Package 'factorAnalytics'

April 9, 2018

Type Package

Title Factor Analytics

Version 2.1.04

Date 2018-04-09

Author Eric Zivot, Doug Martin, Sangeetha Srinivasan, Yi-An Chen, Lingjie Yi, Avinash Acharya and Chindhanai Uthaisaad

Maintainer Sangeetha Srinivasan <sangee@uw.edu>

Description Linear factor model fitting for asset returns (three major types- time series, fundamental and statistical factor models); related risk (volatility, VaR and ES) and performance attribution (factor-contributed vs idiosyncratic returns); tabular displays of risk and performance reports; factor model Monte Carlo, single and multiple imputation methods for simulating returns and backfilling unequal histories.

License GPL-2

Depends R (>= 3.0.0),
foreach (>= 1.4),
xts (>= 0.9),
rrcov (>= 1.3)

Imports PerformanceAnalytics(>= 1.4),
zoo,
corrplot,
robustbase,
robust,
leaps,
lars,
strucchange,
lmtest,
sandwich,
lattice,
MASS,
sn,
boot,
parallel,
doSNOW,
RCurl,
bestglm,
tseries,

HH,
reshape2,
rugarch

Remotes bitbucket::alexiosg/rugarch

Suggests testthat

LazyLoad yes

LazyDataCompression xz

URL http://r-forge.r-project.org/projects/returnanalytics/

RoxygenNote 6.0.1

R topics documented:

`assetDecomp` *Decompose portfolio risk into individual asset contributions and provide tabular report*

Description

Compute the asset contributions to Sd, VaR and ES of returns based on Euler's theorem

Usage

```
assetDecomp(object, weights = NULL, rm, p, type = c("np", "normal"))
```

Arguments

object	fit object of class tsfm, or ffm.
weights	vector of weights of the assets in the portfolio. Default is NULL, in which case an equal weights will be used.
rm	one of "Sd" (Standard Deviation) or "VaR" (Value at Risk) or "ES" (Expected Shortfall)
p	tail probability for calculation. Default is 0.05.
type	one of "np" (non-parametric) or "normal". Default is "np".
...	other optional arguments

Value

Risk Decomposition report for every asset in the portfolio

Author(s)

Avinash Acharya

References

Epperlein and Smillie (2006) "Cracking VAR with Kernels" Risk.net

See Also

riskDecomp for the Risk Decomposition function based on factors in the fitted model.

Examples

```
# Fundamental Factor Model
data("factorDataSetDjia5Yrs")
data("wtsDjiaGmvLo")

# fit a fundamental factor model
exposure.vars <- c("P2B", "MKTCAP")
fit <- fitFfm(data=factorDataSetDjia5Yrs, asset.var="TICKER", ret.var="RETURN",
               date.var="DATE", exposure.vars=exposure.vars, z.score = "crossSection")

#non-parametric
weights = wtsDjiaGmvLo
sd.decomp = assetDecomp(fit, weights, rm = "Sd")
VaR.decomp = assetDecomp(fit, weights, rm = "VaR", p = 0.05)
ES.decomp = assetDecomp(fit, weights, rm = "ES", p = 0.05)

#normal dist
VaR.decomp = assetDecomp(fit, weights, rm = "VaR", p = 0.05, type = "normal")
ES.decomp = assetDecomp(fit, weights, rm = "ES", p = 0.05, type = "normal")
```

chartCusum *cusumActMgr Plots*

Description

Plot the elements of an cusumActMgr object. The EWMA tracking errors are computed month-to-month by the simple weighted average of the "log-excess returns" volatility and its former (adjacent) time period volatility. The excess volatility is the EWMA tracking error difference from the portfolio returns EWMA tracking error and the benchmark returns EWMA tracking error. Annualized IR is computed by using the monthly log-excess returns and the annualized EWMA tracking errors.

