldots, q$, are

real-valued coefficients. Furthermore, $\phi(B) = 1 - \sum_{i=1}^{p} \phi_i B^i$, where ϕ_i, $i = 1, 2, \ldots, p$, are real-valued coefficients. p and q are the model orders definable through the argument `orders`, where p is the first element and q is the second element in the argument. In addition, we have $\mu = E(r_t)$ as a real-valued parameter and $\gamma \in (-1, 1)$ and $d \in [0, 1]$ as the parameter for the level of integration. With $d = 0$ the model reduces to a short-memory APARCH, for $d = 1$ we have a full integration, and for $d \in (0, 1)$, we have fractional integration, where $d \in (0, 0.5)$ is considered to describe a long-memory process. $\omega > 0$ is the intercept. It is assumed that all β_j and ϕ_i are non-negative.

The pre-sample values of $(|\varepsilon_t| - \gamma\varepsilon_t)^{\delta}$ are replaced by the in-sample arithmetic mean of

$$(|r_t^* - \bar{r}_t^*| - \gamma_0 (r_t^* - \bar{r}_t^*))^{\delta_0},$$

where r_t^* are considered as the observations and $\bar{r}_t^*$ is the sample mean of r_t^*. Initial values for γ_0 and δ_0 as initial guesses for γ and δ are obtained from a previous fitting of a short-memory APARCH.

Currently, only a model of orders $p = 1$ with $q = 1$ can be fitted; to ensure the non-negativity of all of the infinite-order coefficient series $\left[1 - \beta^{-1}(B)\phi(B)(1 - B)^d\right]$, which in combination with $\omega > 0$ ensures that all the conditional volatilities are greater than zero, we employ inequality constraints ensuring that the first 50 coefficients of the infinite-order ARCH-representation are non-negative as an approximation to ensuring that all of the coefficients are non-negative. To ensure that they are non-negative, one may in theory consider the sufficient conditions mentioned in Bollerslev and Mikkelsen (1996) or Tse (1998), which are however sometimes restrictive, or the simultaneously necessary and sufficient conditions by Conrad and Haag (2006), which are however complex to implement properly.

See also the reference section for sources on the APARCH (Ding et al., 1993) and the FIAPARCH (Tse, 1998) models.

The truncated infinite order polynomial is computed following the idea by Nielsen and Noel (2021) as is the series of conditional variances for most computational efficiency. To ensure stability of the first fitted in-sample conditional standard deviations, we however use a small, but also adjustable (also to length zero) presample, which may introduce biases into the parameter estimators.

In the current package version, standard errors of parameter estimates are computed from the Hessian at the optimum of the log-likelihood using `hessian`. To ensure numerical stability and applicability to a huge variety of differently scaled data, parametric models are first fitted to data that is scaled to have sample variance 1. Parameter estimates and other quantities are then either retransformed or recalculated afterwards for the original data.

For a conditional average Laplace distribution, an optimal model for each distribution parameter P from 1 to 5 is estimated (assuming that P is then fixed to the corresponding value). Afterwards, P is then estimated by selecting the estimated model among the five fitted models that has the largest log-likelihood. The five models are, by default, fitted simultaneously using parallel programming techniques (see also the arguments `parallel` and `ncores`, which are only relevant for a conditional average Laplace distribution). After the optimal model (including the estimate of P called $\hat{P}$) has been determined, $P = \hat{P}$ is seen as fixed to obtain the standard errors via the Hessian matrix for the estimates of the continuous parameters. A standard error for $\hat{P}$ is therefore not obtained and the ones obtained for the remaining estimates do not account for $\hat{P}$.

An ARMA-FIAPARCH or a FARIMA-FIAPARCH can be fitted by adjusting the argument `meanspec` correspondingly.

As an alternative, a semiparametric extension of the pure models in the conditional variance can be implemented. If `use_nonpar = TRUE`, `meanspec` is omitted and before fitting a zero-mean model in

the conditional volatility following the remaining function arguments, a smooth scale function, i.e. a function representing the unconditional standard deviation over time, is being estimated following the specifications in `nonparspec` and `control_nonpar`. This preliminary step stabilizes the input series `rt`, as long-term changes in the unconditional variance are being estimated and removed before the parametric step using `tsmoothlm`. `control_nonpar` can be adjusted following to make changes to the arguments of `tsmoothlm` for long-memory specifications. These arguments specify settings for the automated bandwidth selection algorithms implemented by this function. By default, we use the settings `pmin = 0`, `pmax = 1`, `qmin = 0`, `qmax = 1`, `InfR = "Nai"`, `bStart = 0.15`, `cb = 0.05`, and `method = "lpr"` for `tsmoothlm`. `locpol_spec` passed to `nonparspec` handles more direct settings of the local polynomial smoother itself. See the documentation for these functions to get a detailed overview of these settings. Assume $\{r_t\}$ to be the observed series, where $t = 1, 2, \ldots, n$, then $r_t^* = r_t - \bar{r}$, with $\bar{r}$ being the arithmetic mean over the observed r_t, is computed and subsequently $y_t = \ln\left[(r_t^*)^2\right]$. The subtraction of $\bar{r}$ is necessary so that r_t^* are all different from zero almost surely. Once y_t are available, its trend $m(x_t)$, with x_t as the rescaled time on the interval $[0, 1]$, is being estimated using `tsmoothlm` and denoted here by $\hat{m}(x_t)$. Then from $\hat{\xi}_t = y_t - \hat{m}(x_t)$ obtain $\hat{C} = -\ln\left\{\sum_{t=1}^{n} \exp\left(\hat{\xi}_t\right)\right\}$, and obtain the estimated scale function as $\hat{s}(x_t) = \exp\left[\left(\hat{\mu}(x_t) - \hat{C}\right)/2\right]$. The stabilized / standardized version of the series $\{r_t\}$ is then $\tilde{r}_t = r_t^*/\hat{s}(x_t)$, to which a purely parametric volatility model following the remaining function arguments is then fitted. The estimated volatility at a given time point is then the product of the estimate of the corresponding scale function value and of the estimated conditional standard deviation (following the parametric model part) for that same time point. See for example Feng et al. (2022) or Letmathe et al. (2023) for more information on the semiparametric extension of volatility models.

The order for manual settings of `start_pars`, `LB` and `UB` is crucial. The correct order is: μ, $\text{ar}_1, \ldots, \text{ar}_{p^*}$, $\text{ma}_1, \ldots, \text{ma}_{q^*}$,$D$,$\omega$, ϕ, β, γ, δ, d, shape parameter, skewness parameter. Depending on the exact model specification, parameters irrelevant for the specification at hand should be dropped in `start_pars`, `LB` and `UB`.

Value

An object of S4-class `"fEGarch_fit_fiaparch"` is returned. It contains the following elements.

`pars`: a named numeric vector with the parameter estimates.

`se`: a named numeric vector with the obtained standard errors in accordance with the parameter estimates.

`vcov_mat`: the variance-covariance matrix of the parameter estimates with named columns and rows.

`rt`: the input object `rt` (or at least the training data, if `n_test` is greater than zero); if `rt` was a `"zoo"` or `"ts"` object, the formatting is kept.

`cmeans`: the estimated conditional means; if `rt` was a `"zoo"` or `"ts"` object, the formatting is also applied to `cmeans`.

`sigt`: the estimated conditional standard deviations (or for `use_nonpar = TRUE` the estimated total volatilities, i.e. scale function value times conditional standard deviation); if `rt` was a `"zoo"` or `"ts"` object, the formatting is also applied to `sigt`.

`etat`: the obtained residuals; if `rt` was a `"zoo"` or `"ts"` object, the formatting is also applied to `etat`.

`orders`: a two-element numeric vector stating the considered model orders.

`cond_dist`: a character value stating the conditional distribution considered in the model fitting.

`long_memo`: a logical value stating whether or not long memory was considered in the model fitting.

`llhood`: the log-likelihood value obtained at the optimal parameter combination.

`inf_criteria`: a named two-element numeric vector with the corresponding AIC (first element) and BIC (second element) of the fitted parametric model part; for purely parametric models, these criteria are valid for the entire model; for semiparametric models, they are only valid for the parametric step and are not valid for the entire model.

`meanspec`: the settings for the model in the conditional mean; is an object of class `"mean_spec"` that is identical to the object passed to the input argument `meanspec`.

`test_obs`: the observations at the end up the input `rt` reserved for testing following `n_test`.

`scale_fun`: the estimated scale function values, if `use_nonpar = TRUE`, otherwise `NULL`; formatting of `rt` is reused.

`nonpar_model`: the estimation object returned by `tsmoothlm` for `use_nonpar = TRUE`.

`trunc`: the input argument `trunc`.

References

- Bollerslev, T., & Mikkelsen, H. O. (1996). Modeling and pricing long memory in stock market volatility. Journal of Econometrics, 73(1): 151-184. DOI: 10.1016/0304-4076(95)01736-4.
- Conrad, C., & Haag, B. R. (2006). Inequality constraints in the fractionally integrated GARCH model. Journal of Financial Econometrics, 4(3): 413-449. DOI: 10.1093/jjfinec/nbj015.
- Ding, Z., Granger, C. W. J., & Engle, R. F. (1993). A long memory property of stock market returns and a new model. Journal of Empirical Finance, 1(1): 83-106. DOI: 10.1016/0927-5398(93)90006-D.
- Feng, Y., Gries, T., Letmathe, S., & Schulz, D. (2022). The smoots Package in R for Semiparametric Modeling of Trend Stationary Time Series. The R Journal, 14(1), 182-195. URL: https://journal.r-project.org/articles/RJ-2022-017/.
- Letmathe, S., Beran, J., & Feng, Y. (2023). An extended exponential SEMIFAR model with application in R. Communications in Statistics - Theory and Methods, 53(22), 7914–7926. DOI: 10.1080/03610926.2023.2276049.
- Nielsen, M. O., & Noel, A. L. (2021). To infinity and beyond: Efficient computation of ARCH(∞) models. Journal of Time Series Analysis, 42(3), 338–354. DOI: 10.1111/jtsa.12570.
- Tse, Y. K. (1998). The conditional heteroskedasticity of the yen-dollar exchange rate. Journal of Applied Econometrics, 13(1): 49-55. DOI: 10.1002/(SICI)1099-1255(199801/02)13:1<49::AID-JAE459>3.0.CO;2-O.

Examples

```
window.zoo <- get("window.zoo", envir = asNamespace("zoo"))
rt <- window.zoo(SP500, end = "2002-12-31")
model <- fiaparch(rt)
model
```

`fiaparch_sim` *Simulate From FIAPARCH Models*

Description

A streamlined simulation function to simulate from fractionally integrated asymmetric power autoregressive conditional heteroskedasticity (FIAPARCH) models.

Usage

```
fiaparch_sim(
 pars = list(mu = 0, ar = numeric(0), ma = numeric(0), D = 0, omega = 4e-04, phi = 0.2,
  beta = 0.4, gamma = 0.1, delta = 2, d = 0.25, df = 10, shape = 2, P = 3, skew = 1),
 cond_dist = c("norm", "std", "ged", "ald", "snorm", "sstd", "sged", "sald"),
 n = 1000,
 nstart = 5000,
 trunc = "none"
)
```

Arguments

`pars`	a named list with the parameter specifications; the user can provide a named list with only the settings they would like to adjust relative to the default settings.
`cond_dist`	a one-element character vector specifying the conditional distribution to consider.
`n`	the number of observations to return.
`nstart`	the number of burn-in observations to simulate before the final `n` values to keep; the first `nstart` values are not returned; if a dual model, i.e. with model in the conditional mean and in the conditional variance, is considered, two times `nstart` is considered in the first simulation step in the conditional variance, so that `n + nstart` values can be fed into the second simulation step for the conditional mean.
`trunc`	a truncation for the finite-order coefficient series in long-memory models; can either be the character `"none"` for truncation back to the very first observation at each time point, or to any positive integer for setting the corresponding truncation length of the infinite-order representation polynomial.

Details

See the documentation on `fiaparch` for information on the FIAPARCH model. This function provides an easy way to simulate from these models.

Value

A list with four elements is returned: `rt` are the simulated observations, `etat` are the underlying innovations, `sigt` are the correspondingly simulated conditional standard deviations, and `cmeans` are the simulated conditional means. These four elements are formatted as `"ts"` class time series objects.

Examples

```
sim <- fiaparch_sim(n = 1000)
mat <- do.call(cbind, sim)
plot(mat, main = "")
```

---

figarch *FIGARCH Model Fitting*

Description

Fit a fractionally integrated GARCH (FIGARCH) model under the six most common and further conditional distributions to observed data using quasi maximum-likelihood estimation.

Usage

```
figarch(
  rt,
  orders = c(1, 1),
  cond_dist = c("norm", "std", "ged", "ald", "snorm", "sstd", "sged", "sald"),
  drange = c(0, 1),
  meanspec = mean_spec(),
  Drange = c(0, 1),
  nonparspec = locpol_spec(),
  use_nonpar = FALSE,
  n_test = 0,
  start_pars = NULL,
  LB = NULL,
  UB = NULL,
  control = list(),
  control_nonpar = list(),
  mean_after_nonpar = FALSE,
  parallel = TRUE,
  ncores = max(1, future::availableCores() - 1),
  trunc = "none",
  presample = 50,
  Prange = c(1, 5),
  skip_vcov = FALSE
)
```

Arguments

rt	the observed series ordered from past to present; can be a numeric vector or a "zoo" class time series object.
orders	a two-element numeric vector containing the two model orders p and q (see Details for more information); currently, only the default orders = c(1, 1) is supported; other specifications of a two-element numeric vector will lead to orders = c(1, 1) being run and a warning message being returned.
cond_dist	the conditional distribution to consider as a character object; the default is a conditional normal distribution "norm"; available are also, however, a t-distribution ("std"), a generalized error distribution ("ged"), an average Laplace distribution ("ald"), and their four skewed variants ("snorm", "sstd", "sged", "sald").
drange	a two-element numeric vector that gives the boundaries of the search interval for the fractional differencing parameter d in the conditional volatility model part; is overwritten by the settings of the arguments LB and UB.
meanspec	an object of class "mean_spec"; indicates the specifications for the model in the conditional mean.
Drange	a two-element numeric vector that indicates the boundaries of the interval over which to search for the fractional differencing parameter D in a long-memory ARMA-type model in the conditional mean model part; by default, D being searched for on the interval from 0 to $0.5 - 1 \times 10^{-6}$; note that specific settings in the arguments LB and UB overwrite this argument.
nonparspec	an object of class "locpol_spec" returned by locpol_spec; defines the settings of the nonparametric smoothing technique for use_nonpar = TRUE.
use_nonpar	a logical indicating whether or not to implement a semiparametric extension of the volatility model defined through spec; see "Details" for more information.
n_test	a single numerical value indicating, how many observations at the end of rt not to include in the fitting process and to reserve for backtesting.
start_pars	the starting parameters for the numerical optimization routine; should be of the same length as the parameter output vector within the output object (also keeping the same order); for NULL, an internally saved default set of values is used; see "Details" for the order of elements; elements should be set with respect to a series rescaled to have sample variance one.
LB	the lower boundaries of the parameters in the numerical optimization routine; should be of the same length as the parameter output vector within the output object (also keeping the same order); for NULL, an internally saved default set of values is used; see "Details" for the order of elements; elements should be set with respect to a series rescaled to have sample variance one.
UB	the upper boundaries of the parameters in the numerical optimization routine; should be of the same length as the parameter output vector within the output object (also keeping the same order); for NULL, an internally saved default set of values is used; see "Details" for the order of elements; elements should be set with respect to a series rescaled to have sample variance one.
control	a list that is passed to control of the function solnp of the package Rsolnp.

`control_nonpar` a list containing changes to the arguments for the hyperparameter estimation algorithm in the nonparametric scale function estimation for `use_nonpar = TRUE`; see "Details" for more information.

`mean_after_nonpar` only for `use_nonpar = TRUE`; considers the unconditional mean of the parametric model part in the QMLE step in a semiparametric model; by default, a zero-mean model is considered for the parametric part in a semiparametric model.

`parallel` only relevant for a (skewed) average Laplace (AL) distribution, i.e. if `cond_dist` in `spec` is set to `cond_dist = "ald"` or `cond_dist = "sald"`; `parallel` is a logical value indicating whether or not the slices for the positive integer-valued parameter of the SM distribution should be fitted in parallel for a speed boost.

`ncores` only relevant for a (skewed) average Laplace (AL) distribution, i.e. if `cond_dist` in `spec` is set to `cond_dist = "ald"` or `cond_dist = "sald"`, and if simultaneously `parallel = TRUE`; `ncores` is a single numeric value indicating the number of cores to use for parallel computations.

`trunc` a positive integer indicating the finite truncation length of the infinite-order polynomials of the infinite-order representations of the long-memory model parts; the character `"none"` is an optional input that specifies that truncation should always be applied back to the first (presample) observation time point, i.e. that maximum length filters should be applied at all times.

`presample` the presample length for initialization (for extended EGARCH- / Log-GARCH-type models only relevant for the FARIMA-part, as series in log-transformed conditional variance are initialized by zero).

`Prange` a two-element vector that indicates the search boundaries for the parameter P in a (skewed) average Laplace distribution.

`skip_vcov` a logical indicating whether or not to skip the computation of the variance-covariance matrix of the parameter estimators and therefore also standard error computation.

Details

Let $\{r_t\}$, with $t \in \mathbb{Z}$ as the time index, be a theoretical time series that follows

$$r_t = \mu + \varepsilon_t \text{ with } \varepsilon_t = \sigma_t \eta_t \text{ and } \eta_t \sim \text{IID}(0,1), \text{ where}$$

$$\sigma_t^2 = \omega + \left[1 - \beta^{-1}(B)\phi(B)(1-B)^d\right]\varepsilon_t^2.$$

Here, $\eta_t \sim \text{IID}(0,1)$ means that the innovations η_t are independent and identically distributed (iid) with mean zero and variance one, whereas $\sigma_t > 0$ are the conditional standard deviations in r_t. Moreover, B is the backshift operator and $\beta(B) = 1 - \sum_{j=1}^{q} \beta_j B^j$, where β_j, $j = 1, 2, \ldots, q$, are real-valued coefficients. Furthermore, $\phi(B) = 1 - \sum_{i=1}^{p} \phi_i B^i$, where ϕ_i, $i = 1, 2, \ldots, p$, are real-valued coefficients. p and q are the model orders definable through the argument `orders`, where p is the first element and q is the second element in the argument. In addition, we have $\mu = E(r_t)$ as a real-valued parameter and $d \in [0,1]$ as the parameter for the level of integration. With $d = 0$ the model reduces to a short-memory GARCH, for $d = 1$ we have a full integration, and for $d \in (0,1)$, we have fractional integration, where $d \in (0, 0.5)$ is usually considered to describe a long-memory process. $\omega > 0$ is the intercept. It is assumed that all β_j and ϕ_i are non-negative. Furthermore, we have $\omega > 0$ as the intercept.

Currently, only a model of orders $p = 1$ with $q = 1$ can be fitted; to ensure the non-negativity of all of the infinite-order coefficient series $\psi(B)$, which in combination with $\omega > 0$ ensures that all the conditional volatilities are greater than zero, we employ inequality constraints ensuring that the first 50 coefficients of the infinite-order ARCH-representation are non-negative as an approximation to ensuring that all of the coefficients are non-negative. To ensure that they are non-negative, one may in theory consider the sufficient conditions mentioned in Bollerslev and Mikkelsen (1996) or Tse (1998), which are however sometimes restrictive, or the simultaneously necessary and sufficient conditions by Conrad and Haag (2006), which are however complex to implement properly.

The truncated infinite order polynomial is computed following the idea by Nielsen and Noel (2021) as is the series of conditional variances for most computational efficiency. To ensure stability of the first fitted in-sample conditional standard deviations, we however use a small, but also adjustable (also to length zero) presample, which may introduce biases into the parameter estimators.

In the current package version, standard errors of parameter estimates are computed from the Hessian at the optimum of the log-likelihood using `hessian`. To ensure numerical stability and applicability to a huge variety of differently scaled data, parametric models are first fitted to data that is scaled to have sample variance 1. Parameter estimates and other quantities are then either retransformed or recalculated afterwards for the original data.

For a conditional average Laplace distribution, an optimal model for each distribution parameter P from 1 to 5 is estimated (assuming that P is then fixed to the corresponding value). Afterwards, P is then estimated by selecting the estimated model among the five fitted models that has the largest log-likelihood. The five models are, by default, fitted simultaneously using parallel programming techniques (see also the arguments `parallel` and `ncores`, which are only relevant for a conditional average Laplace distribution). After the optimal model (including the estimate of P called $\hat{P}$) has been determined, $P = \hat{P}$ is seen as fixed to obtain the standard errors via the Hessian matrix for the estimates of the continuous parameters. A standard error for $\hat{P}$ is therefore not obtained and the ones obtained for the remaining estimates do not account for $\hat{P}$.

An ARMA-FIGARCH or a FARIMA-FIGARCH can be fitted by adjusting the argument `meanspec` correspondingly.

As an alternative, a semiparametric extension of the pure models in the conditional variance can be implemented. If `use_nonpar = TRUE`, `meanspec` is omitted and before fitting a zero-mean model in the conditional volatility following the remaining function arguments, a smooth scale function, i.e. a function representing the unconditional standard deviation over time, is being estimated following the specifications in `nonparspec` and `control_nonpar`. This preliminary step stabilizes the input series `rt`, as long-term changes in the unconditional variance are being estimated and removed before the parametric step using `tsmoothlm`. `control_nonpar` can be adjusted following to make changes to the arguments of `tsmoothlm` for long-memory specifications. These arguments specify settings for the automated bandwidth selection algorithms implemented by this function. By default, we use the settings `pmin = 0`, `pmax = 1`, `qmin = 0`, `qmax = 1`, `InfR = "Nai"`, `bStart = 0.15`, `cb = 0.05`, and `method = "lpr"` for `tsmoothlm`. `locpol_spec` passed to `nonparspec` handles more direct settings of the local polynomial smoother itself. See the documentation for these functions to get a detailed overview of these settings. Assume $\{r_t\}$ to be the observed series, where $t = 1, 2, \ldots, n$, then $r_t^* = r_t - \bar{r}$, with $\bar{r}$ being the arithmetic mean over the observed r_t, is computed and subsequently $y_t = \ln\left[(r_t^*)^2\right]$. The subtraction of $\bar{r}$ is necessary so that r_t^* are all different from zero almost surely. Once y_t are available, its trend $m(x_t)$, with x_t as the rescaled time on the interval $[0, 1]$, is being estimated using `tsmoothlm` and denoted here by $\hat{m}(x_t)$. Then from $\hat{\xi}_t = y_t - \hat{m}(x_t)$ obtain $\hat{C} = -\ln\left\{\sum_{t=1}^n \exp\left(\hat{\xi}_t\right)\right\}$, and obtain the estimated scale function as

$\hat{s}(x_t) = \exp\left[\left(\hat{\mu}(x_t) - \hat{C}\right)/2\right]$. The stabilized / standardized version of the series $\{r_t\}$ is then $\tilde{r}_t = r_t^*/\hat{s}(x_t)$, to which a purely parametric volatility model following the remaining function arguments is then fitted. The estimated volatility at a given time point is then the product of the estimate of the corresponding scale function value and of the estimated conditional standard deviation (following the parametric model part) for that same time point. See for example Feng et al. (2022) or Letmathe et al. (2023) for more information on the semiparametric extension of volatility models.

The order for manual settings of `start_pars`, `LB` and `UB` is crucial. The correct order is: μ, $\text{ar}_1, \ldots, \text{ar}_{p^*}$, $\text{ma}_1, \ldots, \text{ma}_{q^*}$,$D$,$\omega$, ϕ, β, d, shape parameter, skewness parameter. Depending on the exact model specification, parameters irrelevant for the specification at hand should be dropped in `start_pars`, `LB` and `UB`.

Value

An object of S4-class `"fEGarch_fit_figarch"` is returned. It contains the following elements.

`pars`: a named numeric vector with the parameter estimates.

`se`: a named numeric vector with the obtained standard errors in accordance with the parameter estimates.

`vcov_mat`: the variance-covariance matrix of the parameter estimates with named columns and rows.

`rt`: the input object `rt` (or at least the training data, if `n_test` is greater than zero); if `rt` was a `"zoo"` object, the formatting is kept.

`cmeans`: the estimated conditional means; if `rt` was a `"zoo"` or `"ts"` object, the formatting is also applied to `cmeans`.

`sigt`: the estimated conditional standard deviations (or for `use_nonpar = TRUE` the estimated total volatilities, i.e. scale function value times conditional standard deviation); if `rt` was a `"zoo"` object, the formatting is also applied to `sigt`.

`etat`: the obtained residuals; if `rt` was a `"zoo"` object, the formatting is also applied to `etat`.

`orders`: a two-element numeric vector stating the considered model orders.

`cond_dist`: a character value stating the conditional distribution considered in the model fitting.

`long_memo`: a logical value stating whether or not long memory was considered in the model fitting.

`llhood`: the log-likelihood value obtained at the optimal parameter combination.

