Package 'factorAnalytics'

March 10, 2015

Type Package

Title Factor Analytics

Version 2.0.11

Date 2015-03-10

Author Eric Zivot, Sangeetha Srinivasan and Yi-An Chen

Maintainer Sangeetha Srinivasan <sangee@uw.edu>

Description An R package for the estimation and risk analysis of linear factor
models for asset returns and portfolios. It contains model fitting methods
for the three major types of factor models: time series (or, macroeconomic)
factor model, fundamental factor model and statistical factor model. They
allow for different types of distributions to be specified for modeling the
fat-tailed behavior of financial returns, including Edgeworth expansions.
Risk analysis measures such as VaR and ES, as well as performance
attribution for factor models (factor-contributed vs idiosyncratic returns) are included.

License GPL-2

Depends R (>= 3.0.0),xts (>= 0.9)

Imports
PerformanceAnalytics(>= 1.1.0),corrplot,robust,leaps,lars,strucchange,lmtest,sandwich,lattice,MASS

Suggests testthat, quantmod, knitr

LazyLoad yes

LazyDataCompression xz

URL http://r-forge.r-project.org/R/?group_id=579

R topics documented:

CommonFactors   *Factor set of several commonly used factors*

Description

Collection of common factors as both monthly and quarterly time series

- SP500: S&P 500 composite index returns. (Yahoo)

- GS10TR: US Treasury 10y yields total returns from the yeild of the 10 year constant maturity. (FRED)

- USD.Index: Trade Weighted U.S. Dollar Index: Major Currencies - TWEXMMTH. (FRED)

- Term.Spread: Yield spread of Merrill Lynch High-Yield Corporate Master II Index minus 10-year Treasury. (FRED)

- TED.Spread: 3-Month Treasury Bill: Secondary Market Rate(TB3MS) - 3-Month Eurodollar Deposit Rate (MED3). (FRED)

- dVIX: First difference of the end-of-month value of the CBOE Volatility Index (VIX). (Yahoo)

- OILPRICE: Monthly returns of spot price of West Texas Intermediate. (FRED)

- TB3MS: 3-Month Treasury Bill Secondary Market Rate (TB3MS). (FRED)

Usage

    data(CommonFactors)

Format

xts time series object

`factors.M` Jan-1997 through May-2014

`factors.Q` Q1-1997 through Q1-2014

Source

- Federal Reserve Economic Data (FRED): <http://research.stlouisfed.org/fred2/>
- Yahoo Finance: <http://finance.yahoo.com/>

---

dCornishFisher	*Cornish-Fisher expansion*

Description

Density, distribution function, quantile function and random generation using Cornish-Fisher approximation.

Usage

```
dCornishFisher(x, n, skew, ekurt)

pCornishFisher(q, n, skew, ekurt)

qCornishFisher(p, n, skew, ekurt)

rCornishFisher(n, sigma, skew, ekurt, seed = NULL)
```

Arguments

`x,q`	vector of standardized quantiles.
`n`	scalar; number of simulated values in random simulation, sample length in density, distribution and quantile functions.
`skew`	scalar; skewness.
`ekurt`	scalar; excess kurtosis.
`p`	vector of probabilities.
`sigma`	scalar standard deviation.
`seed`	scalar; set seed. Default is `NULL`.

Details

CDF(q) = Pr(sqrt(n)*(x_bar-mu)/sigma < q) `dCornishFisher` Computes Cornish-Fisher density from two term Edgeworth expansion given mean, standard deviation, skewness and excess kurtosis. `pCornishFisher` Computes Cornish-Fisher CDF from two term Edgeworth expansion given mean, standard deviation, skewness and excess kurtosis. `qCornishFisher` Computes Cornish-Fisher quantiles from two term Edgeworth expansion given mean, standard deviation, skewness and excess kurtosis. `rCornishFisher` simulates observations based on Cornish-Fisher quantile expansion given mean, standard deviation, skewness and excess kurtosis.

Value

dCornishFisher gives the density, pCornishFisher gives the distribution function, qCornishFisher gives the quantile function, and rCornishFisher generates n random simulations.

Author(s)

Eric Zivot and Yi-An Chen.

References

DasGupta, A. (2008). Asymptotic theory of statistics and probability. Springer. Severini, T. A., (2000). Likelihood Methods in Statistics. Oxford University Press.

Examples

```
## Not run:
# generate 1000 observation from Cornish-Fisher distribution
rc <- rCornishFisher(1000,1,0,5)
hist(rc, breaks=100, freq=FALSE,
     main="simulation of Cornish Fisher Distribution", xlim=c(-10,10))
lines(seq(-10,10,0.1), dnorm(seq(-10,10,0.1), mean=0, sd=1), col=2)
# compare with standard normal curve

# exponential example from A.dasGupta p.188
# x is iid exp(1) distribution, sample size = 5
# then x_bar is Gamma(shape=5, scale=1/5) distribution
q <- c(0,0.4,1,2)
# exact cdf
pgamma(q/sqrt(5)+1, shape=5, scale=1/5)
# use CLT
pnorm(q)
# use edgeworth expansion
pCornishFisher(q, n=5, skew=2, ekurt=6)

## End(Not run)
```

---

fitSfm *Fit a statistical factor model using principal component analysis*

Description

Fits a statistical factor model using Principal Component Analysis (PCA) for one or more asset returns or excess returns. When the number of assets exceeds the number of time periods, Asymptotic Principal Component Analysis (APCA) is performed. An object of class "sfm" is returned. This function is based on the S+FinMetric function mfactor.