Usage

```
chartCusum(object, digits = 3, which = NULL, ...)
```

Arguments

object	An object of class cusumActMgr returned by cusumActMgr.
digits	The number of digits of numerical values in graphs
which	A number or a vector of numbers to indicate the type of plots. If a subset of the plots is required, specify a subset of the numbers 1 and 2 for plots. The numbers 1 and 2 represent: 1 = Barplot of log-excess returns with annually moving average returns, plot of tracking error, Plot of excess volatility, and Plot of cusum for returns with protractors of slopes representing the annualized returns, 2 = Barplot of the information ratio, plot of cusumIR with protractors of slopes with corresponding IR, and a plot of Lindley's Recursion with the thresholds, and a scatter plot between the fund and benchmark returns with robust regression
...	other graphics parameters in plot

Value

Graph(s) as specified by the user.

Author(s)

Chindhanai Uthaisaad

Examples

```
data("cusumData")
results = cusumActMgr(portfolioName = "Parvest", benchmarkName = "RUS2500", data = cusumData)
chartCusum(results, which = 1)
chartCusum(results, which = c(1,2))
```

chartRobRisk riskBudget *Plots*

Description

Plot the elements of an riskBudget object.

Usage

```
chartRobRisk(object, digits = 3, col = c(4, 2), ...)
```

Arguments

object	An object of class riskBudget returned by robRiskBudget.
digits	The number of digits of numerical values in graphs
col	The vector of numerics representing the color of the initial and final risk budgetings, respectively.
...	other graphics parameters in plot

Value

The graph of initial risk budgets and final risk budgets

Author(s)

Chindhanai Uthaisaad

Examples

```
data("RussellData")
rf = RussellData[, 16]
robRiskData = RussellData[, 1:15]

riskBudget = robRiskBudget(robRiskData, rf = rf, shrink = TRUE, avgCor = TRUE,
ESMethod = "historical", corMatMethod = "mcd")
chartRobRisk(riskBudget)
```

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/

---

cusumActMgr *Using Statistical Process Control to Monitor Active Management*

Description

Monitor the risk adjusted performance (Information Ratio) of an actively managed portfolio and raise an alarm when sufficient evidence has accrued to indicate that its current Information Ratio is 0 or worse. The monitorng is performed using an optimal changepoint detection scheme (the CUSUM algorithm) An object of class `cusumActMgr` is returned.

Usage

```
cusumActMgr(portfolioName, benchmarkName, data, upperIR = 0.5, lowerIR = 0,
  lambda_in = 0.1, lambda_out = 0.2, huberize = 4, filterStd = FALSE)
```

Arguments

`portfolioName`	a character representing the name of the fund. It is a required argument with no default
`benchmarkName`	a character representing the name of the benchamark It is a required argument with no default
`data`	an xts object containing the columns `portfolioName` and `benchmarkName` of monthly returns. This argument is required with no default.
`upperIR`	a numeric value representing the information ratio of a good performance. The default is set to 0.5

lowerIR	a numeric value representing the information ratio of a bad performance. The default is set to 0
lambda_in	the exponential weighting constant when the data seems consistent with the current estimate of volatility. The default is set to 0.1
lambda_out	the exponential weighting constant when the data seems inconsistent with the current estimate of volatility. The default is set to 0.2
huberize	the numeric value, greater than 1, of standard deviations at which we huberize. The default is set to 4.
filterStd	the logical value representing the filter of the estimated standard deviations. The default is set to FALSE.

Details

Assessing the performance of the active managers is hard because active returns (i.e. portfolio return - benchmark return) are noisy. In addition, the risk of these active returns must be taken account of, and this is commonly (though not universally) done by measuring the standard deviation of active returns. Empirical studies of active managers across a wide range of asset classes suggest that an Annualized Information Ratio (IR = Active Return / Std. Dev.(Active Return)) of 0.5 over a period of 5 years or more is exceptional. In addition, public markets are very efficient, and known inefficiencies disappear as they get arbitraged away, though other inefficiencies sometimes appear in their place. Consequently, the majority of active managers deliver active returns and IR close to 0, and even those with a positive IR are at constant risk of having their added value dissipate. Investors, therefore, must continually estimate the current performance of their active portfolios and determine when sufficient evidence has accrued to suggest that their active return and IR have fallen to 0 (or turned negative). Put differently, investors need to reapidly detect changes, particularly negative changes, in the performance of their portfolios.

