Package: facmodTS (via r-universe)

September 14, 2024

Type Package

Title Time Series Factor Models for Asset Returns

Version 1.0

Date 2023-10-27

Description Supports teaching methods of estimating and testing time series factor models for use in robust portfolio construction and analysis. Unique in providing not only classical least squares, but also modern robust model fitting methods which are not much influenced by outliers. Includes returns and risk decompositions, with user choice of standard deviation, value-at-risk, and expected shortfall risk measures. ``Robust Statistics Theory and Methods (with R)'', R. A. Maronna, R. D. Martin, V. J. Yohai, M. Salibian-Barrera (2019) <doi:10.1002/9781119214656>.

License GPL-2

Depends R (>= 3.5)

Imports boot, data.table, lars, lattice, leaps, PerformanceAnalytics, PortfolioAnalytics, R.cache, corpcor, methods, quadprog, RobStatTM, robustbase, sandwich, sn, xts, zoo

Suggests corrplot, HH, lmtest, R.rsp, rugarch, strucchange, tinytest

URL https://github.com/robustport/facmodTS

RoxygenNote 7.2.3

Encoding UTF-8

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

RemoteUrl https://github.com/robustport/facmodts

RemoteRef HEAD

RemoteSha a1b8d1394ee759b8037a109e330a232b38a9787a

Contents

`fitTsfm` *Fit a time series factor model using time series regression*

Description

Fits a time series (a.k.a. macroeconomic) factor model for one or more asset returns or excess returns using time series regression. Users can choose between ordinary least squares-LS, discounted least squares-DLS (or) robust regression. Several variable selection options including Stepwise, Subsets, Lars are available as well. An object of class `"tsfm"` is returned.

Usage

```
fitTsfm(
  asset.names,
  factor.names,
  mkt.name = NULL,
  rf.name = NULL,
  data = data,
  fit.method = c("LS", "DLS", "Robust"),
  variable.selection = c("none", "stepwise", "subsets", "lars"),
  control = fitTsfm.control(),
  ...
```

```
)

## S3 method for class 'tsfm'
coef(object, ...)

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

## S3 method for class 'tsfm'
residuals(object, ...)
```

Arguments

asset.names	vector of syntactically valid asset names, whose returns are the dependent variable in the factor model.
factor.names	vector containing syntactically valid names of the factors.
mkt.name	syntactically valid name of the column for market returns. Default is NULL.
rf.name	syntactically valid name of the column for the risk free rate; if excess returns should be calculated for all assets and factors. Default is NULL.
data	vector, matrix, data.frame, xts, timeSeries or zoo object containing the columns asset.names, factor.names, and optionally, mkt.name and rf.name.
fit.method	the estimation method, one of "LS", "DLS" or "Robust". See details. Default is "LS".
variable.selection	the variable selection method, one of "none", "stepwise","subsets","lars". See details. Default is "none".
control	list of control parameters. Refer to fitTsfm.control for details.
...	arguments passed to fitTsfm.control
object	a fit object of class tsfm which is returned by fitTsfm

Details

Typically, factor models are fit using excess returns. rf.name gives the option to supply a risk free rate variable to subtract from each asset return and factor to compute excess returns.

Estimation method "LS" corresponds to ordinary least squares using lm, "DLS" is discounted least squares (weighted least squares with exponentially declining weights that sum to unity), and, "Robust" is robust regression (using lmrobdetMM).

If variable.selection="none", uses all the factors and performs no variable selection. Whereas, "stepwise" performs traditional stepwise LS or Robust regression (using step or step.lmrobdetMM), that starts from the initial set of factors and adds/subtracts factors only if the regression fit, as measured by the Bayesian Information Criterion (BIC) or Akaike Information Criterion (AIC), improves. And, "subsets" enables subsets selection using regsubsets; chooses the best performing subset of any given size or within a range of subset sizes. Different methods such as exhaustive search (default), forward or backward stepwise, or sequential replacement can be employed. See fitTsfm.control for more details on the control arguments.

variable.selection="lars" corresponds to least angle regression using lars with variants "lasso" (default), "lar", "stepwise" or "forward.stagewise". Note: If variable.selection="lars", fit.method will be ignored.

Argument mkt.name can be used to add market-timing factors to any of the above methods. Please refer to fitTsfmMT, a wrapper to fitTsfm for details.

Data Processing:

Note about NAs: Before model fitting, incomplete cases are removed for every asset (return data combined with respective factors' return data) using na.omit. Otherwise, all observations in data are included.

Note about asset.names and factor.names: Spaces in column names of data will be converted to periods as fitTsfm works with xts objects internally and colnames won't be left as they are.

Value

fitTsfm returns an object of class "tsfm" for which print, plot, predict and summary methods exist.

The generic functions coef, fitted and residuals extract various useful features of the fit object. Additionally, fmCov computes the covariance matrix for asset returns based on the fitted factor model.

An object of class "tsfm" is a list containing the following components:

asset.fit	list of fitted objects for each asset. Each object is of class lm if fit.method="LS" or "DLS", class lmrobdetMM if the fit.method="Robust", or class lars if variable.selection="lars".
alpha	N x 1 data.frame of estimated alphas.
beta	N x K data.frame of estimated betas.
r2	length-N vector of R-squared values.
resid.sd	length-N vector of residual standard deviations.
fitted	xts data object of fitted values; iff variable.selection="lars"
call	the matched function call.
data	xts data object containing the asset(s) and factor(s) returns.
asset.names	syntactically valid asset.names as input.
factor.names	syntactically valid factor.names as input.
mkt.name	syntactically valid mkt.name as input
fit.method	fit.method as input.
variable.selection	variable.selection as input.

Where N is the number of assets, K is the number of factors and T is the number of time periods.

Author(s)

Eric Zivot, Sangeetha Srinivasan and Yi-An Chen.

References

Christopherson, J. A., Carino, D. R., & Ferson, W. E. (2009). Portfolio performance measurement and benchmarking. McGraw Hill Professional.

Efron, B., Hastie, T., Johnstone, I., & Tibshirani, R. (2004). Least angle regression. The Annals of statistics, 32(2), 407-499.

Hastie, T., Tibshirani, R., Friedman, J., Hastie, T., Friedman, J., & Tibshirani, R. (2009). The elements of statistical learning (Vol. 2, No. 1). New York: Springer.

See Also

The `tsfm` methods for generic functions: `plot.tsfm`, `predict.tsfm`, `print.tsfm` and `summary.tsfm`.

And, the following extractor functions: `coef`, `fitted`, `residuals`, `fmCov`, `fmSdDecomp`, `fmVaRDecomp` and `fmEsDecomp`.

`paFm` for Performance Attribution.

Examples

```
 # load data
data(managers, package = 'PerformanceAnalytics')

fit <- fitTsfm(asset.names = colnames(managers[,(1:6)]),
               factor.names = colnames(managers[,(7:9)]),
               data=managers)
summary(fit)
fitted(fit)

# example using "subsets" variable selection
fit.sub <- fitTsfm(asset.names=colnames(managers[,(1:6)]),
                   factor.names=colnames(managers[,(7:9)]),
                   data=managers,
                   variable.selection="subsets",
                   method="exhaustive",
                   nvmin=2)

# example using "lars" variable selection and subtracting risk-free rate
fit.lar <- fitTsfm(asset.names=colnames(managers[,(1:6)]),
                   factor.names=colnames(managers[,(7:9)]),
                   rf.name="US 3m TR",
                   data=managers,
                   variable.selection="lars",
                   lars.criterion="cv")
```

`fitTsfm.control` *List of control parameters for* `fitTsfm`

Description

Creates a list of control parameters for `fitTsfm`. All control parameters that are not passed to this function are set to default values. This function is meant for internal use only!!