Usage

```
fitSfm(data, k = 1, max.k = NULL, refine = TRUE, sig = 0.05,
  check = FALSE, corr = FALSE, ...)

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

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

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

Arguments

data	vector, matrix, data.frame, xts, timeSeries or zoo object with asset returns. See details.
k	number of factors; a number (or) a method for determining the optimal number of factors, one of "bn" or "ck". See details. Default is 1.
max.k	scalar; the maximum number of factors to be considered for methods "bn" or "ck". Default is NULL. See details.
refine	logical; whether to use the Connor-Korajczyk refinement for APCA. Default is TRUE.
sig	scalar; desired level of significance when "ck" method is specified. Default is 0.05.
check	logical; to check if any asset has identical observations. Default is FALSE.
corr	logical; whether to use the correlation instead of the covariance matrix when finding the principal components. Default is FALSE.
...	optional arguments passed to lm.
object	a fit object of class sfm which is returned by fitSfm

Details

If data is not of class "xts", rownames must provide an "xts" compatible time index. Before model fitting, incomplete cases in data are removed using na.omit. Specifying check=TRUE, issues a warning if any asset is found to have identical observations.

Let N be the number of columns or assets and T be the number of rows or observations. When $N < T$, Principal Component Analysis (PCA) is performed. Otherwise, Asymptotic Principal Component Analysis (APCA) is performed. In either case, any number of factors less than min(N,T) can be chosen via argument k. Default is 1. Refer to Zivot and Wang (2007) for more details and references.

Alternately, for APCA, a method to determine the number of factors can be specified: k="bn" corresponds to Bai and Ng (2002) and k="ck" corresponds to Connor and Korajczyk (1993). User can specify the maximum number of factors, max.k to consider with these methods. If not, it is assumed to be either 10 or $T-1$, whichever is smaller.

refine specifies whether a refinement of the APCA procedure from Connor and Korajczyk (1988) that may improve efficiency is to be used.

When corr=TRUE, the correlation matrix of returns are used for finding the principal components instead of the covariance matrix. This is typically decided by practioners on a case-by-case basis. The variable with the highest variance dominates the PCA when the covariance matrix is used. However, this may be justified if a volatile asset is more interesting for some reason and volatility information shouldn't be discarded. On the other hand, using the correlation matrix standardizes the variables and makes them comparable, avoiding penalizing variables with less dispersion.

Finally, if the median of the 1st principal component is negative, all it's factor realizations are automatically inverted to enable more meaningful interpretation.

Value

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

The generic accessor 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 "sfm" is a list containing the following components:

asset.fit	fitted object of class "mlm" or "lm" from the time-series OLS regression of asset returns on estimated factors.
k	number of factors; as input or determined by "ck" or "bn" methods.
factors	T x K xts object of estimated factor realizations.
loadings	N x K matrix of factor loadings estimated by regressing the asset returns on estimated factors.
alpha	length-N vector of estimated alphas.
r2	length-N vector of R-squared values.
resid.sd	length-N vector of residual standard deviations.
residuals	T x N xts object of residuals from the OLS regression.
Omega	N x N return covariance matrix estimated by the factor model.
eigen	length-N (or length-T for APCA) vector of eigenvalues of the sample covariance matrix.
mimic	N x K matrix of factor mimicking portfolio weights.
call	the matched function call.
data	T x N xts data object containing the asset returns.
asset.names	length-N vector of column names from data.

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

Author(s)

Eric Zivot, Sangeetha Srinivasan and Yi-An Chen

References

Bai, J., & Ng, S. (2002). Determining the number of factors in approximate factor models. Econometrica, 70(1), 191-221.

Connor, G., & Korajczyk, R. A. (1988). Risk and return in an equilibrium APT: Application of a new test methodology. Journal of Financial Economics, 21(2), 255-289.

Connor, G., & Korajczyk, R. A. (1993). A test for the number of factors in an approximate factor model. The Journal of Finance, 48(4), 1263-1291.

Zivot, E., & Wang, J. (2007). Modeling Financial Time Series with S-PLUS (Vol. 191). Springer.

See Also

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

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

paFm for Performance Attribution.

Examples

```
# load return data
data(StockReturns)

# PCA is performed on r.M and APCA on r.W
class(r.M)
dim(r.M)
range(rownames(r.M))
class(r.W)
dim(r.W)

# PCA
args(fitSfm)
fit.pca <- fitSfm(r.M, k=2)
class(fit.pca)
names(fit.pca)
head(fit.pca$factors)
head(fit.pca$loadings)
fit.pca$r2
fit.pca$resid.sd
fit.pca$mimic