`inf_criteria`: a named two-element numeric vector with the corresponding AIC (first element) and BIC (second element) of the fitted parametric model part; for purely parametric models, these criteria are valid for the entire model; for semiparametric models, they are only valid for the parametric step and are not valid for the entire model.

`meanspec`: the settings for the model in the conditional mean; is an object of class `"mean_spec"` that is identical to the object passed to the input argument `meanspec`.

`test_obs`: the observations at the end up the input `rt` reserved for testing following `n_test`.

`scale_fun`: the estimated scale function values, if `use_nonpar = TRUE`, otherwise `NULL`; formatting of `rt` is reused.

`nonpar_model`: the estimation object returned by `tsmoothlm` for `use_nonpar = TRUE`.

`trunc`: the input argument `trunc`.

References

- Baillie, R., Bollerslev, T., & Mikkelsen, H. O. (1996). Fractionally integrated generalized autoregressive conditional heteroskedasticity. Journal of Econometrics, 74(1), 3-30. DOI: 10.1016/S0304-4076(95)01749-6.
- Bollerslev, T., & Mikkelsen, H. O. (1996). Modeling and pricing long memory in stock market volatility. Journal of Econometrics, 73(1): 151-184. DOI: 10.1016/0304-4076(95)01736-4.
- Conrad, C., & Haag, B. R. (2006). Inequality constraints in the fractionally integrated GARCH model. Journal of Financial Econometrics, 4(3): 413-449. DOI: 10.1093/jjfinec/nbj015.
- Conrad, C., & Karanasos, M. (2006). The impulse response function of the long memory GARCH process. Economics Letters, 90(1): 34-41. DOI: 10.1016/j.econlet.2005.07.001.
- Feng, Y., Gries, T., Letmathe, S., & Schulz, D. (2022). The smoots Package in R for Semiparametric Modeling of Trend Stationary Time Series. The R Journal, 14(1), 182-195. URL: https://journal.r-project.org/articles/RJ-2022-017/.
- Karanasos, M., Psaradakis, Z., & Sola, M. (2004). On the autocorrelation properties of long-memory GARCH processes. Journal of Time Series Analysis, 25(2): 265-281. DOI: 10.1046/j.0143-9782.2003.00349.x.
- Letmathe, S., Beran, J., & Feng, Y. (2023). An extended exponential SEMIFAR model with application in R. Communications in Statistics - Theory and Methods, 53(22), 7914–7926. DOI: 10.1080/03610926.2023.2276049.
- Nielsen, M. O., & Noel, A. L. (2021). To infinity and beyond: Efficient computation of ARCH(∞) models. Journal of Time Series Analysis, 42(3), 338–354. DOI: 10.1111/jtsa.12570.

Examples

```
window.zoo <- get("window.zoo", envir = asNamespace("zoo"))
rt <- window.zoo(SP500, end = "2002-12-31")
model <- figarch(rt)
model
```

---

figarch_sim *Simulate From FIGARCH Models*

Description

A streamlined simulation function to simulate from fractionally integrated generalized autoregressive conditional heteroskedasticity (FIGARCH) models.

Usage

```
figarch_sim(
 pars = list(mu = 0, ar = numeric(0), ma = numeric(0), D = 0, omega = 4e-04, phi = 0.2,
    beta = 0.4, d = 0.25, df = 10, shape = 2, P = 3, skew = 1),
 cond_dist = c("norm", "std", "ged", "ald", "snorm", "sstd", "sged", "sald"),
 n = 1000,
```

```
  nstart = 5000,
  trunc = "none"
)
```

Arguments

pars	a named list with the parameter specifications; the user can provide a named list with only the settings they would like to adjust relative to the default settings.
cond_dist	a one-element character vector specifying the conditional distribution to consider.
n	the number of observations to return.
nstart	the number of burn-in observations to simulate before the final n values to keep; the first nstart values are not returned; if a dual model, i.e. with model in the conditional mean and in the conditional variance, is considered, two times nstart is considered in the first simulation step in the conditional variance, so that n + nstart values can be fed into the second simulation step for the conditional mean.
trunc	a truncation for the finite-order coefficient series in long-memory models; can either be the character "none" for truncation back to the very first observation at each time point, or to any positive integer for setting the corresponding truncation length of the infinite-order representation polynomial.

Details

See the documentation on figarch for information on the FIGARCH model. This function provides an easy way to simulate from these models.

Value

A list with four elements is returned: rt are the simulated observations, etat are the underlying innovations, sigt are the correspondingly simulated conditional standard deviations, and cmeans are the simulated conditional means. These four elements are formatted as "ts" class time series objects.

Examples

```
sim <- figarch_sim(n = 1000)
mat <- do.call(cbind, sim)
plot(mat, main = "")
```

figjrgarch *FIGJR-GARCH Model Fitting*

Description

Fit a fractionally integrated GJR-GARCH model under the six most common and further conditional distributions to observed data using quasi maximum-likelihood estimation.

Usage

```
figjrgarch(
  rt,
  orders = c(1, 1),
  cond_dist = c("norm", "std", "ged", "ald", "snorm", "sstd", "sged", "sald"),
  drange = c(0, 1),
  meanspec = mean_spec(),
  Drange = c(0, 1),
  nonparspec = locpol_spec(),
  use_nonpar = FALSE,
  n_test = 0,
  start_pars = NULL,
  LB = NULL,
  UB = NULL,
  control = list(),
  control_nonpar = list(),
  mean_after_nonpar = FALSE,
  parallel = TRUE,
  ncores = max(1, future::availableCores() - 1),
  trunc = "none",
  presample = 50,
  Prange = c(1, 5),
  skip_vcov = FALSE
)
```

Arguments

`rt`	the observed series ordered from past to present; can be a numeric vector, a `"zoo"` class time series object, or a `"ts"` class time series object.
`orders`	a two-element numeric vector containing the two model orders p and q (see Details for more information); currently, only the default `orders = c(1, 1)` is supported; other specifications of a two-element numeric vector will lead to `orders = c(1, 1)` being run and a warning message being returned.
`cond_dist`	the conditional distribution to consider as a character object; the default is a conditional normal distribution `"norm"`; available are also, however, a t-distribution (`"std"`), a generalized error distribution (`"ged"`), an average Laplace distribution (`"ald"`), and their four skewed variants (`"snorm"`, `"sstd"`, `"sged"`, `"sald"`).

drange — a two-element numeric vector that gives the boundaries of the search interval for the fractional differencing parameter d in the conditional volatility model part; is overwritten by the settings of the arguments LB and UB.

meanspec — an object of class "mean_spec"; indicates the specifications for the model in the conditional mean.

Drange — a two-element numeric vector that indicates the boundaries of the interval over which to search for the fractional differencing parameter D in a long-memory ARMA-type model in the conditional mean model part; by default, D being searched for on the interval from 0 to $0.5 - 1 \times 10^{-6}$; note that specific settings in the arguments LB and UB overwrite this argument.

nonparspec — an object of class "locpol_spec" returned by locpol_spec; defines the settings of the nonparametric smoothing technique for use_nonpar = TRUE.

use_nonpar — a logical indicating whether or not to implement a semiparametric extension of the volatility model defined through spec; see "Details" for more information.

n_test — a single numerical value indicating, how many observations at the end of rt not to include in the fitting process and to reserve for backtesting.

start_pars — the starting parameters for the numerical optimization routine; should be of the same length as the parameter output vector within the output object (also keeping the same order); for NULL, an internally saved default set of values is used; see "Details" for the order of elements; elements should be set with respect to a series rescaled to have sample variance one.

LB — the lower boundaries of the parameters in the numerical optimization routine; should be of the same length as the parameter output vector within the output object (also keeping the same order); for NULL, an internally saved default set of values is used; see "Details" for the order of elements; elements should be set with respect to a series rescaled to have sample variance one.

UB — the upper boundaries of the parameters in the numerical optimization routine; should be of the same length as the parameter output vector within the output object (also keeping the same order); for NULL, an internally saved default set of values is used; see "Details" for the order of elements; elements should be set with respect to a series rescaled to have sample variance one.

control — a list that is passed to control of the function solnp of the package Rsolnp.

control_nonpar — a list containing changes to the arguments for the hyperparameter estimation algorithm in the nonparametric scale function estimation for use_nonpar = TRUE; see "Details" for more information.

mean_after_nonpar — only for use_nonpar = TRUE; considers the unconditional mean of the parametric model part in the QMLE step in a semiparametric model; by default, a zero-mean model is considered for the parametric part in a semiparametric model.

parallel — only relevant for a (skewed) average Laplace (AL) distribution, i.e. if cond_dist in spec is set to cond_dist = "ald" or cond_dist = "sald"; parallel is a logical value indicating whether or not the slices for the positive integer-valued parameter of the SM distribution should be fitted in parallel for a speed boost.

ncores	only relevant for a (skewed) average Laplace (AL) distribution, i.e. if cond_dist in spec is set to cond_dist = "ald" or cond_dist = "sald", and if simultaneously parallel = TRUE; ncores is a single numeric value indicating the number of cores to use for parallel computations.
trunc	a positive integer indicating the finite truncation length of the infinite-order polynomials of the infinite-order representations of the long-memory model parts; the character "none" is an optional input that specifies that truncation should always be applied back to the first (presample) observation time point, i.e. that maximum length filters should be applied at all times.
presample	the presample length for initialization (for extended EGARCH- / Log-GARCH-type models only relevant for the FARIMA-part, as series in log-transformed conditional variance are initialized by zero).
Prange	a two-element vector that indicates the search boundaries for the parameter P in a (skewed) average Laplace distribution.
skip_vcov	a logical indicating whether or not to skip the computation of the variance-covariance matrix of the parameter estimators and therefore also standard error computation.

Details

Consider a FIGJR-GARCH(p, d, q) (Ayensu et al., 2025) with constant asymmetry term γ regardless of the lag. Let $\{r_t\}$, with $t \in \mathbb{Z}$ as the time index, be a theoretical time series that follows

$$r_t = \mu + \varepsilon_t \text{ with } \varepsilon_t = \sigma_t \eta_t \text{ and } \eta_t \sim \text{IID}(0,1), \text{ where}$$

$$\sigma_t^2 = \omega + \left[1 - \beta^{-1}(B)\phi(B)(1-B)^d\right] \left(|\varepsilon_t| - \gamma\varepsilon_t\right)^2 .$$

Here, $\eta_t \sim \text{IID}(0,1)$ means that the innovations η_t are independent and identically distributed (iid) with mean zero and variance one, whereas $\sigma_t > 0$ are the conditional standard deviations in r_t. Moreover, B is the backshift operator and $\beta(B) = 1 - \sum_{j=1}^{q} \beta_j B^j$, where β_j, $j = 1, 2, \ldots, q$, are real-valued coefficients. Furthermore, $\phi(B) = 1 - \sum_{i=1}^{p} \phi_i B^i$, where ϕ_i, $i = 1, 2, \ldots, p$, are real-valued coefficients. p and q are the model orders definable through the argument orders, where p is the first element and q is the second element in the argument. In addition, we have $\mu = E(r_t)$ as a real-valued parameter, $\gamma \in (-1, 1)$ and $d \in [0, 1]$ as the parameter for the level of integration. With $d = 0$ the model reduces to a short-memory GJR-GARCH, for $d = 1$ we have a full integration, and for $d \in (0, 1)$, we have fractional integration, where $d \in (0, 0.5)$ is considered to describe a long-memory process. $\omega > 0$ is the intercept. It is assumed that all β_j and ϕ_i are non-negative.

The pre-sample values of $\left(|\varepsilon_t| - \gamma\varepsilon_t\right)^2$ are replaced by the in-sample arithmetic mean of

$$\left(|r_t^* - \bar{r}_t^*| - \gamma_0 \left(r_t^* - \bar{r}_t^*\right)\right)^2 ,$$

where r_t^* are considered as the observations and $\bar{r}_t^*$ is the sample mean of r_t^*. An initial value for γ_0 as initial guess for γ is obtained from a previous fitting of a short-memory GJR-GARCH.

The truncated infinite order polynomial is computed following the idea by Nielsen and Noel (2021) as is the series of conditional variances for most computational efficiency. To ensure stability of the first fitted in-sample conditional standard deviations, we however use a small, but also adjustable (also to length zero) presample, which may introduce biases into the parameter estimators.

Currently, only a model of orders $p = 1$ with $q = 1$ can be fitted; to ensure the non-negativity of all of the infinite-order coefficient series $\left[1 - \beta^{-1}(B)\phi(B)(1-B)^d\right]$, which in combination with $\omega > 0$ ensures that all the conditional volatilities are greater than zero, we employ inequality constraints ensuring that the first 50 coefficients of the infinite-order ARCH-representation are non-negative as an approximation to ensuring that all of the coefficients are non-negative. To ensure that they are non-negative, one may in theory consider the sufficient conditions mentioned in Bollerslev and Mikkelsen (1996) or Tse (1998), which are however sometimes restrictive, or the simultaneously necessary and sufficient conditions by Conrad and Haag (2006), which are however complex to implement properly.

See also the reference section for sources on the GJR-GARCH (Glosten et al., 1993), the more general APARCH (Ding et al., 1993) and the FIAPARCH (Tse, 1998) models.

In the current package version, standard errors of parameter estimates are computed from the Hessian at the optimum of the log-likelihood using `hessian`. To ensure numerical stability and applicability to a huge variety of differently scaled data, parametric models are first fitted to data that is scaled to have sample variance 1. Parameter estimates and other quantities are then either retransformed or recalculated afterwards for the original data.

For a conditional average Laplace distribution, an optimal model for each distribution parameter P from 1 to 5 is estimated (assuming that P is then fixed to the corresponding value). Afterwards, P is then estimated by selecting the estimated model among the five fitted models that has the largest log-likelihood. The five models are, by default, fitted simultaneously using parallel programming techniques (see also the arguments `parallel` and `ncores`, which are only relevant for a conditional average Laplace distribution). After the optimal model (including the estimate of P called $\hat{P}$) has been determined, $P = \hat{P}$ is seen as fixed to obtain the standard errors via the Hessian matrix for the estimates of the continuous parameters. A standard error for $\hat{P}$ is therefore not obtained and the ones obtained for the remaining estimates do not account for $\hat{P}$.

An ARMA-FIGJR-GARCH or a FARIMA-FIGJR-GARCH can be fitted by adjusting the argument `meanspec` correspondingly.

As an alternative, a semiparametric extension of the pure models in the conditional variance can be implemented. If `use_nonpar = TRUE`, `meanspec` is omitted and before fitting a zero-mean model in the conditional volatility following the remaining function arguments, a smooth scale function, i.e. a function representing the unconditional standard deviation over time, is being estimated following the specifications in `nonparspec` and `control_nonpar`. This preliminary step stabilizes the input series `rt`, as long-term changes in the unconditional variance are being estimated and removed before the parametric step using `tsmoothlm`. `control_nonpar` can be adjusted following to make changes to the arguments of `tsmoothlm` for long-memory specifications. These arguments specify settings for the automated bandwidth selection algorithms implemented by this function. By default, we use the settings `pmin = 0`, `pmax = 1`, `qmin = 0`, `qmax = 1`, `InfR = "Nai"`, `bStart = 0.15`, `cb = 0.05`, and `method = "lpr"` for `tsmoothlm`. `locpol_spec` passed to `nonparspec` handles more direct settings of the local polynomial smoother itself. See the documentation for these functions to get a detailed overview of these settings. Assume $\{r_t\}$ to be the observed series, where $t = 1, 2, \ldots, n$, then $r_t^* = r_t - \bar{r}$, with $\bar{r}$ being the arithmetic mean over the observed r_t, is computed and subsequently $y_t = \ln\left[(r_t^*)^2\right]$. The subtraction of $\bar{r}$ is necessary so that r_t^* are all different from zero almost surely. Once y_t are available, its trend $m(x_t)$, with x_t as the rescaled time on the interval $[0, 1]$, is being estimated using `tsmoothlm` and denoted here by $\hat{m}(x_t)$. Then from $\hat{\xi}_t = y_t - \hat{m}(x_t)$ obtain $\hat{C} = -\ln\left\{\sum_{t=1}^n \exp\left(\hat{\xi}_t\right)\right\}$, and obtain the estimated scale function as $\hat{s}(x_t) = \exp\left[\left(\hat{\mu}(x_t) - \hat{C}\right)/2\right]$. The stabilized / standardized version of the series $\{r_t\}$ is then $\tilde{r}_t =$

$r_t^*/\hat{s}(x_t)$, to which a purely parametric volatility model following the remaining function arguments is then fitted. The estimated volatility at a given time point is then the product of the estimate of the corresponding scale function value and of the estimated conditional standard deviation (following the parametric model part) for that same time point. See for example Feng et al. (2022) or Letmathe et al. (2023) for more information on the semiparametric extension of volatility models.

The order for manual settings of `start_pars`, `LB` and `UB` is crucial. The correct order is: μ, $\text{ar}_1, \ldots, \text{ar}_{p^*}$, $\text{ma}_1, \ldots, \text{ma}_{q^*}$,$D$,$\omega$, ϕ, β, γ, d, shape parameter, skewness parameter. Depending on the exact model specification, parameters irrelevant for the specification at hand should be dropped in `start_pars`, `LB` and `UB`.

Value

An object of S4-class `"fEGarch_fit_figjrgarch"` is returned. It contains the following elements.

`pars`: a named numeric vector with the parameter estimates.

`se`: a named numeric vector with the obtained standard errors in accordance with the parameter estimates.

`vcov_mat`: the variance-covariance matrix of the parameter estimates with named columns and rows.

`rt`: the input object `rt` (or at least the training data, if `n_test` is greater than zero); if `rt` was a `"zoo"` or `"ts"` object, the formatting is kept.

`cmeans`: the estimated conditional means; if `rt` was a `"zoo"` or `"ts"` object, the formatting is also applied to `cmeans`.

`sigt`: the estimated conditional standard deviations (or for `use_nonpar = TRUE` the estimated total volatilities, i.e. scale function value times conditional standard deviation); if `rt` was a `"zoo"` or `"ts"` object, the formatting is also applied to `sigt`.

`etat`: the obtained residuals; if `rt` was a `"zoo"` or `"ts"` object, the formatting is also applied to `etat`.

`orders`: a two-element numeric vector stating the considered model orders.

`cond_dist`: a character value stating the conditional distribution considered in the model fitting.

`long_memo`: a logical value stating whether or not long memory was considered in the model fitting.

`llhood`: the log-likelihood value obtained at the optimal parameter combination.

`inf_criteria`: a named two-element numeric vector with the corresponding AIC (first element) and BIC (second element) of the fitted parametric model part; for purely parametric models, these criteria are valid for the entire model; for semiparametric models, they are only valid for the parametric step and are not valid for the entire model.

`meanspec`: the settings for the model in the conditional mean; is an object of class `"mean_spec"` that is identical to the object passed to the input argument `meanspec`.

`test_obs`: the observations at the end up the input `rt` reserved for testing following `n_test`.

`scale_fun`: the estimated scale function values, if `use_nonpar = TRUE`, otherwise `NULL`; formatting of `rt` is reused.

`nonpar_model`: the estimation object returned by `tsmoothlm` for `use_nonpar = TRUE`.

`trunc`: the input argument `trunc`.

References

- Ayensu, O. K., Feng, Y., & Schulz, D. (2025). Recent Extensions of Exponential GARCH Models: Theory and Application. Forthcoming preprint, Paderborn University.
- Bollerslev, T., & Mikkelsen, H. O. (1996). Modeling and pricing long memory in stock market volatility. Journal of Econometrics, 73(1): 151-184. DOI: 10.1016/0304-4076(95)01736-4.
- Conrad, C., & Haag, B. R. (2006). Inequality constraints in the fractionally integrated GARCH model. Journal of Financial Econometrics, 4(3): 413-449. DOI: 10.1093/jjfinec/nbj015.
- Ding, Z., Granger, C. W. J., & Engle, R. F. (1993). A long memory property of stock market returns and a new model. Journal of Empirical Finance, 1(1): 83-106. DOI: 10.1016/0927-5398(93)90006-D.
- Feng, Y., Gries, T., Letmathe, S., & Schulz, D. (2022). The smoots Package in R for Semi-parametric Modeling of Trend Stationary Time Series. The R Journal, 14(1), 182-195. URL: https://journal.r-project.org/articles/RJ-2022-017/.
- Glosten, L. R., Jagannathan, R., & Runkle, D. E. (1993). On The Relation between The Expected Value and The Volatility of Nominal Excess Return on stocks. Journal of Finance 48(5), 1779-1801. DOI: 10.1111/j.1540-6261.1993.tb05128.x.
- Letmathe, S., Beran, J., & Feng, Y. (2023). An extended exponential SEMIFAR model with application in R. Communications in Statistics - Theory and Methods, 53(22), 7914–7926. DOI: 10.1080/03610926.2023.2276049.
- Nielsen, M. O., & Noel, A. L. (2021). To infinity and beyond: Efficient computation of ARCH(∞) models. Journal of Time Series Analysis, 42(3), 338-354. DOI: 10.1111/jtsa.12570.
- Tse, Y. K. (1998). The conditional heteroskedasticity of the yen-dollar exchange rate. Journal of Applied Econometrics, 13(1): 49-55. DOI: 10.1002/(SICI)1099-1255(199801/02)13:1<49::AID-JAE459>3.0.CO;2-O.

Examples

```
window.zoo <- get("window.zoo", envir = asNamespace("zoo"))
rt <- window.zoo(SP500, end = "2002-12-31")
model <- figjrgarch(rt)
model
```

`figjrgarch_sim` *Simulate From FIGJR-GARCH Models*

Description

A streamlined simulation function to simulate from FIGJR-GARCH models.

Usage

```
figjrgarch_sim(
 pars = list(mu = 0, ar = numeric(0), ma = numeric(0), D = 0, omega = 4e-04, phi = 0.2,
    beta = 0.4, gamma = 0.1, d = 0.25, df = 10, shape = 2, P = 3, skew = 1),
  cond_dist = c("norm", "std", "ged", "ald", "snorm", "sstd", "sged", "sald"),
  n = 1000,
  nstart = 5000,
  trunc = "none"
)
```

Arguments

pars	a named list with the parameter specifications; the user can provide a named list with only the settings they would like to adjust relative to the default settings.
cond_dist	a one-element character vector specifying the conditional distribution to consider.
n	the number of observations to return.
nstart	the number of burn-in observations to simulate before the final n values to keep; the first nstart values are not returned; if a dual model, i.e. with model in the conditional mean and in the conditional variance, is considered, two times nstart is considered in the first simulation step in the conditional variance, so that n + nstart values can be fed into the second simulation step for the conditional mean.
trunc	a truncation for the finite-order coefficient series in long-memory models; can either be the character "none" for truncation back to the very first observation at each time point, or to any positive integer for setting the corresponding truncation length of the infinite-order representation polynomial.

Details

See the documentation on figjrgarch for information on the FIGJR-GARCH model. This function provides an easy way to simulate from these models.

Value

A list with four elements is returned: rt are the simulated observations, etat are the underlying innovations, sigt are the correspondingly simulated conditional standard deviations, and cmeans are the simulated conditional means. These four elements are formatted as "ts" class time series objects.

Examples

```
sim <- figjrgarch_sim(n = 1000)
mat <- do.call(cbind, sim)
plot(mat, main = "")
```

find_dist *Optimal Distribution Fitting to IID Data*

Description

Given a series supposed to be from independent and identically distributed (iid) random variables, fit all eight distributions of this package to the data using maximum-likelihood estimation (MLE) and select the best one following either the BIC (the default) or the AIC.

Usage

```
find_dist(
  x,
  dists = c("norm", "std", "ged", "ald", "snorm", "sstd", "sged", "sald"),
  fix_mean = NULL,
  fix_sdev = NULL,
  Prange = c(1, 5),
  criterion = c("bic", "aic"),
  skip_vcov = FALSE
)
```

Arguments

x	the vector of iid values to fit distributions to.
dists	a vector with all the distribution abbreviations which should be considered in the selection process; by default, all eight distributions of this package are considered.
fix_mean	a value to fix the unconditional mean of the distribution to; with the default NULL, the unconditional mean is estimated as an extra parameter.
fix_sdev	a value to fix the unconditional standard deviation of the distribution to; with the default NULL, the unconditional standard deviation is estimated as an extra parameter.
Prange	a two-element vector giving the search range for the shape parameter P of the (skewed) average Laplace distribution.
criterion	either "bic" or "aic" to use BIC or AIC as the final selection criterion; by default "bic" is implemented.
skip_vcov	a logical indicating whether or not the computation of the variance-covariance matrix should be skipped.

Details

For information on the method and distributions, we refer the reader to distr_est.

Value

Returns an object of class "fEGarch_distr_est" with various slots representing the estimation results of the selected fitted distribution.

Examples

```
x <- rnorm(2000) * 2.1 + 10.5
find_dist(x)
```

`fit_test_suite,fEGarch_fit-method`
Post-Estimation Fit-Tests

Description

Apply a collection of fit-tests, including a weighted Ljung-Box test for the simple and the squared standardized residuals, a sign-bias test, and an adjusted Pearson goodness-of-fit test.