Usage

```
fitTsfm.control(
  decay = 0.95,
  weights,
  model = TRUE,
  x = FALSE,
  y = FALSE,
  qr = TRUE,
  nrep = NULL,
  bb = 0.5,
  efficiency = 0.95,
  family = "mopt",
  tuning.psi,
  tuning.chi,
  compute.rd = FALSE,
  corr.b = TRUE,
  split.type = "f",
  initial = "S",
  max.it = 100,
  refine.tol = 1e-07,
  rel.tol = 1e-07,
  refine.PY = 10,
  solve.tol = 1e-07,
  trace.lev = 0,
  psc_keep = 0.5,
  resid_keep_method = "threshold",
  resid_keep_thresh = 2,
  resid_keep_prop = 0.2,
  py_maxit = 20,
  py_eps = 1e-05,
  mscale_maxit = 50,
  mscale_tol = 1e-06,
  mscale_rho_fun = "bisquare",
  scope,
  scale,
  direction,
  steps = 1000,
  k = 2,
  nvmin = 1,
  nvmax = 8,
  force.in = NULL,
  force.out = NULL,
  method,
```

```
  really.big = FALSE,
  type,
  normalize = TRUE,
  eps = .Machine$double.eps,
  max.steps,
  plot.it = FALSE,
  lars.criterion = "Cp",
  K = 10
)
```

Arguments

decay	a scalar in (0, 1] to specify the decay factor for "DLS". Default is 0.95.
weights	an optional vector of weights to be used in the fitting process for `fit.method="LS","Robust"`, or `variable.selection="subsets"`. Should be `NULL` or a numeric vector. The length of `weights` must be the same as the number of observations. The weights must be nonnegative and it is strongly recommended that they be strictly positive.
model, x, y, qr	logicals passed to `lm` for `fit.method="LS"`. If `TRUE` the corresponding components of the fit (the model frame, the model matrix, the response, the QR decomposition) are returned.
nrep	the number of random subsamples to be drawn for `fit.method="Robust"`. If the data set is small and "Exhaustive" resampling is being used, the value of `nrep` is ignored.
bb	tuning constant (between 0 and 1/2) for the M-scale used to compute the initial S-estimator. It determines the robustness (breakdown point) of the resulting MM-estimator, which is bb. Defaults to 0.5.
efficiency	desired asymptotic efficiency of the final regression M-estimator. Defaults to 0.85.
family	string specifying the name of the family of loss function to be used (current valid options are "bisquare", "optimal" and "modopt" from the RobStatTM package). Incomplete entries will be matched to the current valid options.
tuning.psi	tuning parameters for the regression M-estimator computed with a rho function as specified with argument `family`. If missing, it is computed inside `lmrobdet.control` to match the value of `efficiency` according to the family of rho functions specified in `family`. Appropriate values for `tuning.psi` for a given desired efficiency for Gaussian errors can be constructed using the functions `bisquare`, `mopt` and `opt`.
tuning.chi	tuning constant for the function used to compute the M-scale used for the initial S-estimator. If missing, it is computed inside `lmrobdet.control` to match the value of bb according to the family of rho functions specified in `family`.
compute.rd	logical value indicating whether robust leverage distances need to be computed.
corr.b	logical value indicating whether a finite-sample correction should be applied to the M-scale parameter bb.
split.type	determines how categorical and continuous variables are split. See `splitFrame`.

initial	string specifying the initial value for the M-step of the MM-estimator. Valid options are 'S', for an S-estimator and 'MS' for an M-S estimator which is appropriate when there are categorical explanatory variables in the model.
max.it	maximum number of IRWLS iterations for the MM-estimator
refine.tol	relative convergence tolerance for the S-estimator
rel.tol	relative convergence tolerance for the IRWLS iterations for the MM-estimator
refine.PY	number of refinement steps for the Pen~a-Yohai candidates
solve.tol	relative tolerance for inversion
trace.lev	positive values (increasingly) provide details on the progress of the MM-algorithm
psc_keep	For pyinit, proportion of observations to remove based on PSCs. The effective proportion of removed observations is adjusted according to the sample size to be prosac*(1-p/n). See pyinit.
resid_keep_method	For pyinit, how to clean the data based on large residuals. If "threshold", all observations with scaled residuals larger than C.res will be removed, if "proportion", observations with the largest prop residuals will be removed. See pyinit.
resid_keep_thresh	See parameter resid_keep_method above. See pyinit.
resid_keep_prop	See parameter resid_keep_method above. See pyinit.
py_maxit	Maximum number of iterations. See pyinit.
py_eps	Relative tolerance for convergence. See pyinit.
mscale_maxit	Maximum number of iterations for the M-scale algorithm. See pyinit.
mscale_tol	Convergence tolerance for the M-scale algorithm. See pyinit.
mscale_rho_fun	String indicating the loss function used for the M-scale. See pyinit.
scope	defines the range of models examined in the "stepwise" search. This should be either a single formula, or a list containing components upper and lower, both formulae. See step for how to specify the formulae and usage.
scale	optional parameter for variable.selection="stepwise". The argument is passed to step or step.lmrobdetMM as appropriate.
direction	the mode of "stepwise" search, can be one of "both", "backward", or "forward", with a default of "both". If the scope argument is missing the default for direction is "backward".
steps	the maximum number of steps to be considered for "stepwise". Default is 1000 (essentially as many as required). It is typically used to stop the process early.
k	the multiple of the number of degrees of freedom used for the penalty in "stepwise". Only k = 2 gives the genuine AIC. k = log(n) is sometimes referred to as BIC or SBC. Default is 2.
nvmin	minimum size of subsets to examine for "subsets". Default is 1.
nvmax	maximum size of subsets to examine for "subsets". Default is 8.

force.in	index to columns of design matrix that should be in all models for "subsets". Default is NULL.
force.out	index to columns of design matrix that should be in no models for "subsets". Default is NULL.
method	one of "exhaustive", "forward", "backward" or "seqrep" (sequential replacement) to specify the type of subset search/selection. Required if variable selection="subsets" is chosen. Default is "exhaustive".
really.big	option for "subsets"; Must be TRUE to perform exhaustive search on more than 50 variables.
type	option for "lars". One of "lasso", "lar", "forward.stagewise" or "stepwise". The names can be abbreviated to any unique substring. Default is "lasso".
normalize	option for "lars". If TRUE, each variable is standardized to have unit L2 norm, otherwise they are left alone. Default is TRUE.
eps	option for "lars"; An effective zero.
max.steps	Limit the number of steps taken for "lars"; the default is 8 * min(m, n-intercept), with m the number of variables, and n the number of samples. For type="lar" or type="stepwise", the maximum number of steps is min(m,n-intercept). For type="lasso" and especially type="forward.stagewise", there can be many more terms, because although no more than min(m,n-intercept) variables can be active during any step, variables are frequently droppped and added as the algorithm proceeds. Although the default usually guarantees that the algorithm has proceeded to the saturated fit, users should check.
plot.it	option to plot the output for cv.lars. Default is FALSE.
lars.criterion	an option to assess model selection for the "lars" method; one of "Cp" or "cv". See details. Default is "Cp".
K	number of folds for computing the K-fold cross-validated mean squared prediction error for "lars". Default is 10.
trace	If positive (or, not FALSE), info is printed during the running of step, lars or cv.lars as relevant. Larger values may give more detailed information. Default is FALSE.

Details

This control function is used to process optional arguments passed via ... to fitTsfm. These arguments are validated and defaults are set if necessary before being passed internally to one of the following functions: lm, lmrobdetMM, step, regsubsets, lars and cv.lars. See their respective help files for more details. The arguments to each of these functions are listed above in approximately the same order for user convenience.

The scalar decay is used by fitTsfm to compute exponentially decaying weights for fit.method="DLS". Alternately, one can directly specify weights, a weights vector, to be used with "LS" or "Robust". Especially when fitting multiple assets, care should be taken to ensure that the length of the weights vector matches the number of observations (excluding cases ignored due to NAs).

lars.criterion selects the criterion (one of "Cp" or "cv") to determine the best fitted model for variable.selection="lars". The "Cp" statistic (defined in page 17 of Efron et al. (2004)) is calculated using summary.lars. While, "cv" computes the K-fold cross-validated mean squared prediction error using cv.lars.