# APCA with number of factors, k=15
fit.apca <- fitSfm(r.W, k=15, refine=TRUE)

# APCA with the Bai & Ng method
fit.apca.bn <- fitSfm(r.W, k="bn")

# APCA with the Connor-Korajczyk method
fit.apca.ck <- fitSfm(r.W, k="ck")
```

---

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-OLS, 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("OLS", "DLS", "Robust"),
  variable.selection = c("none", "stepwise", "subsets", "lars"),
  mkt.timing = NULL, 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 containing names of assets, whose returns or excess returns are the dependent variable.
factor.names	vector containing names of the macroeconomic factors.
mkt.name	name of the column for market excess returns (Rm-Rf); this is necessary to add market timing factors. Default is NULL.
rf.name	name of the column of risk free rate variable to calculate excess returns for all assets (in asset.names) and factors (in factor.names). 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 "OLS", "DLS" or "Robust". See details. Default is "OLS".
variable.selection	
	the variable selection method, one of "none", "stepwise","subsets","lars". See details. Default is "none".
mkt.timing	one of "HM" or "TM". See Details. Default is NULL. mkt.name is required if any of these options are to be implemented.
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
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 "OLS" 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 lmRob).

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.lmRob), 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.

Arguments mkt.name and mkt.timing allow for market-timing factors to be added to any of the above methods. A wrapper function fitTsfmMT is built based on this functionality. Please refer to fitTsfmMT for detail.

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 accessor 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="OLS"` or `"DLS"`, class `lmRob` if the `fit.method="Robust"`, or class `lars` if `variable.selection="lars"`.
alpha	length-N vector of estimated alphas.
beta	N x K matrix 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 assets and factors.
asset.names	asset.names as input.
factor.names	factor.names 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.

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 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 from the database
data(managers)
fit <- fitTsfm(asset.names=colnames(managers[,(1:6)]),
               factor.names=colnames(managers[,(7:9)]), data=managers)
summary(fit)
fitted(fit)
# plot actual returns vs. fitted factor model returns for HAM1
plot(fit, plot.single=TRUE, asset.name="HAM1", which.plot.single=1,
     loop=FALSE)
# group plot; type selected from menu prompt; auto-looped for multiple plots
# plot(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, scope, scale, direction,
  trace = FALSE, 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,
  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="OLS","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="OLS". 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.
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.lmRob 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".
trace	If positive (or, not FALSE), info is printed during the running of step, step.lmRob, lars or cv.lars as relevant. Larger values may give more detailed information. Default is FALSE.
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.
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.

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, lmRob, 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 "OLS" 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, lmRob, step, regsubsets, lars and cv.lars

Examples

```
## Not run:
# check argument list passed by fitTsfm.control
tsfm.ctrl <- fitTsfm.control(method="exhaustive", nvmin=2)
print(tsfm.ctrl)
```

```
## End(Not run)

# used internally by fitTsfm in the example below
data(managers)
fit <- fitTsfm(asset.names=colnames(managers[,(1:6)]),
               factor.names=colnames(managers[,(7:9)]),
               data=managers, variable.selection="subsets",
               method="exhaustive", nvmin=2)
```

---

fitTsfmLagBeta *Fit a lagged 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-OLS, discounted least squares-DLS (or) robust regression like fitTsfm.An object of class "tsfm" is returned.

Usage

```
fitTsfmLagBeta(asset.names, mkt.name, rf.name = NULL, data = data,
  fit.method = c("OLS", "DLS", "Robust"), LagBeta = 1,
  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 "OLS", "DLS" or "Robust". See details. Default is "OLS".
LagBeta	A integer number to specify numbers of lags of Betas to include in the model. The Default is 1.
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 returns model estimates lagged market Beta. Specifically,

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

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 Betas:

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

Value

fitTsfmLagBeta 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="OLS" 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 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

```
# load data from the database
data(managers)

# example: A lagged Beetas model with OLS fit
fit <- fitTsfmLagBeta(asset.names=colnames(managers[,(1:6)]),LagBeta=2,
                      mkt.name="SP500.TR",rf.name="US.3m.TR",data=managers)
summary(fit)
fitted(fit)
```

`fitTsfmMT`   *Fit a time serie market timing factor model using time series regression*

Description

This is a wrapper function to fits a time series market timing factor model for one or more asset returns or excess returns using time series regression. Users can choose between ordinary least squares-OLS, 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("OLS", "DLS", "Robust"), 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 market timing 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 "OLS", "DLS" or "Robust". See details. Default is "OLS".
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 Market timing accounts for the price movement of the general stock market relative to fixed income securities. It includes down.market = max(0, R_f-R_m) as a factor, following Henriksson & Merton (1981). 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 kills.

Value

`fitTsfmMT` 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="OLS" 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 assets and factors.
asset.names	asset.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

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.