Usage

```
## S4 method for signature 'fEGarch_fit'
fit_test_suite(
  object,
  args_lbt = list(),
  args_sbt = list(),
  args_goft = list(),
  silent = FALSE,
  ...
)
```

Arguments

`object`	an object of class `"fEGarch_fit"` as returned by the fitting functions of this package like for example `fEGarch`.
`args_lbt`	a list of changes to make to the default argument settings in `ljung_box_test`.
`args_sbt`	a list of changes to make to the default argument settings in `sign_bias_test`.
`args_goft`	a list of changes to make to the default argument settings in `goodn_of_fit_test`.
`silent`	a logical indicating whether or not to print the test results in a well-formatted manner to the console.
`...`	currently without purpose.

Value

Returns a list with the four test results invisibly.

Examples

```
window.zoo <- get("window.zoo", envir = asNamespace("zoo"))
rt <- window.zoo(SP500, end = "2002-12-31")
spec <- fEGarch_spec()
model <- fEGarch(spec, rt)
fit_test_suite(model)
```

`fitgarch` *FITGARCH Model Fitting*

Description

Fit a fractionally integrated TGARCH model under the six most common and further conditional distributions to observed data using quasi maximum-likelihood estimation.

Usage

```
fitgarch(
  rt,
  orders = c(1, 1),
  cond_dist = c("norm", "std", "ged", "ald", "snorm", "sstd", "sged", "sald"),
  drange = c(0, 1),
  meanspec = mean_spec(),
  Drange = c(0, 1),
  nonparspec = locpol_spec(),
  use_nonpar = FALSE,
  n_test = 0,
  start_pars = NULL,
  LB = NULL,
  UB = NULL,
  control = list(),
  control_nonpar = list(),
  mean_after_nonpar = FALSE,
  parallel = TRUE,
  ncores = max(1, future::availableCores() - 1),
  trunc = "none",
  presample = 50,
  Prange = c(1, 5),
  skip_vcov = FALSE
)
```

Arguments

`rt`	the observed series ordered from past to present; can be a numeric vector, a `"zoo"` class time series object, or a `"ts"` class time series object.

`orders` a two-element numeric vector containing the two model orders p and q (see Details for more information); currently, only the default `orders = c(1, 1)` is supported; other specifications of a two-element numeric vector will lead to `orders = c(1, 1)` being run and a warning message being returned.

`cond_dist` the conditional distribution to consider as a character object; the default is a conditional normal distribution `"norm"`; available are also, however, a t-distribution (`"std"`), a generalized error distribution (`"ged"`), an average Laplace distribution (`"ald"`), and their four skewed variants (`"snorm"`, `"sstd"`, `"sged"`, `"sald"`).

`drange` a two-element numeric vector that gives the boundaries of the search interval for the fractional differencing parameter d in the conditional volatility model part; is overwritten by the settings of the arguments `LB` and `UB`.

`meanspec` an object of class "mean_spec"; indicates the specifications for the model in the conditional mean.

`Drange` a two-element numeric vector that indicates the boundaries of the interval over which to search for the fractional differencing parameter D in a long-memory ARMA-type model in the conditional mean model part; by default, D being searched for on the interval from 0 to $0.5 - 1 \times 10^{-6}$; note that specific settings in the arguments `LB` and `UB` overwrite this argument.

`nonparspec` an object of class `"locpol_spec"` returned by `locpol_spec`; defines the settings of the nonparametric smoothing technique for `use_nonpar = TRUE`.

`use_nonpar` a logical indicating whether or not to implement a semiparametric extension of the volatility model defined through `spec`; see "Details" for more information.

`n_test` a single numerical value indicating, how many observations at the end of `rt` not to include in the fitting process and to reserve for backtesting.

`start_pars` the starting parameters for the numerical optimization routine; should be of the same length as the parameter output vector within the output object (also keeping the same order); for `NULL`, an internally saved default set of values is used; see "Details" for the order of elements; elements should be set with respect to a series rescaled to have sample variance one.

`LB` the lower boundaries of the parameters in the numerical optimization routine; should be of the same length as the parameter output vector within the output object (also keeping the same order); for `NULL`, an internally saved default set of values is used; see "Details" for the order of elements; elements should be set with respect to a series rescaled to have sample variance one.

`UB` the upper boundaries of the parameters in the numerical optimization routine; should be of the same length as the parameter output vector within the output object (also keeping the same order); for `NULL`, an internally saved default set of values is used; see "Details" for the order of elements; elements should be set with respect to a series rescaled to have sample variance one.

`control` a list that is passed to `control` of the function `solnp` of the package `Rsolnp`.

`control_nonpar` a list containing changes to the arguments for the hyperparameter estimation algorithm in the nonparametric scale function estimation for `use_nonpar = TRUE`; see "Details" for more information.

`mean_after_nonpar`
: only for `use_nonpar = TRUE`; considers the unconditional mean of the parametric model part in the QMLE step in a semiparametric model; by default, a zero-mean model is considered for the parametric part in a semiparametric model.

`parallel`
: only relevant for a (skewed) average Laplace (AL) distribution, i.e. if `cond_dist` in `spec` is set to `cond_dist = "ald"` or `cond_dist = "sald"`; `parallel` is a logical value indicating whether or not the slices for the positive integer-valued parameter of the SM distribution should be fitted in parallel for a speed boost.

`ncores`
: only relevant for a (skewed) average Laplace (AL) distribution, i.e. if `cond_dist` in `spec` is set to `cond_dist = "ald"` or `cond_dist = "sald"`, and if simultaneously `parallel = TRUE`; `ncores` is a single numeric value indicating the number of cores to use for parallel computations.

`trunc`
: a positive integer indicating the finite truncation length of the infinite-order polynomials of the infinite-order representations of the long-memory model parts; the character `"none"` is an optional input that specifies that truncation should always be applied back to the first (presample) observation time point, i.e. that maximum length filters should be applied at all times.

`presample`
: the presample length for initialization (for extended EGARCH- / Log-GARCH-type models only relevant for the FARIMA-part, as series in log-transformed conditional variance are initialized by zero).

`Prange`
: a two-element vector that indicates the search boundaries for the parameter P in a (skewed) average Laplace distribution.

`skip_vcov`
: a logical indicating whether or not to skip the computation of the variance-covariance matrix of the parameter estimators and therefore also standard error computation.

Details

Consider a FITGARCH(p, d, q) (Ayensu et al., 2025) with constant asymmetry term γ regardless of the lag. Let $\{r_t\}$, with $t \in \mathbb{Z}$ as the time index, be a theoretical time series that follows

$$r_t = \mu + \varepsilon_t \text{ with } \varepsilon_t = \sigma_t \eta_t \text{ and } \eta_t \sim \text{IID}(0,1), \text{ where}$$

$$\sigma_t = \omega + \left[1 - \beta^{-1}(B)\phi(B)(1-B)^d\right](|\varepsilon_t| - \gamma\varepsilon_t).$$

Here, $\eta_t \sim \text{IID}(0,1)$ means that the innovations η_t are independent and identically distributed (iid) with mean zero and variance one, whereas $\sigma_t > 0$ are the conditional standard deviations in r_t. Moreover, B is the backshift operator and $\beta(B) = 1 - \sum_{j=1}^{q} \beta_j B^j$, where β_j, $j = 1, 2, \ldots, q$, are real-valued coefficients. Furthermore, $\phi(B) = 1 - \sum_{i=1}^{p} \phi_i B^i$, where ϕ_i, $i = 1, 2, \ldots, p$, are real-valued coefficients. p and q are the model orders definable through the argument `orders`, where p is the first element and q is the second element in the argument. In addition, we have $\mu = E(r_t)$ as a real-valued parameter, $\gamma \in (-1, 1)$ and $d \in [0, 1]$ as the parameter for the level of integration. With $d = 0$ the model reduces to a short-memory TGARCH, for $d = 1$ we have a full integration, and for $d \in (0, 1)$, we have fractional integration, where $d \in (0, 0.5)$ is considered to describe a long-memory process. $\omega > 0$ is the intercept. It is assumed that all β_j and ϕ_i are non-negative.

The pre-sample values of $(|\varepsilon_t| - \gamma\varepsilon_t)$ are replaced by the in-sample arithmetic mean of

$$(|r_t^* - \bar{r}_t^*| - \gamma_0 (r_t^* - \bar{r}_t^*)),$$

where r_t^* are considered as the observations and $\bar{r}_t^*$ is the sample mean of r_t^*. An initial value for γ_0 as initial guess for γ is obtained from a previous fitting of a short-memory TGARCH.

The truncated infinite order polynomial is computed following the idea by Nielsen and Noel (2021) as is the series of conditional variances for most computational efficiency. To ensure stability of the first fitted in-sample conditional standard deviations, we however use a small, but also adjustable (also to length zero) presample, which may introduce biases into the parameter estimators.

Currently, only a model of orders $p = 1$ with $q = 1$ can be fitted; to ensure the non-negativity of all of the infinite-order coefficient series $\left[1 - \beta^{-1}(B)\phi(B)(1 - B)^d\right]$, which in combination with $\omega > 0$ ensures that all the conditional volatilities are greater than zero, we employ inequality constraints ensuring that the first 50 coefficients of the infinite-order ARCH-representation are non-negative as an approximation to ensuring that all of the coefficients are non-negative. To ensure that they are non-negative, one may in theory consider the sufficient conditions mentioned in Bollerslev and Mikkelsen (1996) or Tse (1998), which are however sometimes restrictive, or the simultaneously necessary and sufficient conditions by Conrad and Haag (2006), which are however complex to implement properly.

See also the reference section for sources on the TGARCH model (Zakoian, 1994), the more general APARCH (Ding et al., 1993) and the FIAPARCH (Tse, 1998) models.

In the current package version, standard errors of parameter estimates are computed from the Hessian at the optimum of the log-likelihood using `hessian`. To ensure numerical stability and applicability to a huge variety of differently scaled data, parametric models are first fitted to data that is scaled to have sample variance 1. Parameter estimates and other quantities are then either retransformed or recalculated afterwards for the original data.

For a conditional average Laplace distribution, an optimal model for each distribution parameter P from 1 to 5 is estimated (assuming that P is then fixed to the corresponding value). Afterwards, P is then estimated by selecting the estimated model among the five fitted models that has the largest log-likelihood. The five models are, by default, fitted simultaneously using parallel programming techniques (see also the arguments `parallel` and `ncores`, which are only relevant for a conditional average Laplace distribution). After the optimal model (including the estimate of P called $\hat{P}$) has been determined, $P = \hat{P}$ is seen as fixed to obtain the standard errors via the Hessian matrix for the estimates of the continuous parameters. A standard error for $\hat{P}$ is therefore not obtained and the ones obtained for the remaining estimates do not account for $\hat{P}$.

An ARMA-FITGARCH or a FARIMA-FITGARCH can be fitted by adjusting the argument `meanspec` correspondingly.

As an alternative, a semiparametric extension of the pure models in the conditional variance can be implemented. If `use_nonpar = TRUE`, `meanspec` is omitted and before fitting a zero-mean model in the conditional volatility following the remaining function arguments, a smooth scale function, i.e. a function representing the unconditional standard deviation over time, is being estimated following the specifications in `nonparspec` and `control_nonpar`. This preliminary step stabilizes the input series `rt`, as long-term changes in the unconditional variance are being estimated and removed before the parametric step using `tsmoothlm`. `control_nonpar` can be adjusted following to make changes to the arguments of `tsmoothlm` for long-memory specifications. These arguments specify settings for the automated bandwidth selection algorithms implemented by this function. By default, we use the settings `pmin = 0`, `pmax = 1`, `qmin = 0`, `qmax = 1`, `InfR = "Nai"`, `bStart = 0.15`, `cb = 0.05`, and `method = "lpr"` for `tsmoothlm`. `locpol_spec` passed to `nonparspec` handles more direct settings of the local polynomial smoother itself. See the documentation for these functions to get a detailed overview of these settings. Assume $\{r_t\}$ to be the observed series, where $t =$

$1, 2, \ldots, n$, then $r_t^* = r_t - \bar{r}$, with $\bar{r}$ being the arithmetic mean over the observed r_t, is computed and subsequently $y_t = \ln\left[\left(r_t^*\right)^2\right]$. The subtraction of $\bar{r}$ is necessary so that r_t^* are all different from zero almost surely. Once y_t are available, its trend $m(x_t)$, with x_t as the rescaled time on the interval $[0, 1]$, is being estimated using `tsmoothlm` and denoted here by $\hat{m}(x_t)$. Then from $\hat{\xi}_t = y_t - \hat{m}(x_t)$ obtain $\hat{C} = -\ln\left\{\sum_{t=1}^{n} \exp\left(\hat{\xi}_t\right)\right\}$, and obtain the estimated scale function as $\hat{s}(x_t) = \exp\left[\left(\hat{\mu}(x_t) - \hat{C}\right)/2\right]$. The stabilized / standardized version of the series $\{r_t\}$ is then $\tilde{r}_t = r_t^*/\hat{s}(x_t)$, to which a purely parametric volatility model following the remaining function arguments is then fitted. The estimated volatility at a given time point is then the product of the estimate of the corresponding scale function value and of the estimated conditional standard deviation (following the parametric model part) for that same time point. See for example Feng et al. (2022) or Letmathe et al. (2023) for more information on the semiparametric extension of volatility models.

The order for manual settings of `start_pars`, `LB` and `UB` is crucial. The correct order is: μ, $\text{ar}_1, \ldots, \text{ar}_{p^*}$, $\text{ma}_1, \ldots, \text{ma}_{q^*}$,$D$,$\omega$, ϕ, β, γ, d, shape parameter, skewness parameter. Depending on the exact model specification, parameters irrelevant for the specification at hand should be dropped in `start_pars`, `LB` and `UB`.

Value

An object of S4-class `"fEGarch_fit_fitgarch"` is returned. It contains the following elements.

`pars`: a named numeric vector with the parameter estimates.

`se`: a named numeric vector with the obtained standard errors in accordance with the parameter estimates.

`vcov_mat`: the variance-covariance matrix of the parameter estimates with named columns and rows.

`rt`: the input object `rt` (or at least the training data, if `n_test` is greater than zero); if `rt` was a `"zoo"` or `"ts"` object, the formatting is kept.

`cmeans`: the estimated conditional means; if `rt` was a `"zoo"` or `"ts"` object, the formatting is also applied to `cmeans`.

`sigt`: the estimated conditional standard deviations (or for `use_nonpar = TRUE` the estimated total volatilities, i.e. scale function value times conditional standard deviation); if `rt` was a `"zoo"` or `"ts"` object, the formatting is also applied to `sigt`.

`etat`: the obtained residuals; if `rt` was a `"zoo"` or `"ts"` object, the formatting is also applied to `etat`.

`orders`: a two-element numeric vector stating the considered model orders.

`cond_dist`: a character value stating the conditional distribution considered in the model fitting.

`long_memo`: a logical value stating whether or not long memory was considered in the model fitting.

`llhood`: the log-likelihood value obtained at the optimal parameter combination.

`inf_criteria`: a named two-element numeric vector with the corresponding AIC (first element) and BIC (second element) of the fitted parametric model part; for purely parametric models, these criteria are valid for the entire model; for semiparametric models, they are only valid for the parametric step and are not valid for the entire model.

`meanspec`: the settings for the model in the conditional mean; is an object of class `"mean_spec"` that is identical to the object passed to the input argument `meanspec`.

`test_obs`: the observations at the end up the input `rt` reserved for testing following `n_test`.

`scale_fun`: the estimated scale function values, if `use_nonpar = TRUE`, otherwise `NULL`; formatting of `rt` is reused.

`nonpar_model`: the estimation object returned by `tsmoothlm` for `use_nonpar = TRUE`.

`trunc`: the input argument `trunc`.

References

- Ayensu, O. K., Feng, Y., & Schulz, D. (2025). Recent Extensions of Exponential GARCH Models: Theory and Application. Forthcoming preprint, Paderborn University.
- Bollerslev, T., & Mikkelsen, H. O. (1996). Modeling and pricing long memory in stock market volatility. Journal of Econometrics, 73(1): 151-184. DOI: 10.1016/0304-4076(95)01736-4.
- Conrad, C., & Haag, B. R. (2006). Inequality constraints in the fractionally integrated GARCH model. Journal of Financial Econometrics, 4(3): 413-449. DOI: 10.1093/jjfinec/nbj015.
- Ding, Z., Granger, C. W. J., & Engle, R. F. (1993). A long memory property of stock market returns and a new model. Journal of Empirical Finance, 1(1): 83-106. DOI: 10.1016/0927-5398(93)90006-D.
- Feng, Y., Gries, T., Letmathe, S., & Schulz, D. (2022). The smoots Package in R for Semi-parametric Modeling of Trend Stationary Time Series. The R Journal, 14(1), 182-195. URL: https://journal.r-project.org/articles/RJ-2022-017/.
- Letmathe, S., Beran, J., & Feng, Y. (2023). An extended exponential SEMIFAR model with application in R. Communications in Statistics - Theory and Methods, 53(22), 7914–7926. DOI: 10.1080/03610926.2023.2276049.
- Nielsen, M. O., & Noel, A. L. (2021). To infinity and beyond: Efficient computation of ARCH(∞) models. Journal of Time Series Analysis, 42(3), 338-354. DOI: 10.1111/jtsa.12570.
- Tse, Y. K. (1998). The conditional heteroskedasticity of the yen-dollar exchange rate. Journal of Applied Econometrics, 13(1): 49-55. DOI: 10.1002/(SICI)1099-1255(199801/02)13:1<49::AID-JAE459>3.0.CO;2-O.
- Zakoian, J.-M. (1994). Threshold heteroskedastic models. Journal of Economic Dynamics and Control, 18(5): 931-955. DOI: 10.1016/0165-1889(94)90039-6.

Examples

```
window.zoo <- get("window.zoo", envir = asNamespace("zoo"))
rt <- window.zoo(SP500, end = "2002-12-31")
model <- fitgarch(rt)
model
```

fitgarch_sim *Simulate From FITGARCH Models*

Description

A streamlined simulation function to simulate from FITGARCH models.

Usage

```
fitgarch_sim(
 pars = list(mu = 0, ar = numeric(0), ma = numeric(0), D = 0, omega = 4e-04, phi = 0.2,
    beta = 0.4, gamma = 0.1, d = 0.25, df = 10, shape = 2, P = 3, skew = 1),
  cond_dist = c("norm", "std", "ged", "ald", "snorm", "sstd", "sged", "sald"),
  n = 1000,
  nstart = 5000,
  trunc = "none"
)
```

Arguments

pars	a named list with the parameter specifications; the user can provide a named list with only the settings they would like to adjust relative to the default settings.
cond_dist	a one-element character vector specifying the conditional distribution to consider.
n	the number of observations to return.
nstart	the number of burn-in observations to simulate before the final n values to keep; the first nstart values are not returned; if a dual model, i.e. with model in the conditional mean and in the conditional variance, is considered, two times nstart is considered in the first simulation step in the conditional variance, so that n + nstart values can be fed into the second simulation step for the conditional mean.
trunc	a truncation for the finite-order coefficient series in long-memory models; can either be the character "none" for truncation back to the very first observation at each time point, or to any positive integer for setting the corresponding truncation length of the infinite-order representation polynomial.

Details

See the documentation on fitgarch for information on the FITGARCH model. This function provides an easy way to simulate from these models.

Value

A list with four elements is returned: rt are the simulated observations, etat are the underlying innovations, sigt are the correspondingly simulated conditional standard deviations, and cmeans are the simulated conditional means. These four elements are formatted as "ts" class time series objects.

Examples

```
sim <- fitgarch_sim(n = 1000)
mat <- do.call(cbind, sim)
plot(mat, main = "")
```

---

fitted,fEGarch_fit-method

Extract Fitted Conditional Means

Description

An alternative to cmeans,fEGarch_fit-method to extract fitted conditional means from an estimation object in this package.

Usage

```
## S4 method for signature 'fEGarch_fit'
fitted(object)

## S4 method for signature 'fEGarch_forecast'
fitted(object)
```

Arguments

object	an object either of class "fEGarch_fit" or "fEGarch_forecast".

Details

Extract fitted conditional means from an estimation object in this package.

Value

The element within the input object with name cmeans is returned. Depending on the element that can be a numeric vector, an object of class "zoo" or a numeric matrix.

Examples

```
window.zoo <- get("window.zoo", envir = asNamespace("zoo"))
rt <- window.zoo(SP500, end = "2002-12-31")
model <- fEGarch(egarch_spec(), rt, n_test = 250)
fitted(model)
```

`fitted_object_generics` *Generics for Accessing Model Estimation Output Elements*

Description

Accessors for the output returned by the main fitting functions of the `fEGarch` package. The generics themselves are without use.

Usage

```
sigt(x)

cmeans(x)

etat(x)

llhood(x)

inf_criteria(x)

pars(x)

se(x)

vcov_mat(x)
```

Arguments

`x`	an object returned by either `fEGarch`, `fiaparch` or `figarch`, etc.

Details

These generics are without direct use. Consider the specific methods based on them.

Value

These generics do not return anything. Inspect the methods based on them for practical purpose.

garch *GARCH Model Fitting*

Description

Fit a standard GARCH model under the six most common conditional distributions (and more) to observed data using quasi maximum-likelihood estimation.

Usage

```
garch(
  rt,
  orders = c(1, 1),
  cond_dist = c("norm", "std", "ged", "ald", "snorm", "sstd", "sged", "sald"),
  meanspec = mean_spec(),
  Drange = c(0, 1),
  nonparspec = locpol_spec(),
  use_nonpar = FALSE,
  n_test = 0,
  start_pars = NULL,
  LB = NULL,
  UB = NULL,
  control = list(),
  control_nonpar = list(),
  mean_after_nonpar = FALSE,
  parallel = TRUE,
  ncores = max(1, future::availableCores() - 1),
  trunc = "none",
  presample = 50,
  Prange = c(1, 5),
  skip_vcov = FALSE
)
```

Arguments

`rt`	the observed series ordered from past to present; can be a numeric vector, a `"zoo"` class time series object, or a `"ts"` class time series object.
`orders`	a two-element numeric vector containing the two model orders p and q (see Details for more information); currently, only the default `orders = c(1, 1)` is supported; other specifications of a two-element numeric vector will lead to `orders = c(1, 1)` being run and a warning message being returned.
`cond_dist`	the conditional distribution to consider as a character object; the default is a conditional normal distribution `"norm"`; available are also, however, a t-distribution (`"std"`), a generalized error distribution (`"ged"`), an average Laplace distribution (`"ald"`), and their four skewed variants (`"snorm"`, `"sstd"`, `"sged"`, `"sald"`).

`meanspec` an object of class "mean_spec"; indicates the specifications for the model in the conditional mean.

`Drange` a two-element numeric vector that indicates the boundaries of the interval over which to search for the fractional differencing parameter D in a long-memory ARMA-type model in the conditional mean model part; by default, D being searched for on the interval from 0 to $0.5 - 1 \times 10^{-6}$; note that specific settings in the arguments `LB` and `UB` overwrite this argument.

`nonparspec` an object of class `"locpol_spec"` returned by `locpol_spec`; defines the settings of the nonparametric smoothing technique for `use_nonpar = TRUE`.

`use_nonpar` a logical indicating whether or not to implement a semiparametric extension of the volatility model defined through `spec`; see "Details" for more information.

`n_test` a single numerical value indicating, how many observations at the end of `rt` not to include in the fitting process and to reserve for backtesting.

`start_pars` the starting parameters for the numerical optimization routine; should be of the same length as the parameter output vector within the output object (also keeping the same order); for `NULL`, an internally saved default set of values is used; see "Details" for the order of elements; elements should be set with respect to a series rescaled to have sample variance one.

`LB` the lower boundaries of the parameters in the numerical optimization routine; should be of the same length as the parameter output vector within the output object (also keeping the same order); for `NULL`, an internally saved default set of values is used; see "Details" for the order of elements; elements should be set with respect to a series rescaled to have sample variance one.

`UB` the upper boundaries of the parameters in the numerical optimization routine; should be of the same length as the parameter output vector within the output object (also keeping the same order); for `NULL`, an internally saved default set of values is used; see "Details" for the order of elements; elements should be set with respect to a series rescaled to have sample variance one.

`control` a list that is passed to `control` of the function `solnp` of the package `Rsolnp`.

`control_nonpar` a list containing changes to the arguments for the hyperparameter estimation algorithm in the nonparametric scale function estimation for `use_nonpar = TRUE`; see "Details" for more information.

`mean_after_nonpar` only for `use_nonpar = TRUE`; considers the unconditional mean of the parametric model part in the QMLE step in a semiparametric model; by default, a zero-mean model is considered for the parametric part in a semiparametric model.

`parallel` only relevant for a (skewed) average Laplace (AL) distribution, i.e. if `cond_dist` in `spec` is set to `cond_dist = "ald"` or `cond_dist = "sald"`; `parallel` is a logical value indicating whether or not the slices for the positive integer-valued parameter of the SM distribution should be fitted in parallel for a speed boost.