Value

A list of the above components. This is only meant to be used by `fitTsfm`.

Author(s)

Sangeetha Srinivasan

References

Efron, B., Hastie, T., Johnstone, I., & Tibshirani, R. (2004). Least angle regression. The Annals of statistics, 32(2), 407-499.

See Also

`fitTsfm`, `lm`, `lmrobdetMM`, `step`, `regsubsets`, `lars` and `cv.lars`

Examples

```
# check argument list passed by fitTsfm.control
tsfm.ctrl <- fitTsfm.control(method="exhaustive", nvmin=2)
print(tsfm.ctrl)


# used internally by fitTsfm in the example below
 # load data
data(managers, package = 'PerformanceAnalytics')
 # Make syntactically valid column names
colnames(managers)
colnames(managers) <- make.names( colnames(managers))
colnames(managers)

fit <- fitTsfm(asset.names=colnames(managers[,(1:6)]),
               factor.names=colnames(managers[,(7:9)]),
               data=managers, variable.selection="subsets",
               method="exhaustive", nvmin=2)
```

`fitTsfmLagLeadBeta` *Fit a lagged and lead Betas factor model using time series regression*

Description

This is a wrapper function to fits a time series lagged Betas factor model for one or more asset returns or excess returns using time series regression. Users can choose between ordinary least squares-LS, discounted least squares-DLS (or) robust regression like `fitTsfm`.An object of class `"tsfm"` is returned.

Usage

```
fitTsfmLagLeadBeta(
  asset.names,
  mkt.name,
  rf.name = NULL,
  data = data,
  fit.method = c("LS", "DLS", "Robust"),
  LagLeadBeta = 1,
  LagOnly = FALSE,
  control = fitTsfm.control(),
  ...
)
```

Arguments

asset.names	vector containing names of assets, whose returns or excess returns are the dependent variable.
mkt.name	name of the column for market returns. It is required for a lagged Betas factor model.
rf.name	name of the column of risk free rate variable to calculate excess returns for all assets (in asset.names) and the market factor (in mkt.name).Default is NULL, and no action is taken.
data	vector, matrix, data.frame, xts, timeSeries or zoo object containing column(s) named in asset.names, factor.names and optionally, mkt.name and rf.name.
fit.method	the estimation method, one of "LS", "DLS" or "Robust". See details. Default is "LS".
LagLeadBeta	A integer number to specify numbers of lags (and leads when LagOnly is FALSE) of Betas to include in the model. The Default is 1.
LagOnly	Flag variable to only include the lags (or have both lags and leads). The Default is FALSE (both lags and leads).
control	list of control parameters. The default is constructed by the function fitTsfm.control. See the documentation for fitTsfm.control for details.
...	arguments passed to fitTsfm.control

Details

The lagged and lead returns model estimates lagged and lead market Beta. Specifically,

$$r_t = \alpha + \beta_0 MKT_t + \beta_1^- MKT_t - 1 + \ldots + \beta_K^- + 1 MKT_t - K + \beta_1^+ MKT_t + 1 + \ldots + \beta_K^+ MKT_t + K \epsilon_t, t = 1 \ldots T$$

where r_t is the asset returns, and MKT is the market factor. It is usually needed for illiquid securities with stale prices. One can also report the sum of the lagged and lead Betas:

$$\beta = \beta_0 + \beta_1^+ + \beta_1^+ + \ldots + \beta_K^+ + \beta_1^- + \ldots + \beta_K^-$$

Value

fitTsfmLagLeadBeta also returns an object of class "tsfm" like fitTsfm. The generic function such as print, plot, predict and summary methods exist. Also, the generic accessor functions coef, fitted, residuals and fmCov can be applied as well.

An object of class "tsfm" is a list containing the following components:

asset.fit	list of fitted objects for each asset. Each object is of class lm if fit.method="LS" or "DLS", class lmRob if the fit.method="Robust".
alpha	length-N vector of estimated alphas.
beta	N x (L+1) matrix of estimated betas.
r2	length-N vector of R-squared values.
resid.sd	length-N vector of residual standard deviations.
call	the matched function call.
data	xts data object containing the assets and factors.
asset.names	asset.names as input.
fit.method	fit.method as input.

Where N is the number of assets, L is the number of lagged and lead market Betas and T is the number of time periods.

Author(s)

Yi-An Chen.

References

Scholes, M. and Williams, J. T. (1977). Estimating betas from non-synchronous data, Journal of Financial Economics, vol. 5, 1977, pp. 309-327

See Also

The original time series function fitTsfm and its generic functions application.

Examples

```
## A lagged Betas model with LS fit

 # load data
data(managers, package = 'PerformanceAnalytics')

fit <- fitTsfmLagLeadBeta(asset.names = names(managers[,(1:6)]),
                          mkt.name = "SP500 TR", rf.name = "US 3m TR",
                          data = managers, LagLeadBeta = 2, LagOnly = TRUE)
summary(fit)
fitted(fit)
```

`fitTsfmMT` *Fit a market timing time series factor model*

Description

This is a wrapper function to fit a market timing time series factor model for one or more asset returns or excess returns using time series regression. Users can choose between ordinary least squares-LS, discounted least squares-DLS (or) robust regression. An object of class `"tsfm"` is returned.

Usage

```
fitTsfmMT(
  asset.names,
  mkt.name,
  rf.name = NULL,
  data = data,
  fit.method = c("LS", "DLS", "Robust"),
  control = fitTsfm.control(...),
  ...
)
```

Arguments

`asset.names`	vector containing syntactically valid names of assets, whose returns or excess returns are the dependent variable.
`mkt.name`	syntactically valid name of the column for market returns (required).
`rf.name`	syntactically valid name of the column of risk free rate variable to calculate excess returns for all assets (in `asset.names`) and the market factor (in `mkt.name`). Default is NULL, and no action is taken.
`data`	vector, matrix, data.frame, xts, timeSeries or zoo object containing column(s) named in `asset.names`, `factor.names` and optionally, `mkt.name` and `rf.name`.
`fit.method`	the estimation method, one of "LS", "DLS" or "Robust". See details. Default is "LS".
`control`	list of control parameters passed to `fitTsfm`. Refer to `fitTsfm.control` for details.
`...`	arguments passed to `fitTsfm.control`

Details

Market timing accounts for the price movement of the general stock market relative to fixed income securities. A market-timing factor is added to the time series regression, following Henriksson & Merton (1981). Here, we use down.market = max(0, R_f-R_m), where Rm is the (excess) return on the market. The coefficient of this down-market factor can be interpreted as the number of "free" put options on the market provided by the manager's market-timings skills.

Value

Similar to fitTsfm, fitTsfmMT also returns an object of class "tsfm", for which print, plot, predict and summary methods exist. The generic accessor functions coef, fitted, residuals and fmCov can be applied as well.

An object of class "tsfm" is a list containing the following components:

asset.fit	list of fitted objects for each asset. Each object is of class lm if fit.method="LS" or "DLS", class lmRob if the fit.method="Robust".
alpha	length-N vector of estimated alphas.
beta	N x 2 matrix of estimated betas.
r2	length-N vector of R-squared values.
resid.sd	length-N vector of residual standard deviations.
call	the matched function call.
data	xts data object containing the asset(s) and factor(s) returns.
asset.names	asset.names as input.
factor.names	vector containing the names of the market-timing factor and the market factor
mkt.name	mkt.name as input
fit.method	fit.method as input.

Where N is the number of assets and T is the number of time periods.

Author(s)

Yi-An Chen, Sangeetha Srinivasan.