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

See Also

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

Examples

```
# load data from the database
data(managers)

# example: Market-timing factors with OLS fit
fit <- fitTsfmMT(asset.names=colnames(managers[,(1:6)]),
                 mkt.name="SP500.TR",rf.name="US.3m.TR",data=managers)
summary(fit)
fitted(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-OLS, 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("OLS", "DLS", "Robust"), 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 up/down market 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 "OLS", "DLS" or "Robust". See details. Default is "OLS".
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.

Each object of tsfm contains :

asset.fit	list of fitted objects for each asset. Each object is of class lm if fit.method="OLS" 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 from the database
data(managers)

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

 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, use = "pairwise.complete.obs", ...)

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

Arguments

object	fit object of class `tsfm`, `sfm` or `ffm`.
...	optional arguments passed to `cov`.
use	an optional character string giving a method for computing covariances in the presence of missing values. This must be (an abbreviation of) one of the strings "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`, `fitSfm`, `fitFfm`

`cov` for more details on arguments `use` and `method`.

Examples

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

# Statistical Factor Model
data(StockReturns)
sfm.pca.fit <- fitSfm(r.M, k=2)
fmCov(sfm.pca.fit)

## Not run:
# Fundamental Factor Model
data(stock)
# there are 447 assets
exposure.names <- c("BOOK2MARKET", "LOG.MARKETCAP")
beta.mat <- subset(stock, DATE=="2003-12-31")[, exposure.names]
beta.mat1 <- cbind(rep(1, 447), beta.mat1)
# FM return covariance
fit.fund <- fitFfm(exposure.names=c("BOOK2MARKET", "LOG.MARKETCAP"),
                   data=stock, returnsvar="RETURN", datevar="DATE",
                   assetvar="TICKER", wls=TRUE, regression="classic",
                   covariance="classic", full.resid.cov=FALSE)
ret.cov.fundm <- fmCov(beta.mat1,fit.fund$factor.cov$cov,fit.fund$resid.sd)
fit.fund$returns.cov$cov == ret.cov.fundm

## End(Not run)
```

---

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). VaR is computed as the sample quantile of the historic or simulated data.

Usage

```
fmEsDecomp(object, ...)

## S3 method for class 'tsfm'
fmEsDecomp(object, p = 0.95, method = c("modified",
  "gaussian", "historical", "kernel"), invert = FALSE, ...)

## S3 method for class 'sfm'
fmEsDecomp(object, p = 0.95, method = c("modified",
  "gaussian", "historical", "kernel"), invert = FALSE, ...)
```

Arguments

object	fit object of class `tsfm`, `sfm` or `ffm`.
...	other optional arguments passed to [VaR]().
p	confidence level for calculation. Default is 0.95.
method	method for computing VaR, one of "modified","gaussian", "historical", "kernel". Default is "modified". See details.
invert	logical; whether to invert the VaR measure. Default is `FALSE`.

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.

Computation of the VaR measure is done using [VaR](). Arguments p, method and invert are passed to this function. Refer to their help file for details and other options. invert consistently affects the sign for all VaR and ES measures.

Value

A list containing

ES.fm	length-N vector of factor model ES 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.
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, fitSfm, fitFfm for the different factor model fitting functions.

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

Examples

```
# Time Series Factor Model
data(managers)
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

# Statistical Factor Model
data(StockReturns)
sfm.pca.fit <- fitSfm(r.M, k=2)
ES.decomp <- fmEsDecomp(sfm.pca.fit)
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, use = "pairwise.complete.obs", ...)

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

Arguments

object	fit object of class tsfm, sfm or ffm.
...	optional arguments passed to cov.
use	an optional character string giving a method for computing covariances in the presence of missing values. This must be (an abbreviation of) one of the strings "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, Sangeetha Srinivasan and Yi-An Chen

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`, `fitSfm`, `fitFfm` 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
data(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

# Statistical Factor Model
data(StockReturns)
sfm.pca.fit <- fitSfm(r.M, k=2)
decomp <- fmSdDecomp(sfm.pca.fit)
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 wrt factor beta is computed as the expected factor return given fund return is equal to its VaR and approximated by a kernel estimator. VaR is computed either as the sample quantile or as an estimated quantile using the Cornish-Fisher expansion.

Usage

```
fmVaRDecomp(object, ...)

## S3 method for class 'tsfm'
fmVaRDecomp(object, p = 0.95, method = c("modified",
  "gaussian", "historical", "kernel"), invert = FALSE, ...)

## S3 method for class 'sfm'
fmVaRDecomp(object, p = 0.95, method = c("modified",
  "gaussian", "historical", "kernel"), invert = FALSE, ...)
```

Arguments

object   fit object of class tsfm, sfm or ffm.

...   other optional arguments passed to [VaR](VaR).

p   confidence level for calculation. Default is 0.95.

method   method for computing VaR, one of "modified","gaussian", "historical", "kernel". Default is "modified". See details.

invert   logical; whether to invert the VaR measure. Default is FALSE.

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.

Computation of the risk measure is done using `VaR`. Arguments `p`, `method` and `invert` are passed to this function. Refer to their help file for details and other options.

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, Sangeetha Srinivasan and Yi-An Chen

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`, `fitSfm`, `fitFfm` for the different factor model fitting functions.

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

Examples

```
# Time Series Factor Model
data(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