`ncores` only relevant for a (skewed) average Laplace (AL) distribution, i.e. if `cond_dist` in `spec` is set to `cond_dist = "ald"` or `cond_dist = "sald"`, and if simultaneously `parallel = TRUE`; `ncores` is a single numeric value indicating the number of cores to use for parallel computations.

`trunc`	a positive integer indicating the finite truncation length of the infinite-order polynomials of the infinite-order representations of the long-memory model parts; the character `"none"` is an optional input that specifies that truncation should always be applied back to the first (presample) observation time point, i.e. that maximum length filters should be applied at all times.
`presample`	the presample length for initialization (for extended EGARCH- / Log-GARCH-type models only relevant for the FARIMA-part, as series in log-transformed conditional variance are initialized by zero).
`Prange`	a two-element vector that indicates the search boundaries for the parameter P in a (skewed) average Laplace distribution.
`skip_vcov`	a logical indicating whether or not to skip the computation of the variance-covariance matrix of the parameter estimators and therefore also standard error computation.

Details

Let $\{r_t\}$, with $t \in \mathbb{Z}$ as the time index, be a theoretical time series that follows

$$r_t = \mu + \varepsilon_t \text{ with } \varepsilon_t = \sigma_t \eta_t \text{ and } \eta_t \sim \text{IID}(0,1), \text{ where}$$

$$\sigma_t^2 = \omega + \sum_{i=1}^{p} \phi_i \varepsilon_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2.$$

Here, $\eta_t \sim \text{IID}(0,1)$ means that the innovations η_t are independent and identically distributed (iid) with mean zero and variance one, whereas $\sigma_t > 0$ are the conditional standard deviations in r_t. ϕ_i, $i = 1, 2, \ldots, p$, and β_j, $j = 1, \ldots, q$, are non-negative coefficients. p and q are the model orders definable through the argument `orders`, where p is the first element and q is the second element in the argument. In addition, we have $\mu = E(r_t)$ as a real-valued parameter. $\omega > 0$ is the intercept. This overall definition is in accordance with Bollerslev (1986).

In the current package version, standard errors of parameter estimates are computed from the Hessian at the optimum of the log-likelihood using `hessian`. To ensure numerical stability and applicability to a huge variety of differently scaled data, parametric models are first fitted to data that is scaled to have sample variance 1. Parameter estimates and other quantities are then either retransformed or recalculated afterwards for the original data.

For a conditional average Laplace distribution, an optimal model for each distribution parameter P from 1 to 5 is estimated (assuming that P is then fixed to the corresponding value). Afterwards, P is then estimated by selecting the estimated model among the five fitted models that has the largest log-likelihood. The five models are, by default, fitted simultaneously using parallel programming techniques (see also the arguments `parallel` and `ncores`, which are only relevant for a conditional average Laplace distribution). After the optimal model (including the estimate of P called $\hat{P}$) has been determined, $P = \hat{P}$ is seen as fixed to obtain the standard errors via the Hessian matrix for the estimates of the continuous parameters. A standard error for $\hat{P}$ is therefore not obtained and the ones obtained for the remaining estimates do not account for $\hat{P}$.

An ARMA-GARCH or a FARIMA-GARCH can be fitted by adjusting the argument `meanspec` correspondingly.

As an alternative, a semiparametric extension of the pure models in the conditional variance can be implemented. If `use_nonpar = TRUE`, `meanspec` is omitted and before fitting a model in the conditional volatility following the remaining function arguments, a smooth scale function, i.e. a function

representing the unconditional standard deviation over time, is being estimated following the specifications in nonparspec and control_nonpar. This preliminary step stabilizes the input series rt, as long-term changes in the unconditional variance are being estimated and removed before the parametric step using tsmooth. control_nonpar can be adjusted following to make changes to the arguments of tsmooth for short-memory specifications. These arguments specify settings for the automated bandwidth selection algorithms implemented by this function. By default, we use the settings InfR = "Nai", bStart = 0.15, cb = 0.05, and method = "lpr" for tsmooth. locpol_spec passed to nonparspec handles more direct settings of the local polynomial smoother itself. See the documentation for these functions to get a detailed overview of these settings. Assume $\{r_t\}$ to be the observed series, where $t = 1, 2, \ldots, n$, then $r_t^* = r_t - \bar{r}$, with $\bar{r}$ being the arithmetic mean over the observed r_t, is computed and subsequently $y_t = \ln\left[\left(r_t^*\right)^2\right]$. The subtraction of $\bar{r}$ is necessary so that r_t^* are all different from zero almost surely. Once y_t are available, its trend $m(x_t)$, with x_t as the rescaled time on the interval $[0, 1]$, is being estimated using tsmooth and denoted here by $\hat{m}(x_t)$. Then from $\hat{\xi}_t = y_t - \hat{m}(x_t)$ obtain $\hat{C} = -\ln\left\{\sum_{t=1}^{n} \exp\left(\hat{\xi}_t\right)\right\}$, and obtain the estimated scale function as $\hat{s}(x_t) = \exp\left[\left(\hat{\mu}(x_t) - \hat{C}\right)/2\right]$. The stabilized / standardized version of the series $\{r_t\}$ is then $\tilde{r}_t = r_t^*/\hat{s}(x_t)$, to which a purely parametric volatility model following the remaining function arguments is then fitted. The estimated volatility at a given time point is then the product of the estimate of the corresponding scale function value and of the estimated conditional standard deviation (following the parametric model part) for that same time point. See for example Feng et al. (2022) or Letmathe et al. (2023) for more information on the semiparametric extension of volatility models.

The order for manual settings of start_pars, LB and UB is crucial. The correct order is: μ, $\text{ar}_1, \ldots, \text{ar}_{p^*}$, $\text{ma}_1, \ldots, \text{ma}_{q^*}$,$D$,$\omega$, $\phi_1, \ldots, \phi_p$, $\beta_1, \ldots, \beta_q$, shape parameter, skewness parameter. Depending on the exact model specification, parameters irrelevant for the specification at hand should be dropped in start_pars, LB and UB.

Value

An object of S4-class "fEGarch_fit_garch" is returned. It contains the following elements.

pars: a named numeric vector with the parameter estimates.

se: a named numeric vector with the obtained standard errors in accordance with the parameter estimates.

vcov_mat: the variance-covariance matrix of the parameter estimates with named columns and rows.

rt: the input object rt (or at least the training data, if n_test is greater than zero); if rt was a "zoo" or "ts" object, the formatting is kept.

cmeans: the estimated conditional means; if rt was a "zoo" or "ts" object, the formatting is also applied to cmeans.

sigt: the estimated conditional standard deviations (or for use_nonpar = TRUE the estimated total volatilities, i.e. scale function value times conditional standard deviation); if rt was a "zoo" or "ts" object, the formatting is also applied to sigt.

etat: the obtained residuals; if rt was a "zoo" or "ts" object, the formatting is also applied to etat.

orders: a two-element numeric vector stating the considered model orders.

`cond_dist`: a character value stating the conditional distribution considered in the model fitting.

`long_memo`: a logical value stating whether or not long memory was considered in the model fitting.

`llhood`: the log-likelihood value obtained at the optimal parameter combination.

`inf_criteria`: a named two-element numeric vector with the corresponding AIC (first element) and BIC (second element) of the fitted parametric model part; for purely parametric models, these criteria are valid for the entire model; for semiparametric models, they are only valid for the parametric step and are not valid for the entire model.

`meanspec`: the settings for the model in the conditional mean; is an object of class `"mean_spec"` that is identical to the object passed to the input argument `meanspec`.

`test_obs`: the observations at the end up the input `rt` reserved for testing following `n_test`.

`scale_fun`: the estimated scale function values, if `use_nonpar = TRUE`, otherwise `NULL`; formatting of `rt` is reused.

`nonpar_model`: the estimation object returned by `tsmooth` for `use_nonpar = TRUE`.

`trunc`: the input argument `trunc`.

References

- Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. Journal of Econometrics, 31(3): 307-327. DOI: 10.1016/0304-4076(86)90063-1.
- Feng, Y., Gries, T., Letmathe, S., & Schulz, D. (2022). The smoots Package in R for Semiparametric Modeling of Trend Stationary Time Series. The R Journal, 14(1), 182-195. URL: https://journal.r-project.org/articles/RJ-2022-017/.
- Letmathe, S., Beran, J., & Feng, Y. (2023). An extended exponential SEMIFAR model with application in R. Communications in Statistics - Theory and Methods, 53(22), 7914–7926. DOI: 10.1080/03610926.2023.2276049.

Examples

```
window.zoo <- get("window.zoo", envir = asNamespace("zoo"))
rt <- window.zoo(SP500, end = "2002-12-31")
model <- garch(rt)
model
```

`garch_sim` *Simulate From GARCH Models*

Description

A streamlined simulation function to simulate from generalized autoregressive conditional heteroskedasticity (GARCH) models.

Usage

```
garch_sim(
 pars = list(mu = 0, ar = numeric(0), ma = numeric(0), D = 0, omega = 4e-04, phi = 0.05,
    beta = 0.8, df = 10, shape = 2, P = 3, skew = 1),
  cond_dist = c("norm", "std", "ged", "ald", "snorm", "sstd", "sged", "sald"),
  n = 1000,
  nstart = 5000,
  trunc = "none"
)
```

Arguments

pars	a named list with the parameter specifications; the user can provide a named list with only the settings they would like to adjust relative to the default settings.
cond_dist	a one-element character vector specifying the conditional distribution to consider.
n	the number of observations to return.
nstart	the number of burn-in observations to simulate before the final n values to keep; the first nstart values are not returned; if a dual model, i.e. with model in the conditional mean and in the conditional variance, is considered, two times nstart is considered in the first simulation step in the conditional variance, so that n + nstart values can be fed into the second simulation step for the conditional mean.
trunc	a truncation for the finite-order coefficient series in long-memory models; can either be the character "none" for truncation back to the very first observation at each time point, or to any positive integer for setting the corresponding truncation length of the infinite-order representation polynomial.

Details

See the documentation on garch for information on the GARCH model. This function provides an easy way to simulate from these models.

Value

A list with four elements is returned: rt are the simulated observations, etat are the underlying innovations, sigt are the correspondingly simulated conditional standard deviations, and cmeans are the simulated conditional means. These four elements are formatted as "ts" class time series objects.

Examples

```
sim <- garch_sim(n = 1000)
mat <- do.call(cbind, sim)
plot(mat, main = "")
```

garchm_estim *General GARCH-Type Model Estimation*

Description

Fit any of the additional short- or long-memory GARCH-type models from the fEGarch package aside from those of the extended EGARCH family.

Usage

```
garchm_estim(
  rt,
 model = c("garch", "gjrgarch", "tgarch", "aparch", "figarch", "figjrgarch", "fitgarch",
    "fiaparch"),
  orders = c(1, 1),
  cond_dist = c("norm", "std", "ged", "ald", "snorm", "sstd", "sged", "sald"),
  drange = c(0, 1),
  meanspec = mean_spec(),
  Drange = c(0, 1),
  nonparspec = locpol_spec(),
  use_nonpar = FALSE,
  n_test = 0,
  start_pars = NULL,
  LB = NULL,
  UB = NULL,
  control = list(),
  control_nonpar = list(),
  mean_after_nonpar = FALSE,
  parallel = TRUE,
  ncores = max(1, future::availableCores() - 1),
  trunc = "none",
  presample = 50,
  Prange = c(1, 5),
  skip_vcov = FALSE
)
```

Arguments

rt	the observed series ordered from past to present; can be a numeric vector, a "zoo" class time series object, or a "ts" class time series object.
model	any character object among "garch", "gjrgarch", "aparch", "tgarch", "figarch", "figjrgarch", "fitgarch"and "fiaparch".
orders	a two-element numeric vector containing the two model orders p and q (see Details for more information); currently, only the default orders = c(1, 1) is supported for long-memory models; other specifications of a two-element numeric vector will lead to orders = c(1, 1) being run and a warning message being returned for long-memory models.

cond_dist — the conditional distribution to consider as a character object; the default is a conditional normal distribution "norm"; available are also, however, a t-distribution ("std"), a generalized error distribution ("ged"), an average Laplace distribution ("ald"), and their four skewed variants ("snorm", "sstd", "sged", "sald").

drange — a two-element numeric vector that gives the boundaries of the search interval for the fractional differencing parameter d in the conditional volatility model part of a long-memory model; is overwritten by the settings of the arguments LB and UB.

meanspec — an object of class "mean_spec"; indicates the specifications for the model in the conditional mean.

Drange — a two-element numeric vector that indicates the boundaries of the interval over which to search for the fractional differencing parameter D in a long-memory ARMA-type model in the conditional mean model part; by default, D being searched for on the interval from 0 to $0.5 - 1 \times 10^{-6}$; note that specific settings in the arguments LB and UB overwrite this argument.

nonparspec — an object of class "locpol_spec" returned by locpol_spec; defines the settings of the nonparametric smoothing technique for use_nonpar = TRUE.

use_nonpar — a logical indicating whether or not to implement a semiparametric extension of the volatility model defined through spec; see "Details" for more information.

n_test — a single numerical value indicating, how many observations at the end of rt not to include in the fitting process and to reserve for backtesting.

start_pars — the starting parameters for the numerical optimization routine; should be of the same length as the parameter output vector within the output object (also keeping the same order); for NULL, an internally saved default set of values is used; elements should be set with respect to a series rescaled to have sample variance one.

LB — the lower boundaries of the parameters in the numerical optimization routine; should be of the same length as the parameter output vector within the output object (also keeping the same order); for NULL, an internally saved default set of values is used; elements should be set with respect to a series rescaled to have sample variance one.

UB — the upper boundaries of the parameters in the numerical optimization routine; should be of the same length as the parameter output vector within the output object (also keeping the same order); for NULL, an internally saved default set of values is used; elements should be set with respect to a series rescaled to have sample variance one.

control — a list that is passed to control of the function solnp of the package Rsolnp.

control_nonpar — a list containing changes to the arguments for the hyperparameter estimation algorithm in the nonparametric scale function estimation for use_nonpar = TRUE; see "Details" for more information.

mean_after_nonpar — only for use_nonpar = TRUE; considers the unconditional mean of the parametric model part in the QMLE step in a semiparametric model; by default, a zero-mean model is considered for the parametric part in a semiparametric model.

`parallel` only relevant for a (skewed) average Laplace (AL) distribution, i.e. if `cond_dist` in `spec` is set to `cond_dist = "ald"` or `cond_dist = "sald"`; `parallel` is a logical value indicating whether or not the slices for the positive integer-valued parameter of the SM distribution should be fitted in parallel for a speed boost.

`ncores` only relevant for a (skewed) average Laplace (AL) distribution, i.e. if `cond_dist` in `spec` is set to `cond_dist = "ald"` or `cond_dist = "sald"`, and if simultaneously `parallel = TRUE`; `ncores` is a single numeric value indicating the number of cores to use for parallel computations.

`trunc` a positive integer indicating the finite truncation length of the infinite-order polynomials of the infinite-order representations of the long-memory model parts; the character `"none"` is an optional input that specifies that truncation should always be applied back to the first (presample) observation time point, i.e. that maximum length filters should be applied at all times.

`presample` the presample length for initialization (for extended EGARCH- / Log-GARCH-type models only relevant for the FARIMA-part, as series in log-transformed conditional variance are initialized by zero).

`Prange` a two-element vector that indicates the search boundaries for the parameter P in a (skewed) average Laplace distribution.

`skip_vcov` a logical indicating whether or not to skip the computation of the variance-covariance matrix of the parameter estimators and therefore also standard error computation.

Details

See the documentation on `garch`, `gjrgarch`, `tgarch`, `aparch`, `figarch`, `figjrgarch`, `fitgarch` and `fiaparch` for more detailed information on the corresponding models and functions selectable through this wrapper function.

Value

An object of S4-class `"fEGarch_fit_garch"`, `"fEGarch_fit_gjrgarch"`, `"fEGarch_fit_tgarch"`, `"fEGarch_fit_aparch"`, `"fEGarch_fit_figarch"`, `"fEGarch_fit_figjrgarch"`, `"fEGarch_fit_fitgarch"` or `"fEGarch_fit_fiaparch"` is returned depending on the selected input for the argument `model`. The object then contains the following elements.

`pars:` a named numeric vector with the parameter estimates.

`se:` a named numeric vector with the obtained standard errors in accordance with the parameter estimates.

`vcov_mat:` the variance-covariance matrix of the parameter estimates with named columns and rows.

`rt:` the input object `rt` (or at least the training data, if `n_test` is greater than zero); if `rt` was a `"zoo"` or `"ts"` object, the formatting is kept.

`cmeans:` the estimated conditional means; if `rt` was a `"zoo"` or `"ts"` object, the formatting is also applied to `cmeans`.

`sigt:` the estimated conditional standard deviations (or for `use_nonpar = TRUE` the estimated total volatilities, i.e. scale function value times conditional standard deviation); if `rt` was a `"zoo"` or `"ts"` object, the formatting is also applied to `sigt`.

`etat`: the obtained residuals; if `rt` was a `"zoo"` or `"ts"` object, the formatting is also applied to `etat`.

`orders`: a two-element numeric vector stating the considered model orders.

`cond_dist`: a character value stating the conditional distribution considered in the model fitting.

`long_memo`: a logical value stating whether or not long memory was considered in the model fitting.

`llhood`: the log-likelihood value obtained at the optimal parameter combination.

`inf_criteria`: a named two-element numeric vector with the corresponding AIC (first element) and BIC (second element) of the fitted parametric model part; for purely parametric models, these criteria are valid for the entire model; for semiparametric models, they are only valid for the parametric step and are not valid for the entire model.

`meanspec`: the settings for the model in the conditional mean; is an object of class `"mean_spec"` that is identical to the object passed to the input argument `meanspec`.

`test_obs`: the observations at the end up the input `rt` reserved for testing following `n_test`.

`scale_fun`: the estimated scale function values, if `use_nonpar = TRUE`, otherwise `NULL`; formatting of `rt` is reused.

`nonpar_model`: the estimation object returned by `tsmooth` or `tsmoothlm` for `use_nonpar = TRUE`.

`trunc`: the input argument `trunc`.

Examples

```
window.zoo <- get("window.zoo", envir = asNamespace("zoo"))
rt <- window.zoo(SP500, end = "2002-12-31")
model <- garchm_estim(rt, model = "garch")
model
```

---

gjrgarch *GJR-GARCH Model Fitting*

Description

Fit a GJR-GARCH model under the six most common and further conditional distributions to observed data using quasi maximum-likelihood estimation.

Usage

```
gjrgarch(
  rt,
  orders = c(1, 1),
  cond_dist = c("norm", "std", "ged", "ald", "snorm", "sstd", "sged", "sald"),
  meanspec = mean_spec(),
  Drange = c(0, 1),
  nonparspec = locpol_spec(),
  use_nonpar = FALSE,
```

```
  n_test = 0,
  start_pars = NULL,
  LB = NULL,
  UB = NULL,
  control = list(),
  control_nonpar = list(),
  mean_after_nonpar = FALSE,
  parallel = TRUE,
  ncores = max(1, future::availableCores() - 1),
  trunc = "none",
  presample = 50,
  Prange = c(1, 5),
  skip_vcov = FALSE
)
```

Arguments

rt	the observed series ordered from past to present; can be a numeric vector, a "zoo" class time series object, or a "ts" class time series object.
orders	a two-element numeric vector containing the two model orders p and q (see Details for more information); currently, only the default orders = c(1, 1) is supported; other specifications of a two-element numeric vector will lead to orders = c(1, 1) being run and a warning message being returned.
cond_dist	the conditional distribution to consider as a character object; the default is a conditional normal distribution "norm"; available are also, however, a t-distribution ("std"), a generalized error distribution ("ged"), an average Laplace distribution ("ald"), and their four skewed variants ("snorm", "sstd", "sged", "sald").
meanspec	an object of class "mean_spec"; indicates the specifications for the model in the conditional mean.
Drange	a two-element numeric vector that indicates the boundaries of the interval over which to search for the fractional differencing parameter D in a long-memory ARMA-type model in the conditional mean model part; by default, D being searched for on the interval from 0 to $0.5 - 1 \times 10^{-6}$; note that specific settings in the arguments LB and UB overwrite this argument.
nonparspec	an object of class "locpol_spec" returned by locpol_spec; defines the settings of the nonparametric smoothing technique for use_nonpar = TRUE.
use_nonpar	a logical indicating whether or not to implement a semiparametric extension of the volatility model defined through spec; see "Details" for more information.
n_test	a single numerical value indicating, how many observations at the end of rt not to include in the fitting process and to reserve for backtesting.
start_pars	the starting parameters for the numerical optimization routine; should be of the same length as the parameter output vector within the output object (also keeping the same order); for NULL, an internally saved default set of values is used; see "Details" for the order of elements; elements should be set with respect to a series rescaled to have sample variance one.

LB	the lower boundaries of the parameters in the numerical optimization routine; should be of the same length as the parameter output vector within the output object (also keeping the same order); for NULL, an internally saved default set of values is used; see "Details" for the order of elements; elements should be set with respect to a series rescaled to have sample variance one.
UB	the upper boundaries of the parameters in the numerical optimization routine; should be of the same length as the parameter output vector within the output object (also keeping the same order); for NULL, an internally saved default set of values is used; see "Details" for the order of elements; elements should be set with respect to a series rescaled to have sample variance one.
control	a list that is passed to control of the function solnp of the package Rsolnp.
control_nonpar	a list containing changes to the arguments for the hyperparameter estimation algorithm in the nonparametric scale function estimation for use_nonpar = TRUE; see "Details" for more information.
mean_after_nonpar	only for use_nonpar = TRUE; considers the unconditional mean of the parametric model part in the QMLE step in a semiparametric model; by default, a zero-mean model is considered for the parametric part in a semiparametric model.
parallel	only relevant for a (skewed) average Laplace (AL) distribution, i.e. if cond_dist in spec is set to cond_dist = "ald" or cond_dist = "sald"; parallel is a logical value indicating whether or not the slices for the positive integer-valued parameter of the SM distribution should be fitted in parallel for a speed boost.
ncores	only relevant for a (skewed) average Laplace (AL) distribution, i.e. if cond_dist in spec is set to cond_dist = "ald" or cond_dist = "sald", and if simultaneously parallel = TRUE; ncores is a single numeric value indicating the number of cores to use for parallel computations.
trunc	a positive integer indicating the finite truncation length of the infinite-order polynomials of the infinite-order representations of the long-memory model parts; the character "none" is an optional input that specifies that truncation should always be applied back to the first (presample) observation time point, i.e. that maximum length filters should be applied at all times.
presample	the presample length for initialization (for extended EGARCH- / Log-GARCH-type models only relevant for the FARIMA-part, as series in log-transformed conditional variance are initialized by zero).
Prange	a two-element vector that indicates the search boundaries for the parameter P in a (skewed) average Laplace distribution.
skip_vcov	a logical indicating whether or not to skip the computation of the variance-covariance matrix of the parameter estimators and therefore also standard error computation.

Details

Let $\{r_t\}$, with $t \in \mathbb{Z}$ as the time index, be a theoretical time series that follows

$$r_t = \mu + \varepsilon_t \text{ with } \varepsilon_t = \sigma_t \eta_t \text{ and } \eta_t \sim \text{IID}(0, 1), \text{ where}$$

$$\sigma_t^2 = \omega + \sum_{i=1}^{p} \phi_i \left(|\varepsilon_{t-i}| - \gamma_i \varepsilon_{t-i}\right)^2 + \sum_{j=1}^{j} \beta_j \sigma_{t-j}^2.$$

Here, $\eta_t \sim \text{IID}(0,1)$ means that the innovations η_t are independent and identically distributed (iid) with mean zero and variance one, whereas $\sigma_t > 0$ are the conditional standard deviations in r_t. Moreover, β_j, $j = 1, 2, \ldots, q$, ϕ_i, $i = 1, 2, \ldots, p$, are real-valued coefficients. p and q are the model orders definable through the argument `orders`, where p is the first element and q is the second element in the argument. In addition, we have $\mu = E(r_t)$ as a real-valued parameter and $\gamma \in (-1, 1)$. $\omega > 0$ is the intercept. It is assumed that all β_j and ϕ_i are non-negative.

See also the reference section for sources on the GJR-GARCH (Glosten et al., 1993), the more general APARCH (Ding et al., 1993) and the FIAPARCH (Tse, 1998) models.

In the current package version, standard errors of parameter estimates are computed from the Hessian at the optimum of the log-likelihood using `hessian`. To ensure numerical stability and applicability to a huge variety of differently scaled data, parametric models are first fitted to data that is scaled to have sample variance 1. Parameter estimates and other quantities are then either retransformed or recalculated afterwards for the original data.