References

Christopherson, J. A., Carino, D. R., & Ferson, W. E. (2009). Portfolio performance measurement and benchmarking. McGraw Hill Professional. pp.127-133

Henriksson, R. D., & Merton, R. C. (1981). On market timing and investment performance. II. Statistical procedures for evaluating forecasting skills. Journal of business, 513-533.

Treynor, J., & Mazuy, K. (1966). Can mutual funds outguess the market. Harvard business review, 44(4), 131-136.

See Also

The original time series factor model fitting function fitTsfm and related methods.

Examples

```
 # load data
data(managers, package = 'PerformanceAnalytics')

# example: Market-timing time series factor model with LS fit
fit <- fitTsfmMT(asset.names=colnames(managers[,(1:6)]),
                 mkt.name="SP500 TR", rf.name="US 3m TR",
```

```
                  data=managers)
summary(fit)
```

`fitTsfmUpDn` *Fit a up and down market factor model using time series regression*

Description

This is a wrapper function to fits a up and down market model for one or more asset returns or excess returns using time series regression. Users can choose between ordinary least squares-LS, discounted least squares-DLS (or) robust regression. An object of class `"tsfmUpDn"` is returned.

Usage

```
fitTsfmUpDn(
  asset.names,
  mkt.name,
  rf.name = NULL,
  data = data,
  fit.method = c("LS", "DLS", "Robust"),
  control = fitTsfm.control(...),
  ...
)
```

Arguments

`asset.names`	Vector containing syntactically valid names of assets, whose returns or excess returns are the dependent variable.
`mkt.name`	Syntactically valid name for market returns. Required for an up/down market model.
`rf.name`	Syntactically valid name of the risk free rate to calculate excess returns for all assets (in `asset.names`) and the market factor (in `mkt.name`). Default is `NULL`, and no action is taken.
`data`	vector, matrix, data.frame, xts, timeSeries or zoo object containing column(s) named in `asset.names`, `factor.names` and optionally, `mkt.name` and `rf.name`.
`fit.method`	the estimation method, one of "LS", "DLS" or "Robust". See details. Default is "LS".
`control`	list of control parameters. The default is constructed by the function `fitTsfm.control`. See the documentation for `fitTsfm.control` for details.
`...`	arguments passed to `fitTsfm.control`

Details

`fitTsfmUpDn` will use `fitTsfm` to fit a time series model for up and down market respectively. If risk free rate is provided, the up market is the excess market returns which is no less than 0. The goal of up and down market model is to capture two different market Betas in the up and down markets.

Value

`fitTsfmUpDn` returns an object `tsfmUpDn`. It supports generic function such as `summary`, `predict`, `plot` and `print`.

It is also a list object containing `Up` and `Dn`. Both `Up` and `Dn` are class of `"tsfm"`. As a result, for each list object, The generic function such as `print`, `plot`, `predict` and `summary` methods exist for both `Up` and `Dn`. Also, the generic accessor functions `coef`, `fitted`, `residuals` and `fmCov` can be applied as well.

An object of class `"tsfmUpDn"` is a list containing `Up` and `Dn`:

`Up`	An object of `tsfm` fitted by `fitTsfm` for the up market;
`Dn`	An object of `tsfm` fitted by `fitTsfm` for the down market;

and others useful items:

`call`	Function call.
`data`	Original data used but converted to `xts` class.

Each object of `tsfm` contains :

`asset.fit`	list of fitted objects for each asset. Each object is of class `lm` if `fit.method="LS"` or `"DLS"`, class `lmRob` if the `fit.method="Robust"`
`alpha`	length-N vector of estimated alphas.
`beta`	N x 1 matrix of estimated betas.
`r2`	length-N vector of R-squared values.
`resid.sd`	length-N vector of residual standard deviations.
`call`	the matched function call.
`data`	xts data object containing the assets and factors.
`asset.names`	asset.names as input.
`factor.names`	factor.names as input.
`fit.method`	fit.method as input.

Where N is the number of assets and T is the number of time periods.

Author(s)

Yi-An Chen.

References

Christopherson, J. A., Carino, D. R., & Ferson, W. E. (2009). Portfolio performance measurement and benchmarking. McGraw Hill Professional.

See Also

The `tsfmUpDn` methods for generic functions: `plot.tsfmUpDn`, `predict.tsfmUpDn`, `print.tsfmUpDn` and `summary.tsfmUpDn`.

The original time series function `fitTsfm` and its generic functions application.

Examples

```
 # load data
data(managers, package = 'PerformanceAnalytics')

# example: Up and down market factor model with LS fit
fitUpDn <- fitTsfmUpDn(asset.names = colnames(managers[,(1:6)]),
                       mkt.name = "SP500 TR",
                       data = managers,
                       fit.method = "LS")

 print(fitUpDn)
 summary(fitUpDn)

 # A list object
 fitUpDn
 summary(fitUpDn$Up)
 summary(fitUpDn$Dn)
```

`fmCov` *Covariance Matrix for assets' returns from fitted factor model.*

Description

Computes the covariance matrix for assets' returns based on a fitted factor model. This is a generic function with methods for classes `tsfm`, `sfm` and `ffm`.

Usage

```
fmCov(object, ...)

## S3 method for class 'tsfm'
fmCov(object, factor.cov, use = "pairwise.complete.obs", ...)

## S3 method for class 'sfm'
fmCov(object, use = "pairwise.complete.obs", ...)

## S3 method for class 'ffm'
fmCov(object, use = "pairwise.complete.obs", ...)
```

Arguments

`object`	fit object of class `tsfm`, `sfm` or `ffm`.
`...`	optional arguments passed to `cov`.
`factor.cov`	factor covariance matrix (optional); defaults to the sample covariance matrix.
`use`	method for computing covariances in the presence of missing values; one of "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs". Default is "pairwise.complete.obs".

Details

R(i, t), the return on asset i at time t, is assumed to follow a factor model of the form,

R(i,t) = alpha(i) + beta(i)*f(t) + e(i,t),

where, alpha(i) is the intercept, f(t) is a K x 1 vector of factor returns at time t, beta(i) is a 1 x K vector of factor exposures and the error terms e(i,t) are serially uncorrelated across time and contemporaneously uncorrelated across assets so that e(i,t) ~ iid(0,sig(i)^2). Thus, the variance of asset i's return is given by

var(R(i)) = beta(i)*cov(F)*tr(beta(i)) + sig(i)^2.

And, the N x N covariance matrix of asset returns is

var(R) = B*cov(F)*tr(B) + D,

where, B is the N x K matrix of factor betas and D is a diagonal matrix with sig(i)^2 along the diagonal.

The method for computing covariance can be specified via the ... argument. Note that the default of use="pairwise.complete.obs" for handling NAs restricts the method to "pearson".

Value

The computed N x N covariance matrix for asset returns based on the fitted factor model.

Author(s)

Eric Zivot, Yi-An Chen and Sangeetha Srinivasan.

References

Zivot, E., & Jia-hui, W. A. N. G. (2006). Modeling Financial Time Series with S-Plus Springer-Verlag.

See Also

fitTsfm

cov for more details on arguments use and method.

Examples

```
# Time Series Factor model example
 # load data
data(managers, package = 'PerformanceAnalytics')
 # Make syntactically valid column names
colnames(managers)
colnames(managers) <- make.names( colnames(managers))
colnames(managers)
```

```
fit <- fitTsfm(asset.names = colnames(managers[, (1:6)]),
               factor.names = c("EDHEC.LS.EQ","SP500.TR"),
               data = managers)
fmCov(fit)
```

`fmEsDecomp` *Decompose ES into individual factor contributions*

Description

Compute the factor contributions to Expected Tail Loss or Expected Shortfall (ES) of assets' returns based on Euler's theorem, given the fitted factor model. The partial derivative of ES with respect to factor beta is computed as the expected factor return given fund return is less than or equal to its value-at-risk (VaR). Option to choose between non-parametric and Normal.