There is a rich literature on changepoint detection, and of the many available algorithms to detect changepoints, the CUSUM (an acronym for CUmulative SUM) stands out on account of its simplicity, its robustness to the actual distribution of active returns, and the optimal trade-off between detection time and the rate of false alarms that it offers. It is closely retlated to Wald's Sequential Probability Ratio Test (SPRT) but is much simpler to implement, and requires minimal inputs from the user. In this application, it seeks to determine when the IR of a portfolio has changed from a good level (default = 0.5) to a bad level (default = 0). An alarm is raised when the CUSUM scheme crosses a threshold, which is chosen to make the average time between false alarms 84 months (7 years). By way of comparison, the time taken to detect a transition from good performance to bad is 41 months (or 3 1/2 years). This is much faster than the traditional t-test, which would take 16 years to obtain a t-statistic of 2. The threshold can be recalibrated to meet a user's needs.

Value

cusumActMgr returns a list containing the following xts objects:

Logarithmic_Excess_Returns	Logarithmic excess returns of the fund relative to the benchmark
Annual_Moving_Average	The vector of annual moving average returns
Tracking_Error	The monthly tracking error of the logarithmic excess returns
Information_Ratios	The vector of monthly information ratios
Lindley's_Recursion	The vector Lindley's recursion with a reset after the detection threshold (6.81) is passed.

`Annualized_Cusum_IR`
: The vector annualized CUSUM of the information ratios

`Annualized Cusum Excess Return`
: The vector annualized CUSUM of the excess returns

`Means`
: The xts matrix of estimated means of the fund in the first columns, the benchmark in the second column, and the excess logarithmic returns in the third column

`Standard_deviations`
: The xts matrix of estimated standard deviations of the fund in the first columns, the benchmark in the second column, and the excess logarithmic returns in the third column. It will not be filtered unless `filterStd = TRUE` is specified.

`Protractor`
: The xts matrix of the rays from IR = -3 in the first column to IR = 3 in the seventh column used in the CUSUM IR as a protractor.

`Excess_Volatility`
: The annualized Standard deviations

Author(s)

Chindhanai Uthaisaad.

References

Philips, T. K., Yashchin, E. and Stein, D. M. (2003). "Using Statistical Process Control to Monitor Active Managers", Journal of Portfolio Management, Fall 2003, pp. 86-94.

Examples

```
data("cusumData")
results = cusumActMgr(portfolioName = "Parvest", benchmarkName = "RUS2500", data = cusumData)
```

`cusumData` *Parvest and Russell2500*

Description

Data in the example is an xts object containing two monthly returns in each column. The first column is the fund returns and the second column is the benchmark returns. The data is from Jan 2005 to Dec 2003.

Usage

```
data("cusumData")
```

Source

TBA

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, dp = NULL, 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.
dp	a vector of length 3, whose elements represent sigma, skew and ekurt, respectively. If dp is specified, the individual parameters cannot be set. Default is NULL.
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)
```

`exposuresTseries` *Time series plots of Style Exposures*

Description

Plot the time series plot of returns and style exposures of the data.

Usage

```
exposuresTseries(data, tickers = NULL, which.exposures = c("SIZE", "P2B",
  "ENTVAL"), plot.returns = TRUE, axis.cex = 1, plot.type = "b", ...)
```

Arguments

`data`	Dataframe with time series of style exposures and returns
`tickers`	character or vector of characters of Ticker symbols
`which.exposures`	exposures to plot
`plot.returns`	Logical. If 'TRUE', returns are also plotted along with the exposures.
`axis.cex`	numeric value to control the size of axis labels
`plot.type`	character defining the type of plot. Default is 'b'
`...`	other optional arguments

Value

Time series plots

Author(s)

Avinash Acharya

Examples

```
data("factorDataSetDjia5Yrs")

#One ticker with returns plot and one exposure
tickers = "BAC"
exposuresTseries(factorDataSetDjia5Yrs,tickers = tickers,which.exposures = "MKTCAP")

#Multiple Tickers without returns and all exposures.
tickers = c("AA", "BAC", "IBM")
exposuresTseries(factorDataSetDjia5Yrs,tickers = tickers,plot.returns = FALSE)