For a conditional average Laplace distribution, an optimal model for each distribution parameter P from 1 to 5 is estimated (assuming that P is then fixed to the corresponding value). Afterwards, P is then estimated by selecting the estimated model among the five fitted models that has the largest log-likelihood. The eight models are, by default, fitted simultaneously using parallel programming techniques (see also the arguments `parallel` and `ncores`, which are only relevant for a conditional average Laplace distribution). After the optimal model (including the estimate of P called $\hat{P}$) has been determined, $P = \hat{P}$ is seen as fixed to obtain the standard errors via the Hessian matrix for the estimates of the continuous parameters. A standard error for $\hat{P}$ is therefore not obtained and the ones obtained for the remaining estimates do not account for $\hat{P}$.

An ARMA-GJR-GARCH or a FARIMA-GJR-GARCH can be fitted by adjusting the argument `meanspec` correspondingly.

As an alternative, a semiparametric extension of the pure models in the conditional variance can be implemented. If `use_nonpar = TRUE`, `meanspec` is omitted and before fitting a model in the conditional volatility following the remaining function arguments, a smooth scale function, i.e. a function representing the unconditional standard deviation over time, is being estimated following the specifications in `nonparspec` and `control_nonpar`. This preliminary step stabilizes the input series `rt`, as long-term changes in the unconditional variance are being estimated and removed before the parametric step using `tsmooth`. `control_nonpar` can be adjusted following to make changes to the arguments of `tsmooth` for long-memory specifications. These arguments specify settings for the automated bandwidth selection algorithms implemented by this function. By default, we use the settings `InfR = "Nai"`, `bStart = 0.15`, `cb = 0.05`, and `method = "lpr"` for `tsmooth`. `locpol_spec` passed to `nonparspec` handles more direct settings of the local polynomial smoother itself. See the documentation for these functions to get a detailed overview of these settings. Assume $\{r_t\}$ to be the observed series, where $t = 1, 2, \ldots, n$, then $r_t^* = r_t - \bar{r}$, with $\bar{r}$ being the arithmetic mean over the observed r_t, is computed and subsequently $y_t = \ln\left[(r_t^*)^2\right]$. The subtraction of $\bar{r}$ is necessary so that r_t^* are all different from zero almost surely. Once y_t are available, its trend $m(x_t)$, with x_t as the rescaled time on the interval $[0, 1]$, is being estimated using `tsmoothlm` and denoted here by $\hat{m}(x_t)$. Then from $\hat{\xi}_t = y_t - \hat{m}(x_t)$ obtain $\hat{C} = -\ln\left\{\sum_{t=1}^{n} \exp\left(\hat{\xi}_t\right)\right\}$, and obtain the estimated scale function as $\hat{s}(x_t) = \exp\left[\left(\hat{\mu}(x_t) - \hat{C}\right)/2\right]$. The stabilized / standardized version of

the series $\{r_t\}$ is then $\tilde{r}_t = r_t^*/\hat{s}(x_t)$, to which a purely parametric volatility model following the remaining function arguments is then fitted. The estimated volatility at a given time point is then the product of the estimate of the corresponding scale function value and of the estimated conditional standard deviation (following the parametric model part) for that same time point. See for example Feng et al. (2022) or Letmathe et al. (2023) for more information on the semiparametric extension of volatility models.

The order for manual settings of `start_pars`, `LB` and `UB` is crucial. The correct order is: μ, $\text{ar}_1, \ldots, \text{ar}_{p^*}$, $\text{ma}_1, \ldots, \text{ma}_{q^*}$,$D$,$\omega$, $\phi_1, \ldots, \phi_p$, $\beta_1, \ldots, \beta_q$, $\gamma_1, \ldots, \gamma_p$, shape parameter, skewness parameter. Depending on the exact model specification, parameters irrelevant for the specification at hand should be dropped in `start_pars`, `LB` and `UB`.

Value

An object of S4-class `"fEGarch_fit_gjrgarch"` is returned. It contains the following elements.

`pars`: a named numeric vector with the parameter estimates.

`se`: a named numeric vector with the obtained standard errors in accordance with the parameter estimates.

`vcov_mat`: the variance-covariance matrix of the parameter estimates with named columns and rows.

`rt`: the input object `rt` (or at least the training data, if `n_test` is greater than zero); if `rt` was a `"zoo"` or `"ts"` object, the formatting is kept.

`cmeans`: the estimated conditional means; if `rt` was a `"zoo"` or `"ts"` object, the formatting is also applied to `cmeans`.

`sigt`: the estimated conditional standard deviations (or for `use_nonpar = TRUE` the estimated total volatilities, i.e. scale function value times conditional standard deviation); if `rt` was a `"zoo"` or `"ts"` object, the formatting is also applied to `sigt`.

`etat`: the obtained residuals; if `rt` was a `"zoo"` or `"ts"` object, the formatting is also applied to `etat`.

`orders`: a two-element numeric vector stating the considered model orders.

`cond_dist`: a character value stating the conditional distribution considered in the model fitting.

`long_memo`: a logical value stating whether or not long memory was considered in the model fitting.

`llhood`: the log-likelihood value obtained at the optimal parameter combination.

`inf_criteria`: a named two-element numeric vector with the corresponding AIC (first element) and BIC (second element) of the fitted parametric model part; for purely parametric models, these criteria are valid for the entire model; for semiparametric models, they are only valid for the parametric step and are not valid for the entire model.

`meanspec`: the settings for the model in the conditional mean; is an object of class `"mean_spec"` that is identical to the object passed to the input argument `meanspec`.

`test_obs`: the observations at the end up the input `rt` reserved for testing following `n_test`.

`scale_fun`: the estimated scale function values, if `use_nonpar = TRUE`, otherwise `NULL`; formatting of `rt` is reused.

`nonpar_model`: the estimation object returned by `tsmooth` for `use_nonpar = TRUE`.

`trunc`: the input argument `trunc`.

References

- Ding, Z., Granger, C. W. J., & Engle, R. F. (1993). A long memory property of stock market returns and a new model. Journal of Empirical Finance, 1(1): 83-106. DOI: 10.1016/0927-5398(93)90006-D.
- Feng, Y., Gries, T., Letmathe, S., & Schulz, D. (2022). The smoots Package in R for Semi-parametric Modeling of Trend Stationary Time Series. The R Journal, 14(1), 182-195. URL: https://journal.r-project.org/articles/RJ-2022-017/.
- Glosten, L. R., Jagannathan, R., & Runkle, D. E. (1993). On The Relation between The Expected Value and The Volatility of Nominal Excess Return on stocks. Journal of Finance 48(5), 1779-1801. DOI: 10.1111/j.1540-6261.1993.tb05128.x.
- Letmathe, S., Beran, J., & Feng, Y. (2023). An extended exponential SEMIFAR model with application in R. Communications in Statistics - Theory and Methods, 53(22), 7914–7926. DOI: 10.1080/03610926.2023.2276049.
- Tse, Y. K. (1998). The conditional heteroskedasticity of the yen-dollar exchange rate. Journal of Applied Econometrics, 13(1): 49-55. DOI: 10.1002/(SICI)1099-1255(199801/02)13:1<49::AID-JAE459>3.0.CO;2-O.

Examples

```
window.zoo <- get("window.zoo", envir = asNamespace("zoo"))
rt <- window.zoo(SP500, end = "2002-12-31")
model <- gjrgarch(rt)
model
```

---

gjrgarch_sim *Simulate From GJR-GARCH Models*

Description

A streamlined simulation function to simulate from GJR-GARCH models.

Usage

```
gjrgarch_sim(
 pars = list(mu = 0, ar = numeric(0), ma = numeric(0), D = 0, omega = 4e-04, phi = 0.05,
    beta = 0.8, gamma = 0.1, df = 10, shape = 2, P = 3, skew = 1),
  cond_dist = c("norm", "std", "ged", "ald", "snorm", "sstd", "sged", "sald"),
  n = 1000,
  nstart = 5000,
  trunc = "none"
)
```

Arguments

pars	a named list with the parameter specifications; the user can provide a named list with only the settings they would like to adjust relative to the default settings.
cond_dist	a one-element character vector specifying the conditional distribution to consider.
n	the number of observations to return.
nstart	the number of burn-in observations to simulate before the final n values to keep; the first nstart values are not returned; if a dual model, i.e. with model in the conditional mean and in the conditional variance, is considered, two times nstart is considered in the first simulation step in the conditional variance, so that n + nstart values can be fed into the second simulation step for the conditional mean.
trunc	a truncation for the finite-order coefficient series in long-memory models; can either be the character "none" for truncation back to the very first observation at each time point, or to any positive integer for setting the corresponding truncation length of the infinite-order representation polynomial.

Details

See the documentation on gjrgarch for information on the GJR-GARCH model. This function provides an easy way to simulate from these models.

Value

A list with four elements is returned: rt are the simulated observations, etat are the underlying innovations, sigt are the correspondingly simulated conditional standard deviations, and cmeans are the simulated conditional means. These four elements are formatted as "ts" class time series objects.

Examples

```
sim <- gjrgarch_sim(n = 1000)
mat <- do.call(cbind, sim)
plot(mat, main = "")
```

---

goodn_of_fit_test,fEGarch_fit-method

Adjusted Pearson Goodness-of-Fit Test for Standardized Model Residuals

---

Description

Consider a probability integral transform on the standardized residuals of a fitted model and apply a Pearson goodness-of-fit test to the binned data using a selection of predefined number of bins.

Usage

```
## S4 method for signature 'fEGarch_fit'
goodn_of_fit_test(object, n_bins = c(20, 30, 40, 50), silent = FALSE, ...)
```

Arguments

object	an object of class "fEGarch_fit" as returned by the fitting functions of this package like for example fEGarch.
n_bins	a numeric vector giving the number of bins to use.
silent	a logical indicating whether or not to print the test results in a well-formatted manner to the console.
...	currently without purpose.

Details

Use a probability integral transform on the standardized residuals of a fitted model. This is then the basis to conduct a Pearson goodness-of-fit chi-square test.

Value

Returns a numeric matrix invisibly.

Examples

```
window.zoo <- get("window.zoo", envir = asNamespace("zoo"))
rt <- window.zoo(SP500, end = "2002-12-31")
spec <- fEGarch_spec()
model <- fEGarch(spec, rt)
goodn_of_fit_test(model)
```

---

inf_criteria,fEGarch_distr_est-method

Methods for Accessing Distribution Estimation Elements

Description

Accessors to access the elements of the same name in output objects returned by either distr_est or its various wrappers like norm_est.

Usage

```
## S4 method for signature 'fEGarch_distr_est'
inf_criteria(x)

## S4 method for signature 'fEGarch_distr_est'
llhood(x)

## S4 method for signature 'fEGarch_distr_est'
pars(x)

## S4 method for signature 'fEGarch_distr_est'
se(x)

## S4 method for signature 'fEGarch_distr_est'
vcov_mat(x)
```

Arguments

x	an object returned by either distr_est or its various wrappers like norm_est.

Details

Convenience methods to access the elements of the same name that can otherwise be accessed via the operator @ within objects that inherit from class "fEGarch_distr_est", which covers objects returned by either distr_est or its various wrappers like norm_est.

Value

The element within the input object of the same name as the method is returned.

Examples

```
x <- rged_s(4000, shape = 1.5) * 2.1 + 3.3
est <- ged_est(x)
inf_criteria(est)
pars(est)
```

---

ljung_box_test *Goodness-of-Fit Test Generics*

Description

Generics for different goodness-of-fit tests. Currently without use.

Usage

```
ljung_box_test(
  object,
  m_max = 20,
  weight_f = function(lag) {
     m <- max(lag)
     (m + 1 - lag)/m
 },
  adj_df = NULL,
  silent = FALSE,
  ...
)

sign_bias_test(object, silent = FALSE, ...)

goodn_of_fit_test(object, n_bins = c(20, 30, 40, 50), silent = FALSE, ...)

fit_test_suite(
  object,
  args_lbt = list(),
  args_sbt = list(),
  args_goft = list(),
  silent = FALSE,
  ...
)
```

Arguments

object	currently without use.
m_max	currently without use.
weight_f	currently without use.
adj_df	currently without use.
silent	currently without use.
...	currently without use.
n_bins	currently without use.
args_lbt	currently without use.
args_sbt	currently without use.
args_goft	currently without use.

Value

The generics are currently without use and do not return anything.

`ljung_box_test,fEGarch_fit-method`
Weighted Ljung-Box Test for Autocorrelation

Description

Apply a (weighted) Ljung-Box test (through the Gamma approximation) to check the standardized residuals of a fitted model from this package for remaining autocorrelation. Two different options allow to check either the simple residuals or the squared residuals.

Usage

```
## S4 method for signature 'fEGarch_fit'
ljung_box_test(
  object,
  m_max = 20,
  weight_f = function(lag) {
     m <- max(lag)
     (m + 1 - lag)/m
 },
  adj_df = NULL,
  silent = FALSE,
  type = c("simple", "squared"),
  ...
)
```

Arguments

`object`	an object `"fEGarch_fit"` as returned by the fitting functions of this package, for example by `fEGarch`.
`m_max`	the maximum lag; tests will be conducted for 1 up to `m_max`.
`weight_f`	a function with argument `lag` stating how weights should be calculated.
`adj_df`	degrees of freedom to adjust for as a number or the default `NULL`, which uses automatic values from the fitted object; for squared residuals `adj_df = 0` is the default and for simple returns, it is the sum of ARMA-parameters.
`silent`	a logical value reflecting whether or not test results should be printed in a well-formatted manner to the console.
`type`	either `"simple"` or `"squared"` for applying the test to simple or squared residuals.
`...`	currently without use.

Value

Returns a numeric matrix invisibly.

Examples

```
window.zoo <- get("window.zoo", envir = asNamespace("zoo"))
rt <- window.zoo(SP500, end = "2002-12-31")
spec <- fEGarch_spec()
model <- fEGarch(spec, rt)
ljung_box_test(model)
```

`locpol_spec` *Specification of Nonparametric Local Polynomial Models*

Description

Specify the nonparametric local polynomial model part in a semiparametric volatility model.

Usage

```
locpol_spec(
  poly_order = c(3, 1),
  kernel_order = c(1, 0, 2, 3),
  boundary_method = c("extend", "shorten"),
  bwidth = NULL
)
```

Arguments

`poly_order`	a single numeric value, in detail either `1` or `3`, that represents the local polynomial order.
`kernel_order`	a single numeric value representing the smoothness of the underlying kernel function; available are `0` (uniform), `1` (epanechnikov), `2` (bisquare), and `3` (triweight).
`boundary_method`	a single character value indicating the smoothing concept to use at boundary points; for `"extend"`, the smoothing window is extended toward the interior by the amount that is lost toward the boundary; for `"shorten"`, there is no compensation in the smoothing window toward the interior for the loss of window width toward the boundary, i.e. the total smoothing window width reduces more and more when getting closer to the first and last time points.
`bwidth`	the smoothing bandwidth; for NULL, i.e. the default, an automated bandwidth selection is employed; otherwise a single numeric value between 0 and 0.5 must be provided.

Details

Assume that a time series $\{r_t\}$, $t = 1, \ldots, n$, follows

$$r_t = \mu + \sigma_t \eta_t,$$

where $\mu = E(r_t)$ and η_t are independent and identically distributed random variables with mean zero and variance one. $\sigma_t > 0$ are total volatilities composed of $s(x_t)$, a smooth, deterministic scale function in the unconditional variance over time (with x_t being the rescaled time on the interval $[0, 1]$), and of λ_t, the conditional standard deviation in $\zeta_t = \lambda_t \eta_t$, so that $\sigma_t = s(x_t)\lambda_t$, or alternatively $r_t = \mu + s(x_t)\zeta_t$. It is assumed that the unconditional variance of the ζ_t is one.

The package's estimation of σ_t is based on the following relations:

$r_t^* = r_t - \mu$, $y_t = \ln\left[(r_t^*)^2\right]$, $C_\mu = E\left[\ln\left(\zeta_t^2\right)\right]$, $m(x_t) = \ln\left[s^2(x_t)\right] + C_\mu$, $\xi_t = \ln\left(\zeta_t^2\right) - C_\mu$, so that

$$y_t = m(x_t) + \xi_t,$$

where m describes a smooth, deterministic trend in y_t. Nonparametric estimation of m and subsequent retransformation allows to obtain a suitable estimate of the scale function s in r_t. Following Feng et al. (2022) and Letmathe et al. (2023), we employ local polynomial regression with automatically selected bandwidth (specially for the time-series context). The function `locpol_spec` allows to set the basic characteristics of the local polynomial estimator considered, like the order of polynomial used in the local regressions, and the kernel function order. After the scale function has been estimated, a zero-mean GARCH-type model can be fitted to the estimated ζ_t.

Depending on whether ζ_t is assumed to follow a short-memory or a long-memory model, the bandwidth selection algorithm in the local polynomial regression step differs and follows either Feng et al. (2022) and Letmathe et al. (2023). The algorithm selection is done automatically based on the remaining model specifications in the call to the estimation functions like `fEGarch`.

Value

An object of class `"locpol_spec"` is returned.

References

- Feng, Y., Gries, T., Letmathe, S., & Schulz, D. (2022). The smoots Package in R for Semiparametric Modeling of Trend Stationary Time Series. The R Journal, 14(1), 182-195. URL: https://journal.r-project.org/articles/RJ-2022-017/.
- Letmathe, S., Beran, J., & Feng, Y. (2023). An extended exponential SEMIFAR model with application in R. Communications in Statistics - Theory and Methods, 53(22), 7914–7926. DOI: 10.1080/03610926.2023.2276049.

Examples

```
locpol_spec()
locpol_spec(poly_order = 1)
locpol_spec(kernel_order = 2)
```

locpol_spec_methods *Accessors for Class* "locpol_spec"

Description

Access and change elements in objects of class "locpol_spec". The method names represent the name of the element to access / manipulate.

Usage

```
## S4 method for signature 'locpol_spec'
poly_order(x)

## S4 method for signature 'locpol_spec'
kernel_order(x)

## S4 method for signature 'locpol_spec'
boundary_method(x)

## S4 method for signature 'locpol_spec'
bwidth(x)

## S4 replacement method for signature 'locpol_spec'
poly_order(x) <- value

## S4 replacement method for signature 'locpol_spec'
kernel_order(x) <- value

## S4 replacement method for signature 'locpol_spec'
boundary_method(x) <- value

## S4 replacement method for signature 'locpol_spec'
bwidth(x) <- value
```

Arguments

x	the input object or object to modify.
value	the value to modify the object x with.

Details

These methods are intended to be used for accessing or manipulating individual elements of objects of class "locpol_spec".

Value

These methods return an object of class "locpol_spec".

Examples

```
test_obj <- locpol_spec()
poly_order(test_obj)
poly_order(test_obj) <- 1
poly_order(test_obj)
```

`loss_functions` *Generic for Loss Function Calculation*

Description

Currently without use. Use the methods derived from this generic.

Usage

```
loss_functions(object, penalty = 1e-04, ...)
```

Arguments

`object`	currently without use.
`penalty`	currently without use.
`...`	currently without use.

Value

The generic itself is currently without use and thus does not return anything.

`loss_functions,fEGarch_risk-method`
Loss Function Calculation

Description

Compute loss function values given log-returns and corresponding value at risk (VaR) and expected shortfall (ES) series.

Usage

```
## S4 method for signature 'fEGarch_risk'
loss_functions(object, penalty = 1e-04, ...)
```

Arguments

`object`	an object of class `"fEGarch_risk"` as returned by the model fitting functions of this package, for example `fEGarch`.
`penalty`	the penalty term to use in the opportunity cost terms.
`...`	currently without use.

Details

Let $n \in \mathbb{N}$ be the number of observations of a (log-)return series $\{r_t\}$, $t = 1, \ldots, n$, and let VaR_t and ES_t be the estimated or forecasted VaR and ES (at some confidence level α) at time t, respectively. Such series are included in an object of class `"fEGarch_risk"`. In the following, a risk measure at time t is simply denoted by RM_t and can either mean VaR_t or ES_t.

Based on a calculated VaR and / or expected shortfall (ES), capital needs to be held back following regulatory rules. Commonly, among many models used for forecasting risk measures that fulfill regulatory conditions, loss functions are computed that also consider opportunity costs in to assess, what model that fulfills regulatory rules minimizes such loss functions. Let $\Omega \geq 0$ be the penalty term.

For all loss functions we have

$$\text{LF}_i = \sum_{t=1}^{n} l_{t,i}, \quad i = 0, 1, 2, 3,$$

as the loss function with

$$l_{t,i} = (\text{RM}_t - r_t)^2, \quad i = 0, 1, 2, 3,$$

for $r_t < \text{RM}_t$. They differ in how the case $r_t \geq \text{RM}_t$ is treated.

The regulatory loss function (`rlf`) uses $l_{t,0} = 0$.

The firm's loss function (Sarma et al., 2003) (`flf`) considers $l_{t,1} = \Omega|\text{RM}_t|$.

The adjusted loss function (Abad et al., 2015) (`alf`) makes use of $l_{t,2} = \Omega|\text{RM}_t - r_t|$.

The corrected loss function (Feng, forthcoming) (`clf`) has $l_{t,3} = \Omega \min(|\text{RM}_t - r_t|, |\text{RM}_t|)$.

Value

Returns a list with the four elements `rlf`, `flf`, `alf` and `clf`, each lists with numeric vector elements `VaR` and `ES`. The four elements correspond to the regulatory loss function, the firm's loss function, the adjusted loss function and the corrected loss function.

References

- Abad, P., Muela, S. B., & Martín, C. L. (2015). The role of the loss function in value-at-risk comparisons. The Journal of Risk Model Validation, 9(1): 1-19. DOI: 10.21314/JRMV.2015.132.
- Sarma, M., Thomas, S., & Shah, A. (2003). Selection of Value-at-Risk models. Journal of Forecasting, 22(4): 337-358. DOI: 10.1002/for.868.

Examples

```
window.zoo <- get("window.zoo", envir = asNamespace("zoo"))
rt <- window.zoo(SP500, end = "2002-12-31")
model <- fEGarch(egarch_spec(), rt, n_test = 250)
fcast <- predict_roll(model)
risk <- measure_risk(fcast, measure = c("VaR", "ES"), level = c(0.95, 0.975, 0.99))
loss_functions(risk)
```

`mean_spec` *Specification of Conditional Mean Models*

Description

Specify the model for the conditional mean in a dual model, where the conditional mean is modelled through an ARMA or a FARIMA model and the conditional standard deviations through a GARCH-type model simultaneously.

Usage

```
mean_spec(orders = c(0, 0), long_memo = FALSE, include_mean = TRUE)
```

Arguments

`orders`	a two-element numeric vector with the model orders; the first element is the autoregressive order p^*, while the second element is the moving-average order q^*.
`long_memo`	a logical value that indicates whether the long-memory version of the model should be considered or not.
`include_mean`	a logical value indicating whether or not to include the constant unconditional mean in the estimation procedure; for `include_mean = FALSE`, the unconditional mean of the series is fixed to zero and not being estimated.

Details

Let $\{y_t\}$, with $t \in \mathbb{Z}$ as the time index, be a theoretical time series that follows

$$\beta(B)(1-B)^D(y_t - \mu) = \alpha(B)r_t,$$

where $\beta(B) = 1 - \sum_{i=1}^{p^*} \beta_i B^i$ and $\alpha(B) = 1 + \sum_{j=1}^{q^*} \alpha_j B^j$ are the AR- and MA-polynomials of orders p^* and q^*, respectively, with real coefficients β_i, $i = 1, \ldots, p^*$, and α_j, $j = 1, \ldots, q^*$. B is the backshift operator. $\beta(B)$ and $\alpha(B)$ are commonly assumed to be without common roots and to have roots outside of the unit circle. Furthermore, μ is a real-valued coefficient representing the unconditional mean in $\{y_t\}$. $D \in [0, 0.5)$ is the fractional differencing parameter. $\{r_t\}$ is a zero-mean (weak) white noise process, for example a member of the GARCH-models (with mean set to zero) presented in this package (see the descriptions in fEGarch_spec, fiaparch, figarch, etc.).

The for $D = 0$, which can be achieved through `long_memo = FALSE`, the formulas above describe an autoregressive moving-average (ARMA) model. For $D \in (0, 0.5)$, they describe a fractionally integrated ARMA (FARIMA) model.

Value

An object of class "mean_spec" is returned.

Examples

```
mean_spec()
mean_spec(orders = c(1, 1))
```

---

mean_spec_methods *Accessors for Class "mean_spec"*

Description

Access and change elements in objects of class "mean_spec". The method names represent the name of the element to access / manipulate.

Usage

```
## S4 method for signature 'mean_spec'
orders(x)

## S4 method for signature 'mean_spec'
long_memo(x)

## S4 method for signature 'mean_spec'
include_mean(x)

## S4 replacement method for signature 'mean_spec'
orders(x) <- value

## S4 replacement method for signature 'mean_spec'
long_memo(x) <- value

## S4 replacement method for signature 'mean_spec'
include_mean(x) <- value
```

Arguments

x	the input object or object to modify.
value	the value to modify the object x with.

Details

These methods are intended to be used for accessing or manipulating individual elements of objects of class "mean_spec".

Value

These methods return an object of class "mean_spec".