Usage

```
fmEsDecomp(object, ...)

## S3 method for class 'tsfm'
fmEsDecomp(
  object,
  factor.cov,
  p = 0.05,
  type = c("np", "normal"),
  use = "pairwise.complete.obs",
  ...
)

## S3 method for class 'sfm'
fmEsDecomp(
  object,
  factor.cov,
  p = 0.05,
  type = c("np", "normal"),
  use = "pairwise.complete.obs",
  ...
)

## S3 method for class 'ffm'
fmEsDecomp(
  object,
  factor.cov,
  p = 0.05,
  type = c("np", "normal"),
  use = "pairwise.complete.obs",
  ...
)
```

Arguments

object	fit object of class tsfm, sfm or ffm.
...	other optional arguments passed to quantile.
factor.cov	optional user specified factor covariance matrix with named columns; defaults to the sample covariance matrix.
p	tail probability for calculation. Default is 0.05.
type	one of "np" (non-parametric) or "normal" for calculating VaR. Default is "np".
use	method for computing covariances in the presence of missing values; one of "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs". Default is "pairwise.complete.obs".

Details

The factor model for an asset's return at time t has the form

R(t) = beta'f(t) + e(t) = beta.star'f.star(t)

where, beta.star=(beta,sig.e) and f.star(t)=[f(t)',z(t)]'. By Euler's theorem, the ES of the asset's return is given by:

ES.fm = sum(cES_k) = sum(beta.star_k*mES_k)

where, summation is across the K factors and the residual, cES and mES are the component and marginal contributions to ES respectively. The marginal contribution to ES is defined as the expected value of F.star, conditional on the loss being less than or equal to VaR.fm. This is estimated as a sample average of the observations in that data window.

Refer to Eric Zivot's slides (referenced) for formulas pertaining to the calculation of Normal ES (adapted from a portfolio context to factor models).

Value

A list containing

ES.fm	length-N vector of factor model ES of N-asset returns.
mES	N x (K+1) matrix of marginal contributions to VaR.
cES	N x (K+1) matrix of component contributions to VaR.
pcES	N x (K+1) matrix of percentage component contributions to VaR.

Where, K is the number of factors and N is the number of assets.

Author(s)

Eric Zviot, Sangeetha Srinivasan and Yi-An Chen

References

Epperlein, E., & Smillie, A. (2006). Portfolio risk analysis Cracking VAR with kernels. RISK-LONDON-RISK MAGAZINE LIMITED-, 19(8), 70.

Hallerback (2003). Decomposing Portfolio Value-at-Risk: A General Analysis. The Journal of Risk, 5(2), 1-18.

Meucci, A. (2007). Risk contributions from generic user-defined factors. RISK-LONDON-RISK MAGAZINE LIMITED-, 20(6), 84.

Yamai, Y., & Yoshiba, T. (2002). Comparative analyses of expected shortfall and value-at-risk: their estimation error, decomposition, and optimization. Monetary and economic studies, 20(1), 87-121.

See Also

fitTsfm for the different factor model fitting functions.

fmSdDecomp for factor model SD decomposition. fmVaRDecomp for factor model VaR decomposition.

Examples

```
 # Time Series Factor Model
 # load data
data(managers, package = 'PerformanceAnalytics')

fit.macro <- fitTsfm(asset.names=colnames(managers[,(1:6)]),
                     factor.names=colnames(managers[,(7:8)]),
                     data=managers)

ES.decomp <- fmEsDecomp(fit.macro)

# get the component contributions
ES.decomp$cES
```

---

fmSdDecomp *Decompose standard deviation into individual factor contributions*

Description

Compute the factor contributions to standard deviation (SD) of assets' returns based on Euler's theorem, given the fitted factor model.

Usage

```
fmSdDecomp(object, ...)

## S3 method for class 'tsfm'
fmSdDecomp(object, factor.cov, use = "pairwise.complete.obs", ...)
```

```
## S3 method for class 'sfm'
fmSdDecomp(object, factor.cov, use = "pairwise.complete.obs", ...)

## S3 method for class 'ffm'
fmSdDecomp(object, factor.cov, ...)
```

Arguments

object	fit object of class tsfm or ffm.
...	optional arguments passed to cov.
factor.cov	optional user specified factor covariance matrix with named columns; defaults to the sample covariance matrix.
use	method for computing covariances in the presence of missing values; one of "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs". Default is "pairwise.complete.obs".

Details

The factor model for an asset's return at time t has the form

R(t) = beta'f(t) + e(t) = beta.star'f.star(t)

where, beta.star=(beta,sig.e) and f.star(t)=[f(t)',z(t)]'.

By Euler's theorem, the standard deviation of the asset's return is given as:

Sd.fm = sum(cSd_k) = sum(beta.star_k*mSd_k)

where, summation is across the K factors and the residual, cSd and mSd are the component and marginal contributions to SD respectively. Computing Sd.fm and mSd is very straight forward. The formulas are given below and details are in the references. The covariance term is approximated by the sample covariance.

Sd.fm = sqrt(beta.star''cov(F.star)beta.star)
mSd = cov(F.star)beta.star / Sd.fm

Value

A list containing

Sd.fm	length-N vector of factor model SDs of N-asset returns.
mSd	N x (K+1) matrix of marginal contributions to SD.
cSd	N x (K+1) matrix of component contributions to SD.
pcSd	N x (K+1) matrix of percentage component contributions to SD.

Where, K is the number of factors and N is the number of assets.

Author(s)

Eric Zivot, Yi-An Chen and Sangeetha Srinivasan

References

Hallerback (2003). Decomposing Portfolio Value-at-Risk: A General Analysis. The Journal of Risk, 5(2), 1-18.

Meucci, A. (2007). Risk contributions from generic user-defined factors. RISK-LONDON-RISK MAGAZINE LIMITED-, 20(6), 84.

Yamai, Y., & Yoshiba, T. (2002). Comparative analyses of expected shortfall and value-at-risk: their estimation error, decomposition, and optimization. Monetary and economic studies, 20(1), 87-121.

See Also

`fitTsfm` for the different factor model fitting functions.

`fmCov` for factor model covariance. `fmVaRDecomp` for factor model VaR decomposition. `fmEsDecomp` for factor model ES decomposition.

Examples

```
# Time Series Factor Model

 # load data
data(managers, package = 'PerformanceAnalytics')
colnames(managers)
 # Make syntactically valid column names
colnames(managers) <- make.names( colnames(managers))
colnames(managers)

fit.macro <- fitTsfm(asset.names=colnames(managers[,(1:6)]),
                     factor.names=colnames(managers[,(7:9)]),
                     rf.name="US.3m.TR", data=managers)
decomp <- fmSdDecomp(fit.macro)
# get the percentage component contributions
decomp$pcSd
```

---

fmVaRDecomp *Decompose VaR into individual factor contributions*

Description

Compute the factor contributions to Value-at-Risk (VaR) of assets' returns based on Euler's theorem, given the fitted factor model. The partial derivative of VaR w.r.t. factor beta is computed as the expected factor return given fund return is equal to its VaR and approximated by a kernel estimator. Option to choose between non-parametric and Normal.

Usage

```
fmVaRDecomp(object, ...)

## S3 method for class 'tsfm'
fmVaRDecomp(
  object,
  factor.cov,
  p = 0.05,
  type = c("np", "normal"),
  use = "pairwise.complete.obs",
  ...
)

## S3 method for class 'sfm'
fmVaRDecomp(
  object,
  factor.cov,
  p = 0.05,
  type = c("np", "normal"),
  use = "pairwise.complete.obs",
  ...
)

## S3 method for class 'ffm'
fmVaRDecomp(
  object,
  factor.cov,
  p = 0.05,
  type = c("np", "normal"),
  use = "pairwise.complete.obs",
  ...
)
```

Arguments

object	fit object of class tsfm, sfm or ffm.
...	other optional arguments passed to quantile.
factor.cov	optional user specified factor covariance matrix with named columns; defaults to the sample covariance matrix.
p	tail probability for calculation. Default is 0.05.
type	one of "np" (non-parametric) or "normal" for calculating VaR. Default is "np".
use	method for computing covariances in the presence of missing values; one of "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs". Default is "pairwise.complete.obs".