# Statistical Factor Model
data(StockReturns)
sfm.pca.fit <- fitSfm(r.M, k=2)
VaR.decomp <- fmVaRDecomp(sfm.pca.fit)
VaR.decomp$cVaR
```

---

managers   *Hypothetical Alternative Asset Manager and Benchmark Data*

Description

This dataset and it's documentation have been duplicated from managers in the PerformanceAnalytics package. managers is used in the examples and vignette of the factorAnalytics package.

A xts object that contains columns of monthly returns for six hypothetical asset managers (HAM1 through HAM6), the EDHEC Long-Short Equity hedge fund index, the S\&P 500 total returns, and total return series for the US Treasury 10-year bond and 3-month bill. Monthly returns for all series end in December 2006 and begin at different periods starting from January 1996.

Note that all the EDHEC indices are available in edhec.

Usage

```
managers
```

Format

CSV conformed into an xts object with monthly observations

Details

Please note that the 'managers' data set included with PerformanceAnalytics will be periodically updated with new managers and information. If you intend to use this data set in automated tests, please be sure to subset your data like managers[1:120,1:6] to use the first ten years of observations on HAM1-HAM6.

Examples

```
data(managers)

#preview the data
head(managers)

#summary period statistics
```

```
summary(managers)

#cumulative returns
tail(cumprod(1+managers),1)
```

---

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](), [fitSfm](), [fitFfm]() for the factor model fitting functions.

The pafm methods for generic functions: [plot.pafm](), [print.pafm]() and [summary.pafm]().

Examples

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

Author(s)

Yi-An Chen.

Examples

```
## Not run:
data(managers)
fit <- fitTsfm(asset.names=colnames(managers[,(1:6)]),
               factor.names=c("EDHEC LS EQ","SP500 TR"), data=managers)
fm.attr <- paFm(fit)
# plot all
plot(fm.attr, legend.loc="topleft", max.show=6)
dev.off()
# plot only one assets "HAM1
plot(fm.attr, plot.single=TRUE, fundName="HAM1")

## End(Not run)
```

---

plot.sfm   *Plots from a fitted statistical factor model*

Description

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

Usage

```
## S3 method for class 'sfm'
plot(x, which.plot.group = NULL, factor.subset = 1:4,
  asset.subset = 1:5, n.top = 3, plot.single = FALSE, asset.name,
  which.plot.single = NULL, colorset = (1:12), legend.loc = "topleft",
  las = 1, VaR.method = "historical", cum.var = TRUE, eig.max = 0.9,
  loop = TRUE, ...)
```

Arguments

x	an object of class `sfm` produced by `fitSfm`.
which.plot.group	a number to indicate the type of group plot for multiple assets. If `NULL` (default), the following menu appears: 1 = Screeplot of eigenvalues, 2 = Time series plot of estimated factors, 3 = Estimated factor loadings, 4 = Histogram of R-squared, 5 = Histogram of residual volatility, 6 = Factor model residual correlation 7 = Factor model correlation, 8 = Factor contribution to SD, 9 = Factor contribution to ES, 10 = Factor contribution to VaR, 11 = Factor mimicking portfolio weights - top long and short positions in each factor, 12 = Asset correlations - top long and short positions in each factor
factor.subset	vector of names/indices of factors to show for group plots. Default is 1:4.
asset.subset	vector of names/indices of assets to show for group plots. Default is 1:5.

n.top   scalar; number of largest and smallest weights to display for each factor mim-
                 icking portfolio. Default is 3.

plot.single   logical; If TRUE plots the characteristics of an individual asset's factor model.
                 The type of plot is given by which.plot.single. 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.

which.plot.single
                 a number to indicate the type of group plot for an individual asset. If NULL
                 (default), the following menu appears:
                 1 = Time series plot of actual and fitted asset returns,
                 2 = Time series plot of residuals with standard error bands,
                 3 = Time series plot of squared residuals,
                 4 = Time series plot of absolute residuals,
                 5 = SACF and PACF of residuals,
                 6 = SACF and PACF of squared residuals,
                 7 = SACF and PACF of absolute residuals,
                 8 = Histogram of residuals with normal curve overlayed,
                 9 = Normal qq-plot of residuals,
                 10 = CUSUM test-Recursive residuals,
                 11 = CUSUM test-OLS residuals,
                 12 = Recursive estimates (RE) test of OLS regression coefficients,
                 13 = Rolling estimates over a 24-period observation window

colorset   color palette to use for all the plots. Default is c(1:12). The 1st element will
                 be used for individual time series plots or the 1st series 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 here
                 is 1.

VaR.method   a method for computing VaR; one of "modified", "gaussian", "historical" or "ker-
                 nel". VaR is computed using [VaR](#). Default is "historical".

cum.var   logical; If TRUE, the cumulative fraction of the variance is printed above each
                 bar in the screeplot of eigenvalues. Default is TRUE.

eig.max   scalar in (0,1] for limiting the screeplot to factors that explain a given percent of
                 the variance. Default is 0.9.

loop   logical to indicate if the plot menu should be repeated. Default is TRUE.

...   further arguments to be passed to other plotting functions.

Details

If the plot type argument is not specified, a menu prompts for user input and the corresponding
plot is output. And, the menu is repeated for user convenience in plotting multiple characteristics.
Selecting '0' from the menu exits the current plot.sfm call. Alternately, setting loop=FALSE will
exit after plotting any one chosen characteristic without the need for menu selection.

Group plots are the default. The variables in asset.subset and factors in asset.subset are
plotted depending on the characteristic chosen. The default is to show the first 4 factors and first 5
assets.

Individual asset plots are selected by specifying plot.single=TRUE. In which case, asset.name is necessary if multiple assets were modeled in x. However, if the fitSfm object contains only one asset's factor model fit, plot.sfm can infer this automatically, without user input.

Author(s)

Eric Zivot, Sangeetha Srinivasan and Yi-An Chen

See Also

fitSfm and summary.sfm for details about the time series factor model fit, extractor functions and summary statistics.

efp for CUSUM tests.

plot.xts, chart.TimeSeries, chart.ACFplus, chart.Histogram, chart.QQPlot, barplot, barchart and corrplot for plotting methods used.

fmSdDecomp, fmEsDecomp, fmVaRDecomp for factor model risk measures.

Examples