#Multiple tickers With returns
exposuresTseries(factorDataSetDjia5Yrs,tickers = tickers,plot.returns = TRUE, axis.cex = 0.8,
plot.type="b")
```

---

factorDataSetDjia *DJIA stocks Compustat factors 14yrs*

Description

Contains returns for 30 DJIA stocks spanned across 9 Sectors -ENERGY, COSTAP, INDUS,T MATRLS, FINS, INFOTK, HEALTH, CODISC, and TELCOM stocks along with 4 factor data (MKTCAP, ENTVAL, P2B, EV2S, SIZE) starting from Jan 2000 to march 2013.

The 9 Sectors correspond to Energy,ConsumerStaples, Industrials, Materials, Financials, InformationTechnology, HealthCare, ConsumerDiscretionary and Telecommunications respectively.

Usage

```
data("factorDataSetDjia")
```

Source

TBA

factorDataSetDjia5Yrs *DJIA stocks Compustat factors 5yrs*

Description

Contains returns for 30 DJIA stocks spanned across 9 Sectors -ENERGY, COSTAP, INDUS,T MATRLS, FINS, INFOTK, HEALTH, CODISC, and TELCOM stocks along with 4 factor data (MKTCAP, ENTVAL, P2B, EV2S, SIZE) starting from from Jan 2008 to Dec 2012.

The 9 Sectors correspond to Energy, ConsumerStaples, Industrials, Materials, Financials, InformationTechnology, HealthCare, ConsumerDiscretionary and Telecommunications respectively.

Usage

```
data("factorDataSetDjia5Yrs")
```

Source

TBA

fitFfm *Fit a fundamental factor model using cross-sectional regression*

Description

Fit a fundamental (cross-sectional) factor model using ordinary least squares or robust regression. Fundamental factor models use observable asset specific characteristics (or) fundamentals, like industry classification, market capitalization, style classification (value, growth) etc. to calculate the common risk factors. An object of class "ffm" is returned.

Usage

```
fitFfm(data, asset.var, ret.var, date.var, exposure.vars, weight.var = NULL,
  fit.method = c("LS", "WLS", "Rob", "W-Rob"), rob.stats = FALSE,
  full.resid.cov = FALSE, z.score = c("none", "crossSection", "timeSeries"),
  add.intercept = FALSE, lag.exposures = TRUE,
  resid.scale.type = c("stdDev", "EWMA", "robEWMA", "GARCH"),
  GARCH.params = list(omega = 0.09, alpha = 0.1, beta = 0.81), lambda = 0.9,
  GARCH.MLE = FALSE, std.return = FALSE, analysis = c("none", "ISM",
  "NEW"), target.vol = 0.06, ...)

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

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

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

Arguments

data	data.frame of the balanced panel data containing the variables asset.var, ret.var, exposure.vars, date.var and optionally, weight.var.
asset.var	character; name of the variable for asset names.
ret.var	character; name of the variable for asset returns.
date.var	character; name of the variable containing the dates coercible to class Date.
exposure.vars	vector; names of the variables containing the fundamental factor exposures.
weight.var	character; name of the variable containing the weights used when standarizing style factor exposures. Default is NULL. See Details.
fit.method	method for estimating factor returns; one of "LS", "WLS" "Rob" or "W-Rob". See details. Default is "LS".
rob.stats	logical; If TRUE, robust estimates of covariance, correlation, location and univariate scale are computed as appropriate (see Details). Default is FALSE.
full.resid.cov	logical; If TRUE, a full residual covariance matrix is estimated. Otherwise, a diagonal residual covariance matrix is estimated. Default is FALSE.
z.score	method for exposure standardization; one of "none", "crossSection", or "timeSeries". Default is "none".
add.intercept	logical; If TRUE, intercept is added in the exposure matrix. Default is FALSE.
lag.exposures	logical; If TRUE, the style exposures in the exposure matrix are lagged by one time period. Default is TRUE.
resid.scale.type	character; method for computing weights when fit.method is set to WLS or W-Rob; one of stdDev, EWMA, robEWMA, or GARCH. Default is stdDev. See details.
GARCH.params	list containing GARCH parameters omega, alpha, and beta. Default values are 0.09, 0.1, 0.81 respectively. Valid only when GARCH.MLE is set to FALSE.