Examples

```
test_obj <- mean_spec()
orders(test_obj)
orders(test_obj) <- c(1, 1)
orders(test_obj)
```

---

measure_risk *VaR and ES Computation Following Fitted Models or Forecasts*

Description

Provides easy access to value-at-risk (VaR) and expected shortfall (ES) computation for available models in this package. VaR and ES can either be computed based on (a) fitted conditional means and conditional standard deviations for a training period, or following (b) point forecasts (one-step rolling) of the conditional means and conditional standard deviations.

Usage

```
measure_risk(object, measure = c("VaR", "ES"), level = c(0.975, 0.99), ...)

## S4 method for signature 'fEGarch_fit'
measure_risk(object, measure = c("VaR", "ES"), level = c(0.975, 0.99), ...)

## S4 method for signature 'fEGarch_forecast'
measure_risk(object, measure = c("VaR", "ES"), level = c(0.975, 0.99), ...)

## S4 method for signature 'fEGarch_distr_est'
measure_risk(
  object,
  measure = c("VaR", "ES"),
  level = c(0.975, 0.99),
  test_obs,
  sigt,
  cmeans,
  ...
)
```

Arguments

object	either an object of class "fEGarch_fit" returned by the fitting / estimation functions of this package like returned by for example fEGarch among others, an object of class "fEGarch_forecast" as returned by predict,fEGarch_fit-method

	or predict_roll, fEGarch_fit-method, or an object of class "fEGarch_distr_est" returned by the distribution fitting functions of this package like returned by for example find_dist among others.
measure	a character vector with element "VaR", "ES" or both; indicates, what risk measures should be computed; by default, both VaR and ES are computed.
level	a numeric vector of arbitrary length indicating the confidence levels to compute the VaR and the ES at; by default, the levels 0.975 and 0.99, i.e. 97.5 percent and 99 percent, are considered.
...	currently without use.
test_obs	a series of test observations (only required when object is of class "fEGarch_distr_est").
sigt	a series of forecasted conditional standard deviations for the same time points as test_obs (only required when object is of class "fEGarch_distr_est").
cmeans	a series of forecasted conditional means for the same time points as test_obs (only required when object is of class "fEGarch_distr_est").

Details

Given a fitted model with fitted conditional means and conditional standard deviations or given point forecasts of such series based on a fitted model, the risk measures VaR and ES can be computed (at arbitrary confidence levels) following the conditional loss distribution defined through the estimated / forecasted conditional mean value, the estimated / forecasted conditional standard deviation value, and the assumed conditional distribution (including potential estimates of distribution parameters).

Let $\hat{\mu}_t$ be the estimated / forecasted conditional mean and $\hat{\sigma}_t$ be the estimated / forecasted conditional standard deviation at some time point t. Furthermore, define $\text{VaR}_{\eta,\alpha}$ and $\text{ES}_{\eta,\alpha}$ be the time-invariant VaR and ES, respectively, of some identically but independently distributed random variables η_t with mean zero and variance one. Given that the relationship $r_t = \mu_t + \sigma_t \eta_t$, where μ_t and σ_t are the true conditional mean and conditional standard deviation at time t, is assumed for some return series $\{r_t\}$, the estimated / forecasted conditional VaR and ES of r_t are simply

$$\widehat{\text{VaR}}_{r,\alpha}(t) = \hat{\mu}_t + \hat{\sigma}_t \text{VaR}_{\eta,\alpha} \quad \text{and} \quad \widehat{\text{ES}}_{r,\alpha}(t) = \hat{\mu}_t + \hat{\sigma}_t \text{ES}_{\eta,\alpha}.$$

This definition holds, when losses and therefore also $\text{VaR}_{\eta,\alpha}(t)$ and $\text{ES}_{\eta,\alpha}(t)$ (for common α such as $\alpha = 0.975$ or $\alpha = 0.99$) are considered to be negative in sign.

Define

$$\text{VaR}_{\eta,\alpha} = f_\eta^{-1}(1-\alpha) \quad \text{and} \quad \text{ES}_{\eta,\alpha} = (1-\alpha)^{-1} \int_\alpha^1 \text{VaR}_{\eta,x} dx,$$

which also need to be estimated for some distributions, if a distribution parameter needed to be estimated. f in the previous formula is the cumulative distribution function of the random variables η_t. Therefore, $f_\eta^{-1}(1-\alpha)$ returns the quantile of the innovation distribution at level $1-\alpha$.

In some cases, when rolling one-step forecasts of the conditional standard deviations and the conditional means were obtained following a nonparametric approach, for example through neural networks or similar approaches, VaR and ES are not directly to be calculated because distribution assumptions have not been made. If an object that is a fitted distribution to the model's standardized in-sample residuals is provided, and if also test observations as well as forecasted conditional standard deviations and conditional means for the test time points are passed to the method, VaR and ES will be computed using the fitted distribution in object. Note that object must be of class

"fEGarch_distr_est". A natural selection of object is the output of find_dist, which returns the best fitted model among a normal distribution, a t-distribution, a generalized error distribution, an average Laplace distribution, and their skewed variants, following either BIC (the default) or AIC. It is recommended to then set fix_mean = 0 and fix_sdev = 1 in the call to find_dist to reflect the known property that the residuals are assumed to be estimated from innovations with mean zero and variance one.

Value

The S4 methods all return an object of class "fEGarch_risk" with elements measures, observations and model. observations is the observed series at the time points, for which the risk measures are calculated. measures is a list with elements VaR and ES, distinguishing between computed VaR and ES values. These elements again are list with named elements representing the various computed series. model is the fitted model or a list of multiple fitted model objects.

Examples

```
# In-sample
window.zoo <- get("window.zoo", envir = asNamespace("zoo"))
rt <- window.zoo(SP500, end = "2002-12-31")
model <- fEGarch(egarch_spec(), rt)
risk <- measure_risk(model, measure = c("VaR", "ES"), level = c(0.95, 0.975, 0.99))
risk

# Out-of-sample rolling point forecasts
window.zoo <- get("window.zoo", envir = asNamespace("zoo"))
rt <- window.zoo(SP500, end = "2002-12-31")
model2 <- fEGarch(egarch_spec(), rt, n_test = 250)
# parallel = FALSE only for better compatibility with CRAN checks;
# should be set to TRUE for speed boost
fcast <- predict_roll(model2, parallel = FALSE)
risk2 <- measure_risk(fcast, measure = c("VaR", "ES"), level = c(0.95, 0.975, 0.99))
risk2

# Use some model to obtain rolling point forecasts of
# the conditional mean and the conditional standard deviation for
# some test period; in practice, this will not be from a GARCH-type
# model, because it is parametric and includes a distribution assumption,
# but instead from some nonparametric model
window.zoo <- get("window.zoo", envir = asNamespace("zoo"))
rt <- window.zoo(SP500, end = "2005-12-31")
model <- fEGarch(egarch_spec(), rt, n_test = 250)
# parallel = FALSE only for better compatibility with CRAN checks;
# should be set to TRUE for speed boost
fc <- model %>% predict_roll(refit_after = 50, parallel = FALSE)

test_obs <- model@test_obs   # Test observations
sigt <- fc@sigt              # Conditional volatility forecasts
cmeans <- fc@cmeans          # Conditional mean forecasts

resids <- model@etat         # In-sample standardized residuals
```

```
# Given 'test_obs', 'sigt', 'cmeans' and 'resids', we can now
# compute the VaR and ES forecasts for the test period

dist <- find_dist(resids, fix_mean = 0, fix_sdev = 1)
dist

risk <- dist %>%
 measure_risk(test_obs = test_obs, sigt = sigt, cmeans = cmeans)

plot(risk, which = 0.975)
```

`plot` *S4 Plot Generic*

Description

Imported from base R.

Usage

```
plot(x, y, ...)
```

Arguments

`x`	see base R `plot` documentation.
`y`	see base R `plot` documentation.
`...`	see base R `plot` documentation.

Value

Returns nothing to the console but creates a plot in the plot window.

`plot,fEGarch_fit,ANY-method`
Plot Method for Showing Fitting Step Results

Description

This is method for producing various plots of the estimation results returned by this package.

Usage

```
## S4 method for signature 'fEGarch_fit,ANY'
plot(x, y = NULL, which = NULL, ...)
```

Arguments

x	an object returned by the fitting functions of this package, for example by fEGarch.
y	for compatibility but without use.
which	various plots can be selected either via a keyword or a number; enter "returns" or 1 to show a plot of the input training series; enter "means" or 2 to show the fitted conditional means; enter "stand_deviations" or 3 to show the fitted conditional standard deviations; use "residuals" or 4 to show the standardized residuals following the fitted model; the default is which = NULL which then lets you select a plot interactively in the R console.
...	further arguments to pass to plot.

Details

Create predefined standard plots of the estimation objects returned by the fEGarch package. Plots are created in the base R plot style. The type of plot can be chosen either interactively from the console, or the argument which can be used to directly select the kind of plot to create (see also the description of the argument which) within the function call.

Value

A graphic is created in the plots windows, the function itself, however, returns NULL.

Author(s)

- Dominik Schulz (Research Assistant) (Department of Economics, Paderborn University), Author and Package Creator

Examples

```
window.zoo <- get("window.zoo", envir = asNamespace("zoo"))
rt <- window.zoo(SP500, end = "2002-12-31")
# Pure conditional volatility model
spec <- fEGarch_spec()
model <- fEGarch(spec, rt)
plot(model, which = 3)
```

---

plot,fEGarch_risk,ANY-method

Plotting of Risk Measure Results (Base R)

Description

Plot risk measure results returned by measure_risk as a points-over-threshold plot in style of base R plots.

Usage

```
## S4 method for signature 'fEGarch_risk,ANY'
plot(x, y = NULL, which = NULL, ...)
```

Arguments

x	an object returned by measure_risk.
y	for compatibility but without use.
which	one of the levels of VaR and ES saved in object, usually either 0.975 or 0.99 by default.
...	without use.

Value

Returns nothing but produces a base R plot in the plot window.

Examples

```
window.zoo <- get("window.zoo", envir = asNamespace("zoo"))
rt <- window.zoo(SP500, end = "2003-12-31")

egarch_spec() %>%
 fEGarch(rt = rt, n_test = 250) %>%
 predict_roll() %>%
 measure_risk() %>%
 plot(which = 0.99)
```

---

poly_order *Generics for Nonparametric Smoothing Setting Adjustments*

Description

Generics to build methods on for adjusting certain settings in context of nonparametric smoothing. The generics are currently without use.

Usage

```
poly_order(x)

kernel_order(x)

boundary_method(x)

bwidth(x)
```

```
poly_order(x) <- value

kernel_order(x) <- value

boundary_method(x) <- value

bwidth(x) <- value
```

Arguments

x	the generics are currently without use.
value	the generics are currently without use.

Value

The generics do not work on their own and thus do not return anything.

---

predict,fEGarch_fit-method

Multistep and Rolling Point Forecasts

Description

Given a fitted model object from this package, conduct either multistep point forecasts of the conditional means and the conditional standard deviations into the future or rolling point forecasts of arbitrary step size of these quantities for a future test set.

Usage

```
## S4 method for signature 'fEGarch_fit'
predict(object, n.ahead = 10, trunc = NULL, ...)

## S4 method for signature 'fEGarch_fit'
predict_roll(
  object,
  step_size = 1,
  trunc = NULL,
  refit_after = NULL,
  steady_window = FALSE,
  parallel = TRUE,
  ncores = max(1, future::availableCores() - 1),
  fitting_args = list(),
  ...
)
```

Arguments

object	an object of class "fEGarch_fit", i.e. an object returned by either fEGarch, fiaparch or figarch, etc.; for predict_roll, the slot @test_obs of the fitted model object should not be NULL.
n.ahead	a single numeric value indicating how far into the future the multistep point forecasts should be produced.
trunc	the truncation setting for the infinite-order polynomial of long-memory model parts; the default uses the setting from the fitted input object object.
...	currently without use and included for compatibility with generics.
step_size	the step size of the rolling point forecasts; by default, step_size = 1 is employed, i.e. for the immediately subsequent observation time point for the entire test set.
refit_after	a positive integer indicatinmg the number of observations of the test set, after which the model should be refitted regularly; for the default NULL no refitting is done.
steady_window	a logical; if FALSE, the refitting will use all observations back to the first observation time point; for TRUE, the number of training observations is held constant, so that the window of training observations is shifted at every refitting.
parallel	a logical, indicating whether the models and forecasts should be produced in parallel for a speed boost.
ncores	the number of cores to use for parallel computations.
fitting_args	a list with further fitting arguments for any of the main fitting functions from this package; useful in the refitting, when other settings than the default need to be implemented.

Details

Use predict to compute multistep point forecasts (of the conditional mean and of the conditional standard deviation) into the future. Let n be the number of observations of the data, to which a model was fitted. Then multistep point forecasts are produced for all future time points from n + 1 to n + n.ahead.

Otherwise, if data was reserved for testing when creating object, e.g. through the use of the argument n_test in the corresponding functions, compute rolling point forecasts over the test set using predict_roll. step_size then determines the forecasting horizon for the rolling point forecasts. For example, step_size = 1, i.e. the default, computes one-step rolling point forecasts, whereas for example step_size = 10 computes ten-step rolling point forecasts (starting at the tenth test time point).

Refitting of models during the rolling point forecast procedure is available with package version 1.0.4 for GARCH-type models. refit_after defines after how many test observations the model should be refitted every time. steady_window addresses whether in refitting models the training observations should always start at the very first observation time point available, i.e. steady_window = FALSE, or whether the initial number of training observations should be kept throughout when refitting, so that the window width of training observations is kept but shifted when refitting, i.e. steady_window = TRUE. The behavior of predict_roll from package version 1.0.3 or earlier, i.e. obtaining rolling forecasts without model refitting, can be reproduced by setting refit_after = NULL.

Value

Returns an object of class "fEGarch_forecast" that has the two slots @sigt and @cmeans representing the forecasted conditional standard deviations and conditional means, respectively. If the training series saved in object has a special time series formatting like "zoo" or "ts", the formatting is adopted accordingly to these numeric output series. A third slot @model is either the fitted model input object object (when refitting is disabled) or a list as a collection of fitted model objects of class "fEGarch_fit" (when refitting is enabled).

Examples

```
window.zoo <- get("window.zoo", envir = asNamespace("zoo"))
rt <- window.zoo(SP500, end = "2002-12-31")

# Multistep forecasting (EGARCH with cond. normal distr.)
model1 <- fEGarch(spec = egarch_spec(), rt)
fcast1 <- predict(model1, n.ahead = 10)
fcast1

# Rolling one-step forecasts (EGARCH with cond. normal distr.)
# (Parallel computation disabled for better compatibility with
# CRAN checks)
model2 <- fEGarch(spec = egarch_spec(), rt, n_test = 250)
fcast2 <- predict_roll(model2, step_size = 1, refit_after = 50,
                       parallel = FALSE)
fcast2
```

---

predict_roll *Generics for Forecasts*

Description

The generics are themselves without use.

Usage

```
predict_roll(
  object,
  step_size = 1,
  trunc = NULL,
  refit_after = NULL,
  steady_window = FALSE,
  parallel = TRUE,
  ncores = max(1, future::availableCores() - 1),
  fitting_args = list(),
  ...
)
```

Arguments

object	the generics are currently without use.
step_size	the generics are currently without use.
trunc	the generics are currently without use.
refit_after	the generics are currently without use.
steady_window	the generics are currently without use.
parallel	the generics are currently without use.
ncores	the generics are currently without use.
fitting_args	the generics are currently without use.
...	the generics are currently without use.

Value

The generics do not work themselves and therefore do not return anything.

---

residuals,fEGarch_fit-method

Extract Standardized Residuals

Description

An alternative to etat,fEGarch_fit-method to extract standardized residuals from an estimation object in this package.

Usage

```
## S4 method for signature 'fEGarch_fit'
residuals(object)
```

Arguments

object	an object either of class "fEGarch_fit" or "fEGarch_forecast".

Details

Extract fitted standardized residuals from an estimation object in this package.

Value

The element within the input object with name etat is returned. Depending on the element that can be a numeric vector, an object of class "zoo" or a numeric matrix.

Examples

```
window.zoo <- get("window.zoo", envir = asNamespace("zoo"))
rt <- window.zoo(SP500, end = "2002-12-31")
model <- fEGarch(egarch_spec(), rt, n_test = 250)
residuals(model)
```

---

rugarch_wrappers *Wrapper Functions for Selected* rugarch *GARCH Models*

Description

Easy to use functions for fitting selected GARCH-type models from the widely-known rugarch package by Ghalanos (2024). These functions are meant as an easy to use way to compare the main results from rugarch to the newly established models in the fEGarch package and are by no means considered to be a replacement of rugarch.

Usage

```
aparch_ru(
  rt,
  orders = c(1, 1),
  cond_dist = c("norm", "std", "ged", "snorm", "sstd", "sged"),
  meanspec = mean_spec(),
  nonparspec = locpol_spec(),
  use_nonpar = FALSE,
  n_test = 0,
  start_pars = NULL,
  control_nonpar = list()
)

gjrgarch_ru(
  rt,
  orders = c(1, 1),
  cond_dist = c("norm", "std", "ged", "snorm", "sstd", "sged"),
  meanspec = mean_spec(),
  nonparspec = locpol_spec(),
  use_nonpar = FALSE,
  n_test = 0,
  start_pars = NULL,
  control_nonpar = list()
)

egarch_ru(
  rt,
  orders = c(1, 1),
  cond_dist = c("norm", "std", "ged", "snorm", "sstd", "sged"),
```

```
  meanspec = mean_spec(),
  nonparspec = locpol_spec(),
  use_nonpar = FALSE,
  n_test = 0,
  start_pars = NULL,
  control_nonpar = list()
)
```

Arguments

rt	the input time series to fit the model to ordered from past to present; can also be a "zoo" class object or a "ts" class object.
orders	the ARCH and GARCH orders of the model as a two-element numeric vector.
cond_dist	a single-element character vector with the conditional distribution to consider.
meanspec	an object of class "mean_spec"; indicates the specifications for the model in the conditional mean.
nonparspec	an object of class "locpol_spec" returned by locpol_spec; defines the settings of the nonparametric smoothing technique for use_nonpar = TRUE.
use_nonpar	a logical indicating whether or not to implement a semiparametric extension of the volatility model defined through spec; see "Details" for more information.
n_test	a single numerical value indicating, how many observations at the end of rt not to include in the fitting process and to reserve for backtesting.
start_pars	a named list with starting parameters.
control_nonpar	a list containing changes to the arguments for the hyperparameter estimation algorithm in the nonparametric scale function estimation for use_nonpar = TRUE; see "Details" for more information.

Details

For most details, please see the documentation of the rugarch package (Ghalanos, 2024).

These functions also provide an extension, so that a nonparametric, smooth scale function in the unconditional standard deviation can be estimated before the parametric step. If use_nonpar = TRUE, meanspec is omitted and before fitting a zero-mean model in the conditional volatility following the remaining function arguments, a smooth scale function, i.e. a function representing the unconditional standard deviation over time, is being estimated following the specifications in nonparspec and control_nonpar. This preliminary step stabilizes the input series rt, as long-term changes in the unconditional variance are being estimated and removed before the parametric step using tsmooth. control_nonpar can be adjusted following to make changes to the arguments of tsmooth for short-memory specifications. These arguments specify settings for the automated bandwidth selection algorithms implemented by this function. By default, we use the settings Mcf = "NP", InfR = "Nai", bStart = 0.15, bvc = "Y", cb = 0.05, and method = "lpr" for tsmooth. locpol_spec passed to nonparspec handles more direct settings of the local polynomial smoother itself. See the documentation for these functions to get a detailed overview of these settings. Assume $\{r_t\}$ to be the observed series, where $t = 1, 2, \ldots, n$, then $r_t^* = r_t - \bar{r}$, with $\bar{r}$ being the arithmetic mean over the observed r_t, is computed and subsequently $y_t = \ln\left[(r_t^*)^2\right]$. The subtraction of $\bar{r}$ is necessary so that r_t^* are all different from zero almost surely. Once y_t are available, its

trend $m(x_t)$, with x_t as the rescaled time on the interval $[0, 1]$, is being estimated using tsmoothlm and denoted here by $\hat{m}(x_t)$. Then from $\hat{\xi}_t = y_t - \hat{m}(x_t)$ obtain $\hat{C} = -\ln\left\{\sum_{t=1}^{n} \exp\left(\hat{\xi}_t\right)\right\}$, and obtain the estimated scale function as $\hat{s}(x_t) = \exp\left[\left(\hat{\mu}(x_t) - \hat{C}\right)/2\right]$. The stabilized / standardized version of the series $\{r_t\}$ is then $\tilde{r}_t = r_t^*/\hat{s}(x_t)$, to which a purely parametric volatility model following the remaining function arguments is then fitted. The estimated volatility at a given time point is then the product of the estimate of the corresponding scale function value and of the estimated conditional standard deviation (following the parametric model part) for that same time point. See for example Feng et al. (2022) or Letmathe et al. (2023) for more information on the semiparametric extension of volatility models.

Value

A list with the following named elements is returned.

`pars`: a named numeric vector with the parameter estimates.

`se`: a named numeric vector with the obtained standard errors in accordance with the parameter estimates.

`vcov_mat`: the variance-covariance matrix of the parameter estimates with named columns and rows.

`rt`: the input object `rt` (or at least the training data, if `n_test` is greater than zero); if `rt` was a `"zoo"` or `"ts"` object, the formatting is kept.

`cmeans`: the estimated conditional means; if `rt` was a `"zoo"` or `"ts"` object, the formatting is also applied to `cmeans`.

`sigt`: the estimated conditional standard deviations (or for `use_nonpar = TRUE` the estimated total volatilities, i.e. scale function value times conditional standard deviation); if `rt` was a `"zoo"` or `"ts"` object, the formatting is also applied to `sigt`.

`etat`: the obtained residuals; if `rt` was a `"zoo"` or `"ts"` object, the formatting is also applied to `etat`.

`orders`: a two-element numeric vector stating the considered model orders.

`cond_dist`: a character value stating the conditional distribution considered in the model fitting.

`llhood`: the log-likelihood value obtained at the optimal parameter combination.

`inf_criteria`: a named two-element numeric vector with the corresponding AIC (first element) and BIC (second element) of the fitted model.

`rugarch_model`: the estimation object returned by `ugarchfit()` of the `rugarch` package (Ghalanos, 2024).

`meanspec`: the settings for the model in the conditional mean; is an object of class `"mean_spec"` that is identical to the object passed to the input argument `meanspec`.

`test_obs`: the observations at the end up the input `rt` reserved for testing following `n_test`.

`scale_fun`: the estimated scale function values, if `use_nonpar = TRUE`, otherwise `NULL`; formatting of `rt` is reused.

`nonpar_model`: the estimation object returned by tsmoothlm for `use_nonpar = TRUE`.

References

- Feng, Y., Gries, T., Letmathe, S., & Schulz, D. (2022). The smoots Package in R for Semi-parametric Modeling of Trend Stationary Time Series. The R Journal, 14(1), 182-195. URL: https://journal.r-project.org/articles/RJ-2022-017/.
- Ghalanos, A. (2024). rugarch: Univariate GARCH models. R package version 1.5-3. DOI: 10.32614/CRAN.package.rugarch.
- Letmathe, S., Beran, J., & Feng, Y. (2023). An extended exponential SEMIFAR model with application in R. Communications in Statistics - Theory and Methods, 53(22), 7914–7926. DOI: 10.1080/03610926.2023.2276049.

Examples

```
est <- gjrgarch_ru(SP500)
est@pars
est@se
plot(est@sigt)
```

---

show,fEGarch_fit_egarch-method

Show Method for Estimation Output

Description

Display estimation results from the EGARCH family in a convenient way in the console.

Usage

```
## S4 method for signature 'fEGarch_fit_egarch'
show(object)

## S4 method for signature 'fEGarch_fit_loggarch'
show(object)

## S4 method for signature 'fEGarch_fit_aparch'
show(object)

## S4 method for signature 'fEGarch_fit_gjrgarch'
show(object)

## S4 method for signature 'fEGarch_fit_tgarch'
show(object)

## S4 method for signature 'fEGarch_fit_garch'
show(object)
```

```
## S4 method for signature 'fEGarch_fit_fiaparch'
show(object)

## S4 method for signature 'fEGarch_fit_figjrgarch'
show(object)

## S4 method for signature 'fEGarch_fit_fitgarch'
show(object)

## S4 method for signature 'fEGarch_fit_figarch'
show(object)

## S4 method for signature 'fEGarch_fit_rugarch_wrapper'
show(object)

## S4 method for signature 'fEGarch_distr_est'
show(object)
```

Arguments

object	an object returned by one of this package's fitting functions, for example fEGarch, fiaparch, figarch, etc.

Value

Nothing is returned. Results are printed in the R-console.

---

sigma,fEGarch_fit-method

Extract Fitted Conditional Standard Deviations

Description

An alternative to sigt,fEGarch_fit-method to extract fitted conditional standard deviations from an estimation object in this package.

Usage

```
## S4 method for signature 'fEGarch_fit'
sigma(object)

## S4 method for signature 'fEGarch_forecast'
sigma(object)
```

Arguments

object	an object either of class "fEGarch_fit" or "fEGarch_forecast".