Details

The factor model for an asset's return at time `t` has the form

`R(t) = beta'f(t) + e(t) = beta.star'f.star(t)`

where, `beta.star=(beta,sig.e)` and `f.star(t)=[f(t)',z(t)]'`. By Euler's theorem, the VaR of the asset's return is given by:

`VaR.fm = sum(cVaR_k) = sum(beta.star_k*mVaR_k)`

where, summation is across the `K` factors and the residual, `cVaR` and `mVaR` are the component and marginal contributions to `VaR` respectively. The marginal contribution to VaR is defined as the expectation of `F.star`, conditional on the loss being equal to `VaR.fm`. This is approximated as described in Epperlein & Smillie (2006); a triangular smoothing kernel is used here.

Refer to Eric Zivot's slides (referenced) for formulas pertaining to the calculation of Normal VaR (adapted from a portfolio context to factor models)

Value

A list containing

`VaR.fm`	length-N vector of factor model VaRs of N-asset returns.
`n.exceed`	length-N vector of number of observations beyond VaR for each asset.
`idx.exceed`	list of numeric vector of index values of exceedances.
`mVaR`	N x (K+1) matrix of marginal contributions to VaR.
`cVaR`	N x (K+1) matrix of component contributions to VaR.
`pcVaR`	N x (K+1) matrix of percentage component contributions to VaR.

Where, `K` is the number of factors and N is the number of assets.

Author(s)

Eric Zivot, Yi-An Chen and Sangeetha Srinivasan

References

Hallerback (2003). Decomposing Portfolio Value-at-Risk: A General Analysis. The Journal of Risk, 5(2), 1-18.

Meucci, A. (2007). Risk contributions from generic user-defined factors. RISK-LONDON-RISK MAGAZINE LIMITED-, 20(6), 84.

Yamai, Y., & Yoshiba, T. (2002). Comparative analyses of expected shortfall and value-at-risk: their estimation error, decomposition, and optimization. Monetary and economic studies, 20(1), 87-121.

See Also

`fitTsfm` for the different factor model fitting functions.

`fmSdDecomp` for factor model SD decomposition. `fmEsDecomp` for factor model ES decomposition.

Examples

```
# Time Series Factor Model

 # load data
data(managers, package = 'PerformanceAnalytics')
colnames(managers)
 # Make syntactically valid column names
colnames(managers) <- make.names( colnames(managers))
colnames(managers)

fit.macro <- fitTsfm(asset.names=colnames(managers[,(1:6)]),
                     factor.names=colnames(managers[,(7:8)]),
                     data=managers)

VaR.decomp <- fmVaRDecomp(fit.macro)

# get the component contributions
VaR.decomp$cVaR
```

---

paFm *Compute cumulative mean attribution for factor models*

Description

Decompose total returns into returns attributed to factors and specific returns. An object of class "pafm" is generated, with methods for generic functions plot, summary and print.

Usage

```
paFm(fit, ...)
```

Arguments

fit	an object of class tsfm, sfm or ffm.
...	other arguments/controls passed to the fit methods.

Details

Total returns can be decomposed into returns attributed to factors and specific returns.
$R_t = \sum b_k * f_k t + u_t, t = 1...T$
b_k is exposure to factor k and f_kt is factor k's return at time t. The return attributed to factor k is b_k * f_kt and specific return is u_t.

Value

The returned object is of class "pafm" containing

cum.ret.attr.f	N X K matrix of cumulative return attributed to factors.
cum.spec.ret	length-N vector of cumulative specific returns.
attr.list	list of time series of attributed returns for every portfolio.

Author(s)

Yi-An Chen and Sangeetha Srinivasan

References

Grinold, R. and Kahn, R. (1999) Active Portfolio Management: A Quantitative Approach for Producing Superior Returns and Controlling Risk. McGraw-Hill.

See Also

fitTsfm for the factor model fitting functions.

The pafm methods for generic functions: plot.pafm, print.pafm and summary.pafm.

Examples

```
data(managers, package = 'PerformanceAnalytics')
fit <- fitTsfm(asset.names=colnames(managers[, (1:6)]),
               factor.names=c("EDHEC LS EQ","SP500 TR"),
               data=managers)
# without benchmark
paFm(fit)
```

---

plot.pafm *plot "pafm" object*

Description

Generic function of plot method for paFm. Either plot all assets or choose a single asset to plot.

Usage

```
## S3 method for class 'pafm'
plot(
  x,
  which.plot = c("none", "1L", "2L", "3L"),
  max.show = 6,
  date = NULL,
  plot.single = FALSE,
```

```
  fundName,
  which.plot.single = c("none", "1L", "2L", "3L"),
  ...
)
```

Arguments

x	object of class "pafm" created by paFm.
which.plot	Integer indicates which plot to create: "none" will create a menu to choose. Defualt is none. 1 = attributed cumulative returns, 2 = attributed returns on date selected by user, 3 = time series of attributed returns
max.show	Maximum assets to plot. Default is 6.
date	Indicates for attributed returns, the date format should be xts compatible.
plot.single	Plot a single asset of lm class. Defualt is FALSE.
fundName	Name of the portfolio to be plotted.
which.plot.single	Integer indicates which plot to create: "none" will create a menu to choose. Defualt is none. 1 = attributed cumulative returns, 2 = attributed returns on date selected by user, 3 = time series of attributed returns
...	more arguements for chart.TimeSeries used for plotting time series

Value

plot.pafm returns a plot for an object of class pafm.

Author(s)

Yi-An Chen.

---

plot.tsfm *Plots from a fitted time series factor model*

Description

Generic plot method for object of class tsfm. Plots chosen characteristic(s) for one or more assets.

Usage

```
## S3 method for class 'tsfm'
plot(
  x,
  which = NULL,
  f.sub = 1:2,
  a.sub = 1:6,
  plot.single = FALSE,
  asset.name,
  colorset = c("royalblue", "dimgray", "olivedrab", "firebrick", "goldenrod",
    "mediumorchid", "deepskyblue", "chocolate", "darkslategray"),
  legend.loc = "topleft",
  las = 1,
  lwd = 2,
  maxlag = 15,
  ...
)
```

Arguments

x	an object of class tsfm produced by fitTsfm.
which	a number to indicate the type of plot. If a subset of the plots is required, specify a subset of the numbers 1:12 for group plots and 1:19 for individual plots. If which=NULL (default), the following menu appears:

For plots of a group of assets:
1 = Factor model coefficients: Alpha,
2 = Factor model coefficients: Betas,
3 = Actual and fitted,
4 = R-squared,
5 = Residual volatility,
6 = Scatterplot matrix of residuals, with histograms, density overlays, correlations and significance stars,
7 = Factor model residual correlation
8 = Factor model return correlation,
9 = Factor contribution to SD,
10 = Factor contribution to ES,
11 = Factor contribution to VaR,
12 = Asset returns vs factor returns (single factor model)

For individual asset plots:
1 = Actual and fitted,
2 = Actual vs fitted,
3 = Residuals vs fitted,
4 = Sqrt. of modified residuals vs fitted,
5 = Residuals with standard error bands,
6 = Time series of squared residuals,
7 = Time series of absolute residuals,

8 = SACF and PACF of residuals,
9 = SACF and PACF of squared residuals,
10 = SACF and PACF of absolute residuals,
11 = Non-parametric density of residuals with normal overlaid,
12 = Non-parametric density of residuals with skew-t overlaid,
13 = Histogram of residuals with non-parametric density and normal overlaid,
14 = QQ-plot of residuals,
15 = CUSUM test-Recursive residuals, requires strucchange package,
16 = CUSUM test-LS residuals, requires strucchange package,
17 = Recursive estimates (RE) test of LS regression coefficients, requires strucchange package,
18 = Rolling regression over a 24-period observation window,
19 = Asset returns vs factor returns (single factor model)