```
# load data from the database
data(StockReturns)

# APCA with number of factors, k=15
fit.apca <- fitSfm(r.W, k=15, refine=TRUE)

# group plot(default); select type from menu prompt
# menu is auto-looped to get multiple types of plots based on the same fit
# plot(fit.apca)

# plot the factor betas of 1st 4 assets fitted above
# loop disabled to get one type of plot without interative menu
plot(fit.apca, asset.subset=1:4, which.plot.group=3, loop=FALSE)

# plot factor model return correlation; angular order of the eigenvectors
plot(fit.apca, which.plot.group=7, loop=FALSE,
     order="AOE", method="ellipse", tl.pos = "d")

# histogram of residuals from an individual asset's factor model fit
plot(fit.apca, plot.single=TRUE, asset.name="AFL", which.plot.single=8,
     loop=FALSE)
```

---

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.plot.group = NULL, max.show = 6,
  plot.single = FALSE, asset.name, which.plot.single = NULL,
  colorset = (1:12), legend.loc = "topleft", las = 1,
  VaR.method = "historical", loop = TRUE, ...)
```

Arguments

x   an object of class `tsfm` produced by `fitTsfm`.

which.plot.group
                      a number to indicate the type of group plot for multiple assets. If `NULL` (default),
                      the following menu appears:
                      1 = Factor model coefficients: Alpha,
                      2 = Factor model coefficients: Betas,
                      3 = Actual and Fitted asset returns,
                      4 = R-squared,
                      5 = Residual Volatility,
                      6 = Factor Model Residual Correlation
                      7 = Factor Model Correlation,
                      8 = Factor Contribution to SD,
                      9 = Factor Contribution to ES,
                      10 = Factor Contribution to VaR

max.show   maximum number of assets in a given plot. Default is 6.

plot.single   logical; If `TRUE` plots the characteristics of an individual asset's factor model.
                      The type of plot is given by `which.plot.single`. 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`.

which.plot.single
                      a number to indicate the type of group plot for an individual asset. If `NULL`
                      (default), the following menu appears:
                      1 = Time series plot of actual and fitted asset returns,
                      2 = Time series plot of residuals with standard error bands,
                      3 = Time series plot of squared residuals,
                      4 = Time series plot of absolute residuals,
                      5 = SACF and PACF of residuals,
                      6 = SACF and PACF of squared residuals,
                      7 = SACF and PACF of absolute residuals,
                      8 = Histogram of residuals with normal curve overlayed,
                      9 = Normal qq-plot of residuals,
                      10 = CUSUM test-Recursive residuals,
                      11 = CUSUM test-OLS residuals,
                      12 = Recursive estimates (RE) test of OLS regression coefficients,
                      13 = Rolling estimates over a 24-period observation window

colorset   color palette to use for all the plots. Default is `c(1:12)`. The 1st element will
                      be used for individual time series plots or the 1st series 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 here is 1.
VaR.method	a method for computing VaR; one of "modified", "gaussian", "historical" or "kernel". VaR is computed using [VaR](#). Default is "historical".
loop	logical to indicate if the plot menu should be repeated. Default is TRUE.
...	further arguments to be passed to other plotting functions.

Details

If the plot type argument is not specified, a menu prompts for user input and the corresponding plot is output. And, the menu is repeated for user convenience in plotting multiple characteristics. Selecting '0' from the menu exits the current `plot.tsfm` call. Alternately, setting `loop=FALSE` will exit after plotting any one chosen characteristic without the need for menu selection.

For group plots (the default), the first `max.show` assets are plotted. For individual plots, `asset.name` is necessary if multiple assets were modeled in x and `plot.single=TRUE`. However, if the `fitTsfm` object contains only one asset's factor model fit, `plot.tsfm` can infer this automatically, without user input.

CUSUM plots (individual asset plot options 10, 11 and 12) are applicable only for `fit.method="OLS"`.

Rolling estimates (individual asset plot option 13) is not applicable for `variable.slection="lars"`.

Author(s)

Eric Zivot, Sangeetha Srinivasan and Yi-An Chen

See Also

[fitTsfm](#) and [summary.tsfm](#) for details about the time series factor model fit, extractor functions and summary statistics.

[efp](#) for CUSUM tests.

[plot.xts](#), [chart.TimeSeries](#), [chart.ACFplus](#), [chart.Histogram](#), [chart.QQPlot](#), [barplot](#), [barchart](#) and [corrplot](#) for plotting methods used.