Details

Extract fitted conditional standard deviations from an estimation object in this package.

Value

The element within the input object with name `sigt` is returned. Depending on the element that can be a numeric vector, an object of class "zoo" or a numeric matrix.

Examples

```
window.zoo <- get("window.zoo", envir = asNamespace("zoo"))
rt <- window.zoo(SP500, end = "2002-12-31")
model <- fEGarch(egarch_spec(), rt, n_test = 250)
sigma(model)
```

---

sign_bias_test,fEGarch_fit-method
Sign Bias Test

Description

Apply a sign bias test to check the standardized residuals of a fitted model from this package for remaining significant sign effects.

Usage

```
## S4 method for signature 'fEGarch_fit'
sign_bias_test(object, silent = FALSE, ...)
```

Arguments

`object`	an object `"fEGarch_fit"` as returned by the fitting functions of this package, for example by `fEGarch`.
`silent`	a logical value reflecting whether or not test results should be printed in a well-formatted manner to the console.
`...`	currently without use.

Value

Returns a numeric matrix invisibly.

Examples

```
window.zoo <- get("window.zoo", envir = asNamespace("zoo"))
rt <- window.zoo(SP500, end = "2002-12-31")
spec <- fEGarch_spec()
model <- fEGarch(spec, rt)
sign_bias_test(model)
```

`sim_functions` *Sampling Functions for Innovations*

Description

Draw random samples from a normal distribution, t-distribution, generalized error distribution, or their skewed variants (all standardized to have mean zero and variance one).

Usage

```
rnorm_s(n)

rstd_s(n, df = 10000)

rged_s(n, shape = 2)

rsged_s(n, shape = 2, skew = 1)

rald_s(n, P = 8)

rsnorm_s(n, skew = 1)

rsstd_s(n, df = 10000, skew = 1)

rsald_s(n, P = 8, skew = 1)
```

Arguments

`n`	the number of observations to draw.
`df`	the degrees of freedom for a (skewed) t-distribution.
`shape`	the shape parameter for a (skewed) generalized error distribution.
`skew`	the skewness parameter in the skewed distributions.
`P`	the number of Laplace distributions (minus 1) to derive the arithmetic mean from as the basis for a (skewed) average Laplace (AL) distribution distribution.

Details

Draw random samples from a normal distribution, t-distribution, generalized error distribution, an average Laplace distribution, or their skewed variants (all standardized to have mean zero and variance one).

Value

These functions return a numeric vector of length n.

Examples

```
rnorm_s(10)

rsstd_s(10, df = 7, skew = 0.9)
```

SP500 *Daily Log-Returns of the S&P 500*

Description

A "zoo" object that contains the daily log-returns of the S&P 500 (Standard and Poor's 500) index from January 04, 2000, until November 30, 2024.

Usage

```
SP500
```

Format

An object of class zoo of length 6267.

Source

The data was obtained from Yahoo Finance (accessed: 2024-12-01) and then transformed into a time series "zoo" object using R.

https://finance.yahoo.com/quote/^GSPC/

---

spec_generics *Generics for Model Specification Accessors*

Description

A collection of generics for accessors of the model specification objects.

Usage

```
orders(x)

powers(x)

long_memo(x)

modulus(x)

cond_dist(x)

include_mean(x)

orders(x) <- value

powers(x) <- value

long_memo(x) <- value

modulus(x) <- value

cond_dist(x) <- value

include_mean(x) <- value
```

Arguments

x	the input object or object to modify.
value	the value to modify the object x with.

Details

These generics are intended to provide a basis to construct methods for adjusting the the output of model specification objects like egarch_spec().

Value

These generics themselves do not return anything and are just the foundation for more sophisticated methods.

---

spec_methods *Accessors for Classes* "base_garch_spec" *and* "egarch_spec"

Description

Access and change elements in objects of class "egarch_spec". The method names represent the name of the element to access / manipulate.

Usage

```
## S4 method for signature 'base_garch_spec'
orders(x)

## S4 method for signature 'egarch_type_spec'
powers(x)

## S4 method for signature 'base_garch_spec'
long_memo(x)

## S4 method for signature 'egarch_type_spec'
modulus(x)

## S4 method for signature 'base_garch_spec'
cond_dist(x)

## S4 replacement method for signature 'base_garch_spec'
orders(x) <- value

## S4 replacement method for signature 'egarch_type_spec'
powers(x) <- value

## S4 replacement method for signature 'base_garch_spec'
long_memo(x) <- value

## S4 replacement method for signature 'egarch_type_spec'
modulus(x) <- value

## S4 replacement method for signature 'base_garch_spec'
cond_dist(x) <- value
```

Arguments

x	the input object or object to modify.
value	the value to modify the object x with.

Details

These methods are intended to be used for accessing or manipulating individual elements of objects of class "egarch_spec".

Value

These methods either return an object of class "egarch_type_spec" or "loggarch_type_spec" or the corresponding element of object of such class objects.

Examples

```
test_obj <- egarch_spec()
```

```
orders(test_obj)
orders(test_obj) <- c(2, 1)
orders(test_obj)
```

---

submodel-specs *EGARCH Family Submodel Specification*

Description

Wrappers of `fEGarch_spec()` that create specifications of specific submodels of the broad EGARCH family.

Usage

```
megarch_spec(
  orders = c(1, 1),
  cond_dist = c("norm", "std", "ged", "ald", "snorm", "sstd", "sged", "sald")
)

fimegarch_spec(
  orders = c(1, 1),
  cond_dist = c("norm", "std", "ged", "ald", "snorm", "sstd", "sged", "sald")
)

egarch_spec(
  orders = c(1, 1),
  cond_dist = c("norm", "std", "ged", "ald", "snorm", "sstd", "sged", "sald")
)

fiegarch_spec(
  orders = c(1, 1),
  cond_dist = c("norm", "std", "ged", "ald", "snorm", "sstd", "sged", "sald")
)

mloggarch_spec(
  orders = c(1, 1),
  cond_dist = c("norm", "std", "ged", "ald", "snorm", "sstd", "sged", "sald")
)

fimloggarch_spec(
  orders = c(1, 1),
  cond_dist = c("norm", "std", "ged", "ald", "snorm", "sstd", "sged", "sald")
)

loggarch_spec(
  orders = c(1, 1),
```

```
  cond_dist = c("norm", "std", "ged", "ald", "snorm", "sstd", "sged", "sald")
)

filoggarch_spec(
  orders = c(1, 1),
  cond_dist = c("norm", "std", "ged", "ald", "snorm", "sstd", "sged", "sald")
)
```

Arguments

orders	a two-element numeric vector with the model orders.
cond_dist	a character value stating the underlying conditional distribution to consider; available are a normal distribution ("norm"), a t-distribution ("std"), a generalized error distribution ("ged"), an average Laplace distribution ("ald") and the skewed versions of them ("snorm", "sstd", "sged", "sald").

Details

Available are shortcut specification functions for EGARCH egarch_spec(), FIEGARCH fiegarch_spec(), MEGARCH megarch_spec(), Log-GARCH loggarch_spec(), FILog-GARCH filoggarch_spec(), MLog-GARCH mloggarch_spec(), FIMEGARCH fimegarch_spec() and FIMLog-GARCH fimloggarch_spec().

The following descriptions are following the descriptions in the documentation of the more general fEGarch_spec(). Please go there first to understand the following descriptions on the arguments of fEGarch_spec() to obtain these wrappers.

EGARCH:

model_type = "egarch", long_memo = FALSE, powers = c(1, 1), modulus = c(FALSE, FALSE)

FIEGARCH:

model_type = "egarch", long_memo = TRUE, powers = c(1, 1), modulus = c(FALSE, FALSE)

MEGARCH:

model_type = "egarch", long_memo = FALSE, powers = c(0, 1), modulus = c(TRUE, FALSE)

Log-GARCH:

model_type = "loggarch", long_memo = FALSE

FILog-GARCH:

model_type = "loggarch", long_memo = TRUE

MLog-GARCH:

model_type = "egarch", long_memo = FALSE, powers = c(0, 0), modulus = c(TRUE, TRUE)

FIMLog-GARCH:

model_type = "egarch", long_memo = TRUE, powers = c(0, 0), modulus = c(TRUE, TRUE)

FIMEGARCH:

model_type = "egarch", long_memo = TRUE, powers = c(0, 1), modulus = c(TRUE, FALSE)

Value

Depending on the spec-fun function, either an object of class "egarch-type-spec" or "loggarch-type-spec" is returned.

Examples

```
spec <- megarch_spec(cond_dist = "std")
```

---

tgarch *TGARCH Model Fitting*

Description

Fit a TGARCH model under the six most common and further conditional distributions to observed data using quasi maximum-likelihood estimation.

Usage

```
tgarch(
  rt,
  orders = c(1, 1),
  cond_dist = c("norm", "std", "ged", "ald", "snorm", "sstd", "sged", "sald"),
  meanspec = mean_spec(),
  Drange = c(0, 1),
  nonparspec = locpol_spec(),
  use_nonpar = FALSE,
  n_test = 0,
  start_pars = NULL,
  LB = NULL,
  UB = NULL,
  control = list(),
  control_nonpar = list(),
  mean_after_nonpar = FALSE,
  parallel = TRUE,
  ncores = max(1, future::availableCores() - 1),
  trunc = "none",
  presample = 50,
  Prange = c(1, 5),
  skip_vcov = FALSE
)
```

Arguments

rt	the observed series ordered from past to present; can be a numeric vector, a "zoo" class time series object, or a "ts" class time series object.

`orders` a two-element numeric vector containing the two model orders p and q (see Details for more information); currently, only the default `orders = c(1, 1)` is supported; other specifications of a two-element numeric vector will lead to `orders = c(1, 1)` being run and a warning message being returned.

`cond_dist` the conditional distribution to consider as a character object; the default is a conditional normal distribution `"norm"`; available are also, however, a t-distribution (`"std"`), a generalized error distribution (`"ged"`), an average Laplace distribution (`"ald"`), and their four skewed variants (`"snorm"`, `"sstd"`, `"sged"`, `"sald"`).

`meanspec` an object of class "mean_spec"; indicates the specifications for the model in the conditional mean.

`Drange` a two-element numeric vector that indicates the boundaries of the interval over which to search for the fractional differencing parameter D in a long-memory ARMA-type model in the conditional mean model part; by default, D being searched for on the interval from 0 to $0.5 - 1 \times 10^{-6}$; note that specific settings in the arguments `LB` and `UB` overwrite this argument.

`nonparspec` an object of class `"locpol_spec"` returned by `locpol_spec`; defines the settings of the nonparametric smoothing technique for `use_nonpar = TRUE`.

`use_nonpar` a logical indicating whether or not to implement a semiparametric extension of the volatility model defined through `spec`; see "Details" for more information.

`n_test` a single numerical value indicating, how many observations at the end of `rt` not to include in the fitting process and to reserve for backtesting.

`start_pars` the starting parameters for the numerical optimization routine; should be of the same length as the parameter output vector within the output object (also keeping the same order); for `NULL`, an internally saved default set of values is used; see "Details" for the order of elements; elements should be set with respect to a series rescaled to have sample variance one.

`LB` the lower boundaries of the parameters in the numerical optimization routine; should be of the same length as the parameter output vector within the output object (also keeping the same order); for `NULL`, an internally saved default set of values is used; see "Details" for the order of elements; elements should be set with respect to a series rescaled to have sample variance one.

`UB` the upper boundaries of the parameters in the numerical optimization routine; should be of the same length as the parameter output vector within the output object (also keeping the same order); for `NULL`, an internally saved default set of values is used; see "Details" for the order of elements; elements should be set with respect to a series rescaled to have sample variance one.

`control` a list that is passed to `control` of the function `solnp` of the package `Rsolnp`.

`control_nonpar` a list containing changes to the arguments for the hyperparameter estimation algorithm in the nonparametric scale function estimation for `use_nonpar = TRUE`; see "Details" for more information.

`mean_after_nonpar`
only for `use_nonpar = TRUE`; considers the unconditional mean of the parametric model part in the QMLE step in a semiparametric model; by default, a zero-mean model is considered for the parametric part in a semiparametric model.

parallel	only relevant for a (skewed) average Laplace (AL) distribution, i.e. if cond_dist in spec is set to cond_dist = "ald" or cond_dist = "sald"; parallel is a logical value indicating whether or not the slices for the positive integer-valued parameter of the SM distribution should be fitted in parallel for a speed boost.
ncores	only relevant for a (skewed) average Laplace (AL) distribution, i.e. if cond_dist in spec is set to cond_dist = "ald" or cond_dist = "sald", and if simultaneously parallel = TRUE; ncores is a single numeric value indicating the number of cores to use for parallel computations.
trunc	a positive integer indicating the finite truncation length of the infinite-order polynomials of the infinite-order representations of the long-memory model parts; the character "none" is an optional input that specifies that truncation should always be applied back to the first (presample) observation time point, i.e. that maximum length filters should be applied at all times.
presample	the presample length for initialization (for extended EGARCH- / Log-GARCH-type models only relevant for the FARIMA-part, as series in log-transformed conditional variance are initialized by zero).
Prange	a two-element vector that indicates the search boundaries for the parameter P in a (skewed) average Laplace distribution.
skip_vcov	a logical indicating whether or not to skip the computation of the variance-covariance matrix of the parameter estimators and therefore also standard error computation.

Details

Let $\{r_t\}$, with $t \in \mathbb{Z}$ as the time index, be a theoretical time series that follows

$$r_t = \mu + \varepsilon_t \text{ with } \varepsilon_t = \sigma_t \eta_t \text{ and } \eta_t \sim \text{IID}(0,1), \text{ where}$$

$$\sigma_t = \omega + \sum_{i=1}^{p} \phi_i \left(|\varepsilon_{t-i}| - \gamma_i \varepsilon_{t-i} \right) + \sum_{j=1}^{j} \beta_j \sigma_{t-j}.$$

Here, $\eta_t \sim \text{IID}(0,1)$ means that the innovations η_t are independent and identically distributed (iid) with mean zero and variance one, whereas $\sigma_t > 0$ are the conditional standard deviations in r_t. Moreover, β_j, $j = 1, 2, \ldots, q$, ϕ_i, $i = 1, 2, \ldots, p$, are real-valued coefficients. p and q are the model orders definable through the argument orders, where p is the first element and q is the second element in the argument. In addition, we have $\mu = E(r_t)$ as a real-valued parameter and $\gamma \in (-1, 1)$. $\omega > 0$ is the intercept. It is assumed that all β_j and ϕ_i are non-negative.

See also the reference section for sources on the TGARCH model (Zakoian, 1994), the more general APARCH (Ding et al., 1993) and the FIAPARCH (Tse, 1998) models.

In the current package version, standard errors of parameter estimates are computed from the Hessian at the optimum of the log-likelihood using hessian. To ensure numerical stability and applicability to a huge variety of differently scaled data, parametric models are first fitted to data that is scaled to have sample variance 1. Parameter estimates and other quantities are then either retransformed or recalculated afterwards for the original data.

For a conditional average Laplace distribution, an optimal model for each distribution parameter P from 1 to 5 is estimated (assuming that P is then fixed to the corresponding value). Afterwards, P is then estimated by selecting the estimated model among the five fitted models that has the largest

log-likelihood. The eight models are, by default, fitted simultaneously using parallel programming techniques (see also the arguments `parallel` and `ncores`, which are only relevant for a conditional average Laplace distribution). After the optimal model (including the estimate of P called $\hat{P}$) has been determined, $P = \hat{P}$ is seen as fixed to obtain the standard errors via the Hessian matrix for the estimates of the continuous parameters. A standard error for $\hat{P}$ is therefore not obtained and the ones obtained for the remaining estimates do not account for $\hat{P}$.

An ARMA-TGARCH or a FARIMA-TGARCH can be fitted by adjusting the argument `meanspec` correspondingly.

As an alternative, a semiparametric extension of the pure models in the conditional variance can be implemented. If `use_nonpar = TRUE`, `meanspec` is omitted and before fitting a model in the conditional volatility following the remaining function arguments, a smooth scale function, i.e. a function representing the unconditional standard deviation over time, is being estimated following the specifications in `nonparspec` and `control_nonpar`. This preliminary step stabilizes the input series `rt`, as long-term changes in the unconditional variance are being estimated and removed before the parametric step using `tsmooth`. `control_nonpar` can be adjusted following to make changes to the arguments of `tsmooth` for long-memory specifications. These arguments specify settings for the automated bandwidth selection algorithms implemented by this function. By default, we use the settings `InfR = "Nai"`, `bStart = 0.15`, `cb = 0.05`, and `method = "lpr"` for `tsmooth`. `locpol_spec` passed to `nonparspec` handles more direct settings of the local polynomial smoother itself. See the documentation for these functions to get a detailed overview of these settings. Assume $\{r_t\}$ to be the observed series, where $t = 1, 2, \ldots, n$, then $r_t^* = r_t - \bar{r}$, with $\bar{r}$ being the arithmetic mean over the observed r_t, is computed and subsequently $y_t = \ln\left[(r_t^*)^2\right]$. The subtraction of $\bar{r}$ is necessary so that r_t^* are all different from zero almost surely. Once y_t are available, its trend $m(x_t)$, with x_t as the rescaled time on the interval $[0, 1]$, is being estimated using `tsmoothlm` and denoted here by $\hat{m}(x_t)$. Then from $\hat{\xi}_t = y_t - \hat{m}(x_t)$ obtain $\hat{C} = -\ln\left\{\sum_{t=1}^{n} \exp\left(\hat{\xi}_t\right)\right\}$, and obtain the estimated scale function as $\hat{s}(x_t) = \exp\left[\left(\hat{\mu}(x_t) - \hat{C}\right)/2\right]$. The stabilized / standardized version of the series $\{r_t\}$ is then $\tilde{r}_t = r_t^*/\hat{s}(x_t)$, to which a purely parametric volatility model following the remaining function arguments is then fitted. The estimated volatility at a given time point is then the product of the estimate of the corresponding scale function value and of the estimated conditional standard deviation (following the parametric model part) for that same time point. See for example Feng et al. (2022) or Letmathe et al. (2023) for more information on the semiparametric extension of volatility models.

The order for manual settings of `start_pars`, `LB` and `UB` is crucial. The correct order is: μ, $\text{ar}_1, \ldots, \text{ar}_{p^*}$, $\text{ma}_1, \ldots, \text{ma}_{q^*}$,$D$,$\omega$, $\phi_1, \ldots, \phi_p$, $\beta_1, \ldots, \beta_q$, $\gamma_1, \ldots, \gamma_p$, shape parameter, skewness parameter. Depending on the exact model specification, parameters irrelevant for the specification at hand should be dropped in `start_pars`, `LB` and `UB`.

Value

An object of S4-class `"fEGarch_fit_tgarch"` is returned. It contains the following elements.

`pars`**:** a named numeric vector with the parameter estimates.

`se`**:** a named numeric vector with the obtained standard errors in accordance with the parameter estimates.

`vcov_mat`**:** the variance-covariance matrix of the parameter estimates with named columns and rows.

`rt`: the input object `rt` (or at least the training data, if `n_test` is greater than zero); if `rt` was a `"zoo"` or `"ts"` object, the formatting is kept.

`cmeans`: the estimated conditional means; if `rt` was a `"zoo"` or `"ts"` object, the formatting is also applied to `cmeans`.

`sigt`: the estimated conditional standard deviations (or for `use_nonpar = TRUE` the estimated total volatilities, i.e. scale function value times conditional standard deviation); if `rt` was a `"zoo"` or `"ts"` object, the formatting is also applied to `sigt`.

`etat`: the obtained residuals; if `rt` was a `"zoo"` or `"ts"` object, the formatting is also applied to `etat`.

`orders`: a two-element numeric vector stating the considered model orders.

`cond_dist`: a character value stating the conditional distribution considered in the model fitting.

`long_memo`: a logical value stating whether or not long memory was considered in the model fitting.

`llhood`: the log-likelihood value obtained at the optimal parameter combination.

`inf_criteria`: a named two-element numeric vector with the corresponding AIC (first element) and BIC (second element) of the fitted parametric model part; for purely parametric models, these criteria are valid for the entire model; for semiparametric models, they are only valid for the parametric step and are not valid for the entire model.

`meanspec`: the settings for the model in the conditional mean; is an object of class `"mean_spec"` that is identical to the object passed to the input argument `meanspec`.

`test_obs`: the observations at the end up the input `rt` reserved for testing following `n_test`.

`scale_fun`: the estimated scale function values, if `use_nonpar = TRUE`, otherwise `NULL`; formatting of `rt` is reused.

`nonpar_model`: the estimation object returned by `tsmooth` for `use_nonpar = TRUE`.

`trunc`: the input argument `trunc`.

References

- Ding, Z., Granger, C. W. J., & Engle, R. F. (1993). A long memory property of stock market returns and a new model. Journal of Empirical Finance, 1(1): 83-106. DOI: 10.1016/0927-5398(93)90006-D.
- Feng, Y., Gries, T., Letmathe, S., & Schulz, D. (2022). The smoots Package in R for Semiparametric Modeling of Trend Stationary Time Series. The R Journal, 14(1), 182-195. URL: https://journal.r-project.org/articles/RJ-2022-017/.
- Letmathe, S., Beran, J., & Feng, Y. (2023). An extended exponential SEMIFAR model with application in R. Communications in Statistics - Theory and Methods, 53(22), 7914–7926. DOI: 10.1080/03610926.2023.2276049.
- Tse, Y. K. (1998). The conditional heteroskedasticity of the yen-dollar exchange rate. Journal of Applied Econometrics, 13(1): 49-55. DOI: 10.1002/(SICI)1099-1255(199801/02)13:1<49::AID-JAE459>3.0.CO;2-O.
- Zakoian, J.-M. (1994). Threshold heteroskedastic models. Journal of Economic Dynamics and Control, 18(5): 931-955. DOI: 10.1016/0165-1889(94)90039-6.

Examples

```
window.zoo <- get("window.zoo", envir = asNamespace("zoo"))
rt <- window.zoo(SP500, end = "2002-12-31")
model <- tgarch(rt)
model
```

`tgarch_sim` *Simulate From TGARCH Models*

Description

A streamlined simulation function to simulate from TGARCH models.

Usage

```
tgarch_sim(
 pars = list(mu = 0, ar = numeric(0), ma = numeric(0), D = 0, omega = 4e-04, phi = 0.05,
    beta = 0.8, gamma = 0.1, df = 10, shape = 2, P = 3, skew = 1),
  cond_dist = c("norm", "std", "ged", "ald", "snorm", "sstd", "sged", "sald"),
  n = 1000,
  nstart = 5000,
  trunc = "none"
)
```

Arguments

`pars`	a named list with the parameter specifications; the user can provide a named list with only the settings they would like to adjust relative to the default settings.
`cond_dist`	a one-element character vector specifying the conditional distribution to consider.
`n`	the number of observations to return.
`nstart`	the number of burn-in observations to simulate before the final `n` values to keep; the first `nstart` values are not returned; if a dual model, i.e. with model in the conditional mean and in the conditional variance, is considered, two times `nstart` is considered in the first simulation step in the conditional variance, so that `n + nstart` values can be fed into the second simulation step for the conditional mean.
`trunc`	a truncation for the finite-order coefficient series in long-memory models; can either be the character `"none"` for truncation back to the very first observation at each time point, or to any positive integer for setting the corresponding truncation length of the infinite-order representation polynomial.

Details

See the documentation on `tgarch` for information on the TGARCH model. This function provides an easy way to simulate from these models.

Value

A list with four elements is returned: `rt` are the simulated observations, `etat` are the underlying innovations, `sigt` are the correspondingly simulated conditional standard deviations, and `cmeans` are the simulated conditional means. These four elements are formatted as `"ts"` class time series objects.

Examples

```
sim <- tgarch_sim(n = 1000)
mat <- do.call(cbind, sim)
plot(mat, main = "")
```

---

trafflight_test *Generics for backtests*

Description

Generic functions to build backtesting methods from.

Usage

```
trafflight_test(object, ...)

uncond_cov_test(object, ...)

indep_test(object, ...)

cond_cov_test(object, ...)

cov_tests(object, ...)

backtest_suite(object, ...)

WAD(object, ...)
```

Arguments

`object`	the generics are currently without use.
`...`	the generics are currently without use.

Details

The generics are currently without use.

Value

The generics are currently without use, can therefore not be called and thus don't produce results.

trafflight_test,fEGarch_risk-method
Backtesting VaR and ES

Description

Run traffic light tests for value at risk (VaR) and expected shortfall (ES) as well as a selection of coverage and independence tests for VaR.

Usage

```
## S4 method for signature 'fEGarch_risk'
trafflight_test(object, silent = FALSE, ...)

## S4 method for signature 'fEGarch_risk'
uncond_cov_test(object, silent = FALSE, ...)

## S4 method for signature 'fEGarch_risk'
indep_test(object, silent = FALSE, ...)

## S4 method for signature 'fEGarch_risk'
cond_cov_test(object, silent = FALSE, ...)

## S4 method for signature 'fEGarch_risk'
cov_tests(object, silent = FALSE, ...)

## S4 method for signature 'fEGarch_risk'
backtest_suite(object, silent = FALSE, ...)

## S4 method for signature 'fEGarch_risk'
WAD(object, silent = FALSE, ...)
```

Arguments

object	an object of class "fEGarch_risk".
silent	a logical value indicating whether or not to print test results in a nicely formatted manner to the console.
...	currently without use.