`f.sub`	numeric/character vector; subset of indexes/names of factors to include for group plots. Default is 1:2.
`a.sub`	numeric/character vector; subset of indexes/names of assets to include for group plots. At least 2 assets must be selected. Default is 1:6.
`plot.single`	logical; If `TRUE` plots the characteristics of an individual asset's factor model. The type of plot is given by `which`. Default is `FALSE`.
`asset.name`	name of the individual asset to be plotted. Is necessary if `x` contains multiple asset fits and `plot.single=TRUE`.
`colorset`	color palette to use for all the plots. The 1st element will be used for individual time series plots or the 1st object plotted, the 2nd element for the 2nd object in the plot and so on.
`legend.loc`	places a legend into one of nine locations on the chart: "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right", or "center". Default is "bottomright". Use `legend.loc=NULL` to suppress the legend.
`las`	one of `0, 1, 2, 3` to set the direction of axis labels, same as in `plot`. Default is 1.
`lwd`	set the line width, same as in `plot`. Default is 2.
`maxlag`	optional number of lags to be calculated for ACF. Default is 15.
`...`	further arguments to be passed to other plotting functions.

Details

The function can be used for group plots and individual plots. User can select the type of plot either from the menu prompt (default) or directly via argument `which`.

In case multiple plots are needed, the menu is repeated after each plot (enter 0 to exit). User can also input a numeric vector of plot options via `which`.

Group plots are the default. The selected assets in `a.sub` and selected factors in `f.sub` are plotted depending on the characteristic chosen. The default is to show the first 2 factors and first 6 assets.

Setting `plot.single=TRUE` enables individual plots. If there is more than one asset fit by `x`, `asset.name` should be specified. In case the `tsfm` object `x` contains only a single asset fit, plot.tsfm can infer `asset.name` without user input.

CUSUM plots (individual asset plot options 15, 16 and 17) are applicable only for `fit.method="LS"`.

Modified residuals, rolling regression and single factor model plots (individual asset plot options 4, 18 and 19) are not applicable for `variable.selection="lars"`.

The last option for plotting asset returns vs. factor returns (individual asset plot option 19 and group plot 12) are only applicable for single factor models.

Value

`plot.tsfm` returns a plot for an object of class `tsfm`.

Author(s)

Eric Zivot, Sangeetha Srinivasan and Yi-An Chen

See Also

`fitTsfm`, `residuals.tsfm`, `fitted.tsfm`, `fmCov.tsfm` and `summary.tsfm` for time series factor model fitting and related S3 methods. Refer to `fmSdDecomp`, `fmEsDecomp`, `fmVaRDecomp` for factor model risk measures.

Here is a list of plotting functions used. (I=individual, G=Group) I(1,5,6,7), G(3) - `chart.TimeSeries`, I(2,3,4,19), G(12) - `plot.default`, I(3,4) - `panel.smooth`, I(8,9,10) - `chart.ACFplus`, I(11,12) - `plot.density`, I(13) - `chart.Histogram`, I(14) - `chart.QQPlot`, I(15,16,17) - `plot.efp` (requires strucchange package), I(18) - `plot.zoo`, G(1,2,4,5,9,10,11) - `barchart`, G(6) - `chart.Correlation` and G(7,8) - `corrplot.mixed` (requires corrplot package).

---

plot.tsfmUpDn *Plot actual against fitted values of up and down market time series factor model*

Description

Generic `plot` method for object of class `tsfmUpDn`.

Usage

```
## S3 method for class 'tsfmUpDn'
plot(
  x,
  asset.name = NULL,
  SFM.line = FALSE,
  LSandRob = FALSE,
  line.color = c("blue", "purple"),
  line.type = c("dashed", "solid"),
  line.width = c(1, 2),
  sfm.line.type = "dashed",
  add.legend = TRUE,
  legend.loc = "topleft",
  legend.cex = 0.9,
  ...
)
```

Arguments

x	an object of class tsfmUpDn produced by fitTsfmUpDn.
asset.name	A vector of character to show single or multiple assets names. The defualt if NULL.
SFM.line	A logic flag to add a fitted single factor model. The default is FALSE.
LSandRob	A logic flag to add a comparison Up/Down factor model. If the original model is "LS", the comparison model is "Robust" and vice versa. The default is FALSE. The default is FALSE.
line.color	A vector of color codes of up/dn fitted line. The first element is for the object fitted line and the second for the comparison fitted line. The default is c("blue","purple").
line.type	A vector of line types of up/dn fitted line. The first is for the object fitted line and the second for the comparison fitted line. The default is c("dashed","solid".
line.width	A vector of line width of up/dn fitted line. The first element is for the object fitted line and the second element for the comparison fitted line. The default is c(1,2.
sfm.line.type	SFM line type. The default is "dashed"
add.legend	A logic flag to add a legend. The default is TRUE.
legend.loc	The default is "topleft".
legend.cex	cex of legend.
...	Other arguments can be used in plot. Please refer to plot.

Details

This method plots actual values against fitted value of up and down market time series factor model. The dots are actual values and the dashed lines are fitted values. Users can choose to add a single market factor model and a robust up and down model for comaprsion.

For other types of plots, use the list objects Up and Dn of class tsfmUpDn. The plot.tsfm can be applied.

Value

plot.tsfmUpDn returns a plot for an object of class tsfmUpDn.

Author(s)

Yi-An Chen

See Also

fitTsfmUpDn

predict.tsfm *Predicts asset returns based on a fitted time series factor model*

Description

S3 predict method for object of class tsfm. It calls the predict method for fitted objects of class lm, lmRob or lars as appropriate.

Usage

```
## S3 method for class 'tsfm'
predict(object, newdata = NULL, ...)
```

Arguments

object	an object of class tsfm produced by fitTsfm.
newdata	a vector, matrix, data.frame, xts, timeSeries or zoo object containing the variables with which to predict.
...	optional arguments passed to predict.lm or predict.lmrob, such as se.fit, or, to predict.lars such as mode.

Value

predict.tsfm produces a matrix of return predictions, if all assets have equal history. If not, a list of predicted return vectors of unequal length is produced.

Author(s)

Yi-An Chen and Sangeetha Srinivasan

See Also

fitTsfm, summary.tsfm

Examples

```
# load data from the database
data(managers, package = 'PerformanceAnalytics')

# fit the factor model with LS
fit <- fitTsfm(asset.names = colnames(managers[,(1:6)]),
               factor.names = c("EDHEC LS EQ","SP500 TR"),
               data = managers)

predict_fit <- predict(fit)

newdata <- data.frame(rnorm(n=NROW(fit$data)), rnorm(n=NROW(fit$data)))
colnames(newdata) <- c("EDHEC LS EQ", "SP500 TR")
```

```
rownames(newdata) <- zoo::index(fit$data)

predict_fit_2 <- predict(fit, newdata, interval = "confidence")
```

`predict.tsfmUpDn` *Predicts asset returns based on a fitted up and down market time series factor model*

Description

S3 `predict` method for object of class `tsfmUpDn`. It calls the `predict.tsfm` method for a list object of `Up` and `Dn`

Usage

```
## S3 method for class 'tsfmUpDn'
predict(object, ...)
```

Arguments

`object`	an object of class `tsfmUpDn` produced by `fitTsfmUpDn`.
`...`	optional arguments passed to `predict.lm` or `predict.lmrob`, such as `se.fit`, or, to `predict.lars` such as `mode`.

Value

`predict.tsfmUpDm` produces a list of `Up` and `Dn`. Both `Up` and `Dn` contain a vector or a matrix of predictions.