[fmSdDecomp](#), [fmEsDecomp](#), [fmVaRDecomp](#) for factor model risk measures.

Examples

```
# load data from the database
data(managers)
fit.macro <- fitTsfm(asset.names=colnames(managers[,(1:6)]),
                     factor.names=colnames(managers[,(7:8)]),
                     rf.name="US.3m.TR", data=managers)

# group plot(default); select type from menu prompt
# menu is auto-looped to get multiple types of plots based on the same fit
# plot(fit.macro)

# plot the factor betas of 1st 4 assets fitted above
# loop disabled to get one type of plot without interative menu
plot(fit.macro, max.show=4, which.plot.group=2, loop=FALSE)

# plot factor model return correlation; angular order of the eigenvectors
plot(fit.macro, which.plot.group=7, loop=FALSE,
     order="AOE", method="ellipse", tl.pos = "d")
```

```
# histogram of residuals from an individual asset's factor model fit
plot(fit.macro, plot.single=TRUE, asset.name="HAM1", which.plot.single=8,
     loop=FALSE)
```

---

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(object, mkt.name, assets.name, ...)
```

Arguments

object	an object of class `tsfmUpDn` produced by `fitTsfmUpDn`.
mkt.name	The name of market returns used in `fitTsfmUpDn`. It is necessary to provide the name of market returns.
assets.name	The name of the asset used in `fitTsfmUpDn`. It supports one asset name at once.
...	Other arguments can be used in `plot`.

Details

This method plots actual values against fitted value of up and down market time series factor model. The black dots are actual values and the red lines are fitted values.

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

Author(s)

Yi-An Chen

See Also

[fitTsfmUpDn](#)

Examples

```
# load data from the database
data(managers)
# example: Up and down market factor model with OLS fit
fitUpDn <- fitTsfmUpDn(asset.names=colnames(managers[,(1:6)]),mkt.name="SP500.TR",
                       data=managers, fit.method="OLS",control=NULL)
 # plot the fitted model of the first asset
 plot(fitUpDn,mkt.name="SP500.TR",assets.name="HAM1")
```

---

predict.sfm *Predicts asset returns based on a fitted statistical factor model*

Description

S3 `predict` method for object of class `sfm`. It calls the `predict` method for fitted objects of class `lm`.

Usage

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

Arguments

`object`	an object of class `sfm` produced by `fitSfm`.
`newdata`	a vector, matrix, data.frame, xts, timeSeries or zoo object containing the variables with which to predict.
`...`	optional arguments passed to `predict.lm`.

Value

`predict.sfm` produces a vector or a matrix of predictions.

Author(s)

Yi-An Chen and Sangeetha Srinivasan

See Also

[fitSfm](#), [summary.sfm](#)

Examples

```
# load data from the database
data(StockReturns)
# fit the factor model with PCA
fit <- fitSfm(r.M, k=2)

pred.fit <- predict(fit)
newdata <- data.frame("CITCRP"=rnorm(n=120), "CONED"=rnorm(n=120))
rownames(newdata) <- rownames(fit$data)
pred.fit2 <- predict(fit, newdata, interval="confidence")
```

---

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 vector or a matrix of predictions.

Author(s)

Yi-An Chen and Sangeetha Srinivasan

See Also

`fitTsfm`, `summary.tsfm`

Examples

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

pred.fit <- predict(fit)
newdata <- data.frame("EDHEC.LS.EQ"=rnorm(n=120), "SP500.TR"=rnorm(n=120))
rownames(newdata) <- rownames(fit$data)
pred.fit2 <- 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 from the database
data(managers)
# fit the factor model with OLS
fitUpDn <- fitTsfmUpDn(asset.names=colnames(managers[,(1:6)]),mkt.name="SP500.TR",
                       data=managers, fit.method="OLS",control=NULL)

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.

Author(s)

Yi-An Chen.

Examples

```
# load data from the database
 data(managers)
# fit the factor model with OLS
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.sfm   *Prints out a fitted statictical factor model object*

Description

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

Usage

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

Arguments

x   an object of class sfm produced by fitSfm.

digits   an integer value, to indicate the required number of significant digits. Default is 3.

...   optional arguments passed to the print method.

Author(s)

Yi-An Chen and Sangeetha Srinivasan

See Also

fitSfm, summary.sfm

Examples

```
data(StockReturns)
fit <- fitSfm(r.M, k=2)
print(fit)
```

---

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

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.