Details

backtest_suite runs all the other backtesting methods. cov_tests runs all of uncond_cov_test, indep_test and cond_cov_test.

Traffic light tests (trafflight_test):

Given an input object object of class "fEGarch_risk", traffic light tests for value at risk (VaR) and expected shortfall (ES) are applied to the individual risk measure series in the object. Note that

in order for a traffic light test in context of ES being applicable, the corresponding VaR series of the same confidence level must also be present in `object`. If this is not fulfilled, messages will be printed to the console, making the user aware of this issue.

Let the number of test observations be denoted by $n \in \mathbb{N}$ and let $\{r_t\}$, $t = 1, \ldots, n$, be the test returns. $\{\text{VaR}_t\}$ are the (one-step rolling) VaR point forecasts for the same period with confidence level α. Denote by I_t an indicator that equals 1, whenever $r_t < \text{VaR}_t$, and 0 otherwise, and define $K_1 = \sum_{t=1}^{n} I_t$. I_t are assumed to follow a binomial distribution with probability $P = \alpha$ for any $I_t = 0$. Then C is computed as the cumulative probability of observing K_1 under P. The forecasted VaR series is then classified following C. If $C < 0.95$, then it is sorted into the green zone, if $0.95 \leq C < 0.9999$, then the series belongs to the yellow zone, and if $C \geq 0.9999$, then the class of the VaR series is the red zone (Basel Committee on Banking Supervision, 1996).

The traffic light test for the ES (Costanzino and Curran, 2018) uses a similar classification system based on the severity of breaches

$$B = \sum_{t=1}^{n} \frac{1 - F(\hat{\eta}_t) - \alpha}{1 - \alpha} I_t,$$

where F is the (fitted) cumulative distribution function of the standardized innovations and with $\hat{\eta}_t$ as the standardized residuals of a fitted GARCH-type model (or of its semiparametric extension). Then $B \overset{a}{\sim} N(\mu_{\text{ES}}, \sigma^2_{\text{ES}})$ with $\mu_{\text{ES}} = 0.5(1-\alpha)n$ and $\sigma^2_{\text{ES}} = (1-\alpha)[(1+3\alpha)/12]n$. The cumulative probability of observing a severity of breaches of B or less can be computed and classified in the same way as for the VaR traffic light test using this asymptotic distribution.

Weighted Absolute Deviation (WAD) (`WAD`): Following the standard computation of the 99%-VaR, the 97.5%-VaR and the 97.5%-ES for the traffic light tests, the WAD criterion takes all of these into account and summarizes them into one numeric value. Let N_1 be the observed breaches for the 99%-VaR for the test set and let μ_1 be the corresponding expected number of breaches. N_2 and μ_2 are to understood analogously for the 97.5%-VaR. N_3 is then the severity of breaches of the 97.5%-ES (i.e. it is equal to B from before) and μ_3 is μ_{ES} from before. Then

$$\text{WAD} = \frac{|N_1 - \mu_1|}{\mu_1} + \frac{|N_2 - \mu_2|}{\mu_2} + \frac{|N_3 - \mu_3|}{\mu_3}.$$

See also Letmathe et al. (2022) for further information.

Coverage and independence tests (`cov_tests`):

Following Christoffersen (1998), the backtesting suite also includes a selection of coverage and independence tests regarding the VaR. Let the number of test observations be denoted by $n \in \mathbb{N}$ and let $\{r_t\}$, $t = 1, \ldots, n$, be the test returns. $\{\text{VaR}_t\}$ are the (one-step rolling) VaR point forecasts for the same period with confidence level α. Furthermore, define I_t to be an indicator that equals 1, whenever $r_t < \text{VaR}_t$ and zero otherwise. Let $K_1 = \sum_{t=1}^{n} I_t$ and $K_0 = n - K_1$. Furthermore, $\hat{z}_1 = K_1/(K_0 + K_1)$ and $\hat{z}_0 = K_0/(K_0 + K_1)$ as well as

$$L_{\hat{z}} = \hat{z}_0^{K_0} \hat{z}_1^{K_1}$$

and

$$L_\alpha = \alpha^{K_0}(1 - \alpha)^{K_1}.$$

In addition, we require $I^*_{i,j}(t)$, $t = 2, \ldots, n$ and $i, j \in \{0, 1\}$, to be other indicators that equal 1, whenever $I_t = j$ and simultaneously $I_{t-1} = i$. Per consequence, $K_{i,j} = \sum_{t=2}^{n} I^*_{i,j}(t)$ and

$\hat{z}_{i,j} = K_{i,j}/(K_{i,0} + K_{i,1})$. Moreover, $\hat{z}_1^* = (K_{0,1} + K_{1,1})/(n-1)$ and $\hat{z}_0^* = 1 - \hat{z}_1^*$. Now,

$$L_{\hat{z}_{0,0}} = \hat{z}_{0,0}^{K_{0,0}} \hat{z}_{0,1}^{K_{0,1}} \hat{z}_{1,0}^{K_{1,0}} \hat{z}_{1,1}^{K_{1,1}}$$

and

$$L_{\hat{z}^*} = (\hat{z}_0^*)^{(K_{0,0}+K_{1,0})} (\hat{z}_1^*)^{(K_{0,1}+K_{1,1})}.$$

Ultimately,

$$L_{\alpha^*} = \alpha^{(K_{0,0}+K_{1,0})} (1-\alpha)^{(K_{0,1}+K_{1,1})}.$$

The three test statistics following Christoffersen (1998) are then

$$S_{\text{uc}} = -2 \ln \left[L_\alpha / L_{\hat{z}} \right] \overset{a}{\sim} \chi^2(1),$$

$$S_{\text{ind}} = -2 \ln \left[L_{\hat{z}^*} / L_{\hat{z}_{0,0}} \right] \overset{a}{\sim} \chi^2(1), \quad \text{and}$$

$$S_{\text{cc}} = -2 \ln \left[L_{\alpha^*} / L_{\hat{z}_{0,0}} \right] \overset{a}{\sim} \chi^2(2),$$

where S_{uc} is the test statistic of the unconditional coverage test, S_{ind} is that of the independence test and S_{cc} is that of the conditional coverage test.

Value

All methods return a list invisibly. The elements of the list differ slightly depending on the method. Moreover, for `silent = FALSE`, the default, test results are printed to the console.

References

- Basel Committee on Banking Supervision (1996). Supervisory Framework For The Use of "Backtesting" in Conjunction With The Internal Models Approach to Market Risk Capital Requirements. URL: https://www.bis.org/publ/bcbs22.pdf.
- Christoffersen, P. F. (1998). Evaluating Interval Forecasts. International Economic Review, 39(4): 841-862. DOI: 10.2307/2527341.
- Costanzino, N., & Curran, M. (2018). A Simple Traffic Light Approach to Backtesting Expected Shortfall. Risks, 6(1). DOI: 10.3390/risks6010002.
- Letmathe, S., Feng, Y., & Uhde, A. (2022). Semiparametric GARCH models with long memory applied to Value at Risk and Expected Shortfall. Journal of Risk, 25(2). DOI: 10.21314/JOR.2022.044.

Examples

```
window.zoo <- get("window.zoo", envir = asNamespace("zoo"))
rt <- window.zoo(SP500, end = "2002-12-31")
model <- fEGarch(egarch_spec(), rt, n_test = 250)
# parallel = FALSE for better compatibility with CRAN checks;
# should be set to TRUE for speed boost
fcast <- predict_roll(model, refit_after = 50, parallel = FALSE)
risk <- measure_risk(fcast, measure = c("VaR", "ES"), level = c(0.95, 0.975, 0.99))
trafflight_test(risk)
cov_tests(risk)
backtest_suite(risk)
```

UKinflation *Monthly Inflation Rate of the UK*

Description

A "ts" object that contains the monthly inflation rate of the UK from January 1956 to December 2000. It is the growth rate of the consumer price index from the previous period.

Usage

```
UKinflation
```

Format

An object of class ts of length 540.

Source

The data was obtained from the databank of the Federal Reserve Bank of St. Louis (accessed: 2025-06-17) and then transformed into a time series "ts" object using R.

https://fred.stlouisfed.org/series/CPALTT01GBM657N

VaR_calc *VaR and ES Computation for Standardized Distributions*

Description

Compute the value at risk (VaR) and the expected shortfall (ES) numerically for the standardized distributions available in this package. These quantiles can then be used to obtain the conditional VaR and ES following GARCH-type models.

Usage

```
VaR_calc(
  level = 0.99,
  dist = c("norm", "std", "ged", "ald", "snorm", "sstd", "sged", "sald"),
  ...
)

ES_calc(
  level = 0.975,
  dist = c("norm", "std", "ged", "ald", "snorm", "sstd", "sged", "sald"),
  ...
)
```

Arguments

level	a numeric vector with the confidence level(s) to calculate the VaR or ES for; the quantiles are the VaR or ES computed at one minus the input level; the result is thus with negative sign for common level input such as 0.975 or 0.99.
dist	a single character value (or the default vector) that specifies the distribution to consider (all distributions are considered to be standardized with mean zero and variance one).
...	further arguments to consider for the distributions; for "std" or "sstd", specify the degrees of freedom df, for "ged" or "sged", give the shape parameter shape, and for "ald" or "sald", use the additional argument P; moreover, for the skewed distributions "snorm", "sstd", "sged" and "sald", the skewness parameter skew must be provided as well.

Details

The VaR is found numerically using numerical root finding via uniroot (of the stats package), whereas the ES is obtained through numerical integration, where firstly the VaR at the corresponding confidence level is computed using VaR_calc, and where subsequently integrate (of the stats package) is used at the tail of the distribution.

In detail, let $f(x)$ be the probability density function (pdf) of a standardized random variable with mean zero and variance one. Without the need to state a cumulative distribution function (cdf) mathematically, we can define it in R numerically by integrating over f from -Inf to some quantile x using integrate. To then find a quantile for a given cumulative probability, we can use uniroot to find the quantile, where the numerical cdf function minus the set cumulative probability equals zero. This way, a numerical VaR can be found.

On the other hand, a numerical ES for a (continuous) random variable with mean zero and variance one follows the alternative definition of the ES

$$\mathrm{ES}_\alpha = (1-\alpha)^{-1} \int_{-\infty}^{\mathrm{VaR}_\alpha} x f(x) dx,$$

where α, usually 0.99 or 0.975, is the confidence level, VaR_α is the VaR at the same α. Using the previous approach, VaR_α can be easily identified. Then, in R, a function $g(x) = xf(x)$ can also be defined easily. Ultimately, only integrate needs to be applied from -Inf to the corresponding VaR as the upper bound to the function g. The resulting numerical integral is then divided by $(1-\alpha)$.

Value

Returns a numeric vector of the same length as the input argument level.

Examples

```
# 99-percent VaR, normal distribution
VaR_calc(level = 0.99, dist = "norm")

# 99-percent VaR, t-distribution with df = 10
VaR_calc(level = 0.99, dist = "std", df = 10)

# 97.5-percent ES, normal distribution
```

```
ES_calc(level = 0.975, dist = "norm")

# 97.5-percent ES, t-distribution with df = 10
ES_calc(level = 0.975, dist = "std", df = 10)
```

Index

∗ **datasets**
SP500, 109
UKinflation, 124

accessor_methods, 7
ald_est *(*distr_est*)*, 18
aparch, 9, *15*, *73*
aparch_ru *(*rugarch_wrappers*)*, 102
aparch_sim, *5*, 14
autoplot,fEGarch_fit-method, 15
autoplot,fEGarch_risk-method, 16

backtest_suite *(*trafflight_test*)*, 120
backtest_suite,fEGarch_risk-method, *5*
backtest_suite,fEGarch_risk-method *(*trafflight_test,fEGarch_risk-method*)*, 121
boundary_method *(*poly_order*)*, 97
boundary_method,locpol_spec-method *(*locpol_spec_methods*)*, 87
boundary_method<- *(*poly_order*)*, 97
boundary_method<-,locpol_spec-method *(*locpol_spec_methods*)*, 87
bwidth *(*poly_order*)*, 97
bwidth,locpol_spec-method *(*locpol_spec_methods*)*, 87
bwidth<- *(*poly_order*)*, 97
bwidth<-,locpol_spec-method *(*locpol_spec_methods*)*, 87

close_to_lreturn, 17
cmeans *(*fitted_object_generics*)*, 64
cmeans,ANY-method *(*fitted_object_generics*)*, 64
cmeans,fEGarch_fit-method *(*accessor_methods*)*, 7
cmeans,fEGarch_forecast-method *(*accessor_methods*)*, 7
cond_cov_test *(*trafflight_test*)*, 120
cond_cov_test,fEGarch_risk-method *(*trafflight_test,fEGarch_risk-method*)*, 121
cond_dist *(*spec_generics*)*, 109
cond_dist,ANY-method *(*spec_generics*)*, 109
cond_dist,base_garch_spec-method *(*spec_methods*)*, 110
cond_dist<- *(*spec_generics*)*, 109
cond_dist<-,base_garch_spec-method *(*spec_methods*)*, 110
cov_tests *(*trafflight_test*)*, 120
cov_tests,fEGarch_risk-method *(*trafflight_test,fEGarch_risk-method*)*, 121

distr_est, 18, *54*, *81*, *82*

egarch_ru *(*rugarch_wrappers*)*, 102
egarch_spec, *4*
egarch_spec *(*submodel-specs*)*, 112
egarch_type_spec, 21
ES_calc *(*VaR_calc*)*, 124
etat *(*fitted_object_generics*)*, 64
etat,ANY-method *(*fitted_object_generics*)*, 64
etat,fEGarch_fit-method *(*accessor_methods*)*, 7

fEGarch, *4*, *5*, *7*, *8*, *16*, 22, *55*, *64*, *81*, *84*, *86*, *89*, *92*, *96*, *99*, *106*, *107*
fEGarch-package, 4
fEGarch_sim, *5*, 28
fEGarch_spec, *4*, *5*, *22–25*, *28*, 29, *90*
fiaparch, *4*, 33, *39*, *64*, *73*, *90*, *99*, *106*
fiaparch_sim, *5*, 39
fiegarch_spec, *4*
fiegarch_spec *(*submodel-specs*)*, 112
figarch, *4*, 40, *46*, *64*, *73*, *90*, *99*, *106*
figarch_sim, *5*, 45

figjrgarch, *4*, 47, *53*, *73*
figjrgarch_sim, *5*, 52
filoggarch_spec, *4*
filoggarch_spec *(*submodel-specs*)*, 112
fimegarch_spec, *4*
fimegarch_spec *(*submodel-specs*)*, 112
fimloggarch_spec, *4*
fimloggarch_spec *(*submodel-specs*)*, 112
find_dist, *5*, 54, *93, 94*
fit_test_suite *(*ljung_box_test*)*, 82
fit_test_suite,fEGarch_fit-method, 55
fitgarch, *4*, 56, *62*, *73*
fitgarch_sim, *5*, 62
fitted,fEGarch_fit-method, 63
fitted,fEGarch_forecast-method *(*fitted,fEGarch_fit-method*)*, 63
fitted_object_generics, 64

garch, 65, *70*, *73*
garch_sim, *5*, 69
garchm_estim, *4, 5*, *7, 8*, 71
ged_est *(*distr_est*)*, 18
gjrgarch, *73*, 74, *80*
gjrgarch_ru *(*rugarch_wrappers*)*, 102
gjrgarch_sim, *5*, 79
goodn_of_fit_test *(*ljung_box_test*)*, 82
goodn_of_fit_test,fEGarch_fit-method, 80

hessian, *12*, *24*, *36*, *43*, *50*, *59*, *67*, *77*, *116*

include_mean *(*spec_generics*)*, 109
include_mean,ANY-method *(*spec_generics*)*, 109
include_mean,mean_spec-method *(*mean_spec_methods*)*, 91
include_mean<- *(*spec_generics*)*, 109
include_mean<-,mean_spec-method *(*mean_spec_methods*)*, 91
indep_test *(*trafflight_test*)*, 120
indep_test,fEGarch_risk-method *(*trafflight_test,fEGarch_risk-method*)*, 121
inf_criteria *(*fitted_object_generics*)*, 64
inf_criteria,ANY-method *(*fitted_object_generics*)*, 64
inf_criteria,fEGarch_distr_est-method, 81
inf_criteria,fEGarch_fit-method *(*accessor_methods*)*, 7
integrate, *125*

kernel_order *(*poly_order*)*, 97
kernel_order,locpol_spec-method *(*locpol_spec_methods*)*, 87
kernel_order<- *(*poly_order*)*, 97
kernel_order<-,locpol_spec-method *(*locpol_spec_methods*)*, 87

ljung_box_test, 82
ljung_box_test,fEGarch_fit-method, 84
llhood *(*fitted_object_generics*)*, 64
llhood,ANY-method *(*fitted_object_generics*)*, 64
llhood,fEGarch_distr_est-method *(*inf_criteria,fEGarch_distr_est-method*)*, 81
llhood,fEGarch_fit-method *(*accessor_methods*)*, 7
locpol_spec, *10*, *12*, *23*, *25*, *34*, *37*, *41*, *43*, *48*, *50*, *57*, *59*, *66*, *68*, *72*, *75*, *77*, 85, *103*, *115*, *117*
locpol_spec_methods, 87
loggarch_spec, *4*
loggarch_spec *(*submodel-specs*)*, 112
loggarch_type_spec *(*egarch_type_spec*)*, 21
long_memo *(*spec_generics*)*, 109
long_memo,ANY-method *(*spec_generics*)*, 109
long_memo,base_garch_spec-method *(*spec_methods*)*, 110
long_memo,mean_spec-method *(*mean_spec_methods*)*, 91
long_memo<- *(*spec_generics*)*, 109
long_memo<-,base_garch_spec-method *(*spec_methods*)*, 110
long_memo<-,mean_spec-method *(*mean_spec_methods*)*, 91
loss_functions, 88
loss_functions,fEGarch_risk-method, 88

mean_spec, *5*, *24*, *32*, 90
mean_spec_methods, 91
measure_risk, *5*, 92
measure_risk,fEGarch_distr_est-method *(*measure_risk*)*, 92

measure_risk,fEGarch_fit-method
(measure_risk), 92
measure_risk,fEGarch_forecast-method
(measure_risk), 92
megarch_spec, *4*, *23*
megarch_spec (submodel-specs), 112
mloggarch_spec, *4*, *23*
mloggarch_spec (submodel-specs), 112
modulus (spec_generics), 109
modulus,ANY-method (spec_generics), 109
modulus,egarch_type_spec-method
(spec_methods), 110
modulus<- (spec_generics), 109
modulus<-,egarch_type_spec-method
(spec_methods), 110

norm_est, *81*, *82*
norm_est (distr_est), 18

orders (spec_generics), 109
orders,ANY-method (spec_generics), 109
orders,base_garch_spec-method
(spec_methods), 110
orders,mean_spec-method
(mean_spec_methods), 91
orders<- (spec_generics), 109
orders<-,base_garch_spec-method
(spec_methods), 110
orders<-,mean_spec-method
(mean_spec_methods), 91

pars (fitted_object_generics), 64
pars,ANY-method
(fitted_object_generics), 64
pars,fEGarch_distr_est-method
(inf_criteria,fEGarch_distr_est-method),
81
pars,fEGarch_fit-method
(accessor_methods), 7
plot, 95, *96*
plot,fEGarch_fit,ANY-method, 95
plot,fEGarch_fit-method
(plot,fEGarch_fit,ANY-method),
95
plot,fEGarch_risk,ANY-method, 96
plot,fEGarch_risk-method
(plot,fEGarch_risk,ANY-method),
96
poly_order, 97
poly_order,locpol_spec-method
(locpol_spec_methods), 87
poly_order<- (poly_order), 97
poly_order<-,locpol_spec-method
(locpol_spec_methods), 87
powers (spec_generics), 109
powers,ANY-method (spec_generics), 109
powers,egarch_type_spec-method
(spec_methods), 110
powers<- (spec_generics), 109
powers<-,egarch_type_spec-method
(spec_methods), 110
predict,fEGarch_fit-method, *5*, 98
predict_roll, 100
predict_roll,fEGarch_fit-method, *5*
predict_roll,fEGarch_fit-method
(predict,fEGarch_fit-method),
98

rald_s (sim_functions), 108
residuals,fEGarch_fit-method, 101
rged_s (sim_functions), 108
rnorm_s (sim_functions), 108
rsald_s (sim_functions), 108
rsged_s (sim_functions), 108
rsnorm_s (sim_functions), 108
rsstd_s (sim_functions), 108
rstd_s (sim_functions), 108
rugarch_wrappers, 102

sald_est (distr_est), 18
se (fitted_object_generics), 64
se,ANY-method (fitted_object_generics),
64
se,fEGarch_distr_est-method
(inf_criteria,fEGarch_distr_est-method),
81
se,fEGarch_fit-method
(accessor_methods), 7
sged_est (distr_est), 18
show,fEGarch_distr_est-method
(show,fEGarch_fit_egarch-method),
105
show,fEgarch_distr_est-method
(show,fEGarch_fit_egarch-method),
105
show,fEGarch_fit_aparch-method
(show,fEGarch_fit_egarch-method),
105

show,fEgarch_fit_aparch-method
(show,fEGarch_fit_egarch-method), 105
show,fEGarch_fit_egarch-method, 105
show,fEGarch_fit_fiaparch-method
(show,fEGarch_fit_egarch-method), 105
show,fEgarch_fit_fiaparch-method
(show,fEGarch_fit_egarch-method), 105
show,fEGarch_fit_figarch-method
(show,fEGarch_fit_egarch-method), 105
show,fEgarch_fit_figarch-method
(show,fEGarch_fit_egarch-method), 105
show,fEGarch_fit_figjrgarch-method
(show,fEGarch_fit_egarch-method), 105
show,fEgarch_fit_figjrgarch-method
(show,fEGarch_fit_egarch-method), 105
show,fEGarch_fit_fitgarch-method
(show,fEGarch_fit_egarch-method), 105
show,fEgarch_fit_fitgarch-method
(show,fEGarch_fit_egarch-method), 105
show,fEGarch_fit_garch-method
(show,fEGarch_fit_egarch-method), 105
show,fEgarch_fit_garch-method
(show,fEGarch_fit_egarch-method), 105
show,fEGarch_fit_gjrgarch-method
(show,fEGarch_fit_egarch-method), 105
show,fEgarch_fit_gjrgarch-method
(show,fEGarch_fit_egarch-method), 105
show,fEGarch_fit_loggarch-method
(show,fEGarch_fit_egarch-method), 105
show,fEgarch_fit_loggarch-method
(show,fEGarch_fit_egarch-method), 105
show,fEGarch_fit_rugarch_wrapper-method
(show,fEGarch_fit_egarch-method), 105
show,fEgarch_fit_rugarch_wrapper-method
(show,fEGarch_fit_egarch-method), 105
show,fEGarch_fit_tgarch-method
(show,fEGarch_fit_egarch-method), 105
show,fEgarch_fit_tgarch-method
(show,fEGarch_fit_egarch-method), 105
sigma,fEGarch_fit-method, 106
sigma,fEGarch_forecast-method
(sigma,fEGarch_fit-method), 106
sign_bias_test (ljung_box_test), 82
sign_bias_test,fEGarch_fit-method, 107
sigt (fitted_object_generics), 64
sigt,ANY-method
(fitted_object_generics), 64
sigt,fEGarch_fit-method
(accessor_methods), 7
sigt,fEGarch_forecast-method
(accessor_methods), 7
sim_functions, 108
snorm_est (distr_est), 18
SP500, *5*, 109
spec_generics, 109
spec_methods, 110
sstd_est (distr_est), 18
std_est (distr_est), 18
submodel-specs, 112

tgarch, *73*, 114, *119*
tgarch_sim, *5*, 119
trafflight_test, 120
trafflight_test,fEGarch_risk-method,
121
tsmooth, *12, 13*, *25, 26*, *68, 69*, *74*, *77, 78*,
103, *117, 118*
tsmoothlm, *25, 26*, *37, 38*, *43, 44*, *50, 51*,
59–61, *74*, *77*, *104*, *117*

UKinflation, *5*, 124
uncond_cov_test (trafflight_test), 120
uncond_cov_test,fEGarch_risk-method
(trafflight_test,fEGarch_risk-method),
121
uniroot, *125*

VaR_calc, 124

vcov_mat (fitted_object_generics), 64
vcov_mat,ANY-method
(fitted_object_generics), 64
vcov_mat,fEGarch_distr_est-method
(inf_criteria,fEGarch_distr_est-method),
81
vcov_mat,fEGarch_fit-method
(accessor_methods), 7

WAD (trafflight_test), 120
WAD,fEGarch_risk-method
(trafflight_test,fEGarch_risk-method),
121