Author(s)

Yi-An Chen and Sangeetha Srinivasan

See Also

`predict.tsfm`,`fitTsfmUpDn`, `summary.tsfmUpDn`

Examples

```
 # load data
data(managers, package = 'PerformanceAnalytics')

# fit the factor model with LS. example: Up and down market factor model with LS fit
fitUpDn <- fitTsfmUpDn(asset.names = colnames(managers[,(1:6)]),
                       mkt.name = "SP500 TR",
                       data = managers,
                       fit.method = "LS")
```

```
predict(fitUpDn)
```

`print.pafm` *Print object of class* "pafm".

Description

Generic function of print method for paFm.

Usage

```
## S3 method for class 'pafm'
print(x, ...)
```

Arguments

x	object of class "pafm" created by paFm.
...	Other arguments for print methods.

Value

print.pafm prints a brief summary of an object of class pafm.

Author(s)

Yi-An Chen.

Examples

```
# load data from the database
 data(managers, package = 'PerformanceAnalytics')
# fit the factor model with LS
fit <- fitTsfm(asset.names=colnames(managers[,(1:6)]),
               factor.names=c("EDHEC LS EQ", "SP500 TR"),
               data=managers)
fm.attr <- paFm(fit)
print(fm.attr)
```

print.tsfm *Prints a fitted time series factor model*

Description

S3 print method for object of class tsfm. Prints the call, factor model dimension, regression coefficients, r-squared and residual volatilities from the fitted object.

Usage

```
## S3 method for class 'tsfm'
print(x, digits = max(3, .Options$digits - 3), ...)
```

Arguments

x	an object of class tsfm produced by fitTsfm.
digits	an integer value, to indicate the required number of significant digits. Default is 3.
...	optional arguments passed to the print method.

Value

print.tsfm prints a brief summary of an object of class tsfm.

Author(s)

Yi-An Chen and Sangeetha Srinivasan

See Also

fitTsfm, summary.tsfm

Examples

```
data(managers, package = 'PerformanceAnalytics')
fit <- fitTsfm(asset.names=colnames(managers[,(1:6)]),
               factor.names=colnames(managers[,7:9]),
               mkt.name="SP500.TR", data=managers)
print(fit)
```

print.tsfmUpDn *Prints out a fitted up and down market time series factor model object*

Description

S3 print method for object of class tsfmUpDn. Prints the call, factor model dimension, regression coefficients, r-squared and residual volatilities from the fitted object.

Usage

```
## S3 method for class 'tsfmUpDn'
print(x, digits = max(3, .Options$digits - 3), ...)
```

Arguments

x	an object of class tsfmUpDn produced by fitTsfmUpDn.
digits	an integer value, to indicate the required number of significant digits. Default is 3.
...	optional arguments passed to the print method.

Value

print.tsfmUpDn prints a brief summary of an object of class tsfmUpDn.

Author(s)

Yi-An Chen and Sangeetha Srinivasan

See Also

fitTsfmUpDn, summary.tsfmUpDn

Examples

```
 # load data
data(managers, package = 'PerformanceAnalytics')
colnames(managers)
 # Make syntactically valid column names
colnames(managers) <- make.names( colnames(managers))
colnames(managers)

# example: Up and down market factor model with LS fit
fitUpDn <- fitTsfmUpDn(asset.names=colnames(managers[,(1:6)]),
                       mkt.name="SP500.TR",
                       data=managers,
                       fit.method="LS")

print(fitUpDn)
```

summary.pafm *summary* "pafm" *object.*

Description

Generic function of summary method for paFm.

Usage

```
## S3 method for class 'pafm'
summary(object, digits = max(3, .Options$digits - 3), ...)
```

Arguments

object	"pafm" object created by paFm.
digits	integer indicating the number of decimal places. Default is 3.
...	Other arguments for print methods.

Value

Returns an object of class summary.pafm. The print method for class summary.pafm outputs the means of the specific returns of the factors.

Author(s)

Yi-An Chen.

Examples

```
# load data from the database
data(managers, package = 'PerformanceAnalytics')

# fit the factor model with LS
fit.ts <- fitTsfm(asset.names = colnames(managers[,(1:6)]),
                  factor.names = c("EDHEC LS EQ","SP500 TR"),
                  data = managers)

fm.attr <- paFm(fit.ts)
summary(fm.attr)
```

summary.tsfm *Summarizing a fitted time series factor model*

Description

summary method for object of class tsfm. Returned object is of class summary.tsfm.

Usage

```
## S3 method for class 'tsfm'
summary(object, se.type = c("Default", "HC", "HAC"), ...)

## S3 method for class 'summary.tsfm'
print(x, digits = 3, labels = TRUE, ...)
```

Arguments

object	an object of class tsfm returned by fitTsfm.
se.type	one of "Default", "HC" or "HAC" option for computing HC/HAC standard errors and t-statistics. Default is "Default". If "HC" or "HAC" options are selected, you will need to first load the suggested 'lmtest' package.
...	futher arguments passed to or from other methods.
x	an object of class summary.tsfm.
digits	number of significant digits to use when printing. Default is 3.
labels	option to print labels and legend in the summary. Default is TRUE. When FALSE, only the coefficient matrx with standard errors is printed.

Details

The default summary method for a fitted lm object computes the standard errors and t-statistics under the assumption of homoskedasticty. Argument se.type gives the option to compute heteroskedasticity-consistent (HC) or heteroskedasticity-autocorrelation-consistent (HAC) standard errors and t-statistics using coeftest. This option is meaningful only if fit.method = "LS" or "DLS".

Standard errors are currently not available for variable.selection="lars" as there seems to be no consensus on a statistically valid method of calculating standard errors for the lasso predictions.

Value

Returns an object of class summary.tsfm. The print method for class summary.tsfm outputs the call, coefficients (with standard errors and t-statistics), r-squared and residual volatilty (under the homoskedasticity assumption) for all assets.

Object of class summary.tsfm is a list of length N + 2 containing:

call	the function call to fitTsfm
se.type	standard error type as input
sum.list	list of summaries of the N fit objects (of class lm, lmRob or lars) for each asset in the factor model.

Author(s)

Sangeetha Srinivasan & Yi-An Chen.

See Also

fitTsfm, summary.lm

Examples

```
 # load data
data(managers, package = 'PerformanceAnalytics')

# fit for first 3 assets
fit <- fitTsfm(asset.names=colnames(managers[,1:3]),
               factor.names=colnames(managers[,7:9]),
               data=managers)

# summary of factor model fit for all assets
summary(fit)

# summary of factor model fit for the second of three
summary(fit$asset.fit[[2]])
```

---

summary.tsfmUpDn *Summarizing a fitted up and down market time series factor model*

Description

summary method for object of class tsfmUpDn. Returned object is of class summary.tsfmUpDn. This function provides a summary method to an object returned by a wrapper function fitTsfmUpDn.

Usage

```
## S3 method for class 'tsfmUpDn'
summary(object, ...)

## S3 method for class 'summary.tsfmUpDn'
print(x, digits = 3, ...)
```

Arguments

object	an object of class tsfmUpDn returned by fitTsfmUpDn.
...	futher arguments passed to or from summary.tsfm methods.
x	an object of class summary.tsfmUpDn.
digits	number of significants digits to use when printing. Default is 3.

Details

Since `fitTsfmUpDn` fits both up market and down market, `summary.tsfmUpDn` applies `summary.tsfm` for both markets fitted objects and combines the coefficients interested together.

Value

Returns an object of class `summary.tsfmUpDn`. This object contains a list object of `Up` and `Dn` for up market and down market respectively.

The print method for class `summary.tsfmUpDn` outputs the call, coefficients (with standard errors and t-statistics), r-squared and residual volatilty (under the homoskedasticity assumption) for all assets in up and down market.

Object of class `summary.tsfmUpDn` is a list of 2 containing:

`Up`	A list of the up market fitted object. It is a class of `summary.tsfm`
`Dn`	A list of the down market fitted object. It is a class of `summary.tsfm`

Author(s)

Yi-An Chen and Sangeetha Srinivasan.

See Also

`fitTsfmUpDn`, `summary.tsfm`

Index