Author(s)

Yi-An Chen and Sangeetha Srinivasan

See Also

fitTsfm, summary.tsfm

Examples

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

Author(s)

Yi-An Chen and Sangeetha Srinivasan

See Also

[fitTsfmUpDn](#), [summary.tsfmUpDn](#)

Examples

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

print(fitUpDn)
```

Stock.df   *constructed NYSE 447 assets from 1996-01-01 through 2003-12-31.*

Description

constructed NYSE 447 assets from 1996-01-01 through 2003-12-31.

Details

Continuous data: PRICE, RETURN, TICKER, VOLUME, SHARES.OUT, MARKET.EQUITY,LTDEBT, NET.SALES, COMMON.EQUITY, NET.INCOME, STOCKHOLDERS.EQUITY, LOG.MARKETCAP, LOG.PRICE, BOOK2MARKET Categorical data: GICS, GICS.INDUSTRY, GICS.SECTOR

References

Guy Yullen and Yi-An Chen

StockReturns   *Stock Return Data*

Description

`r.M`: A "data.frame" object with monthly returns (ranging from January 1978 to December 1987) for 15 assets whose names are given in the 'Details'.

`r.W`: A "data.frame" object with weekly returns (ranging from January 8, 1997 to June 28, 2000) for 1618 U.S. stocks.

Usage

```
data(StockReturns)
```

Format

data.frame object

`r.M` monthly from Jan-1998 through Dec-1987

`r.W` weekly from Jan-08-1997 through Jun-28-2000

Details

The 15 assets in `r.M` are as follows: CITCRP monthly returns of Citicorp. CONED monthly returns of Consolidated Edison. CONTIL monthly returns of Continental Illinois. DATGEN monthly returns of Data General. DEC monthly returns of Digital Equipment Company. DELTA monthly returns of Delta Airlines. GENMIL monthly returns of General Mills. GERBER monthly returns of Gerber. IBM monthly returns of International Business Machines. MARKET a value-weighted composite monthly returns based on transactions from the New York Stock Exchange and the American Exchange. MOBIL monthly returns of Mobile. PANAM monthly returns of Pan American Airways. PSNH monthly returns of Public Service of New Hampshire. TANDY monthly returns of Tandy. TEXACO monthly returns of Texaco. WEYER monthly returns of Weyerhauser. RKFREE monthly returns on 30-day U.S. Treasury bills.

Source

S+FinMetrics Berndt.dat & folio.dat

References

Berndt, E. R. (1991). The practice of econometrics: classic and contemporary. Reading, MA: Addison-Wesley.

Examples

```
data(StockReturns)
dim(r.M)
range(rownames(r.M))
dim(r.W)
range(rownames(r.W))
```

---

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.

Author(s)

Yi-An Chen.

Examples

```
# load data from the database
data(managers)
# fit the factor model with OLS
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.sfm   *Summarizing a fitted time series factor model*

Description

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

Usage

```
## S3 method for class 'sfm'
summary(object, se.type = c("Default", "HC", "HAC"),
  n.top = 3, ...)

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

Arguments

object	an object of class sfm returned by fitSfm.
se.type	one of "Default", "HC" or "HAC"; option for computing HC/HAC standard errors and t-statistics. Default is "Default".
n.top	scalar; number of largest and smallest weights to display for each factor mimicking portfolio. Default is 3.
...	futher arguments passed to or from other methods.
x	an object of class summary.sfm.
digits	number of significants digits to use when printing. Default is 3.

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.

Value

Returns an object of class summary.sfm. The print method for class summary.sfm outputs the call, coefficients (with standard errors and t-statistics), r-squared and residual volatilty (under the homoskedasticity assumption) for all assets as well as a summary of the factor mimicking portfolio weights.

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

call	the function call to fitSfm
se.type	standard error type as input
sum.list	list of summaries for the N fit objects of class lm for each asset in the factor model.
mimic.sum	list of data.frame objects containing n.top largest and smallest weights for each factor mimicking portfolio.

Author(s)

Sangeetha Srinivasan

See Also

fitSfm, summary.lm

Examples

```
data(StockReturns)
# fit the factor model with PCA
fit <- fitSfm(r.M, k=2)

# summary of factor model fit for all assets
summary(fit, "HAC")
```

---

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, ...)
```

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".
...	futher arguments passed to or from other methods.
x	an object of class `summary.tsfm`.
digits	number of significants digits to use when printing. Default is 3.

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 = "OLS"` 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

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

# summary of factor model fit for all assets
summary(fit, "HAC")

# summary of lm fit for a single asset
summary(fit$asset.fit[[1]])
```

---

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(uD.list, digits = 3, ...)
```

Arguments

object	an object of class tsfmUpDn returned by fitTsfmUpDn.
...	futher arguments passed to or from summary.tsfm methods.
uD.list	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]()

Examples

```
# load data from the database
data(managers)

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

 summary(fitUpDn)
```

TreasuryYields	*Treasury yields at different maturities*

Description

The following is adapted from chapter 17 of Ruppert (2010).

The data object contains yields on Treasury bonds at 11 maturities, T = 1, 3, and 6 months and 1, 2, 3, 5, 7, 10, 20, and 30 years. Daily yields were taken from a U.S. Treasury website for the time period January 2, 1990, to October 31, 2008.

Daily yields were missing from some values of T because, for example to quote the website, "Treasury discontinued the 20-year constant maturity series at the end of calendar year 1986 and reinstated that series on October 1, 1993." Dif- ferencing may cause a few additional days to have missing values.

Usage

```
data(TreasuryYields)
```

Format

xts time series object

`tr.yields` Jan-02-1990 through Oct-31-2008

Source

SDAFE author's website: <http://people.orie.cornell.edu/davidr/SDAFE/index.html>

References

Ruppert, D. (2010). Statistics and data analysis for financial engineering. Springer.

Examples

```
data(TreasuryYields)
# preview the data
head(tr.yields)
```

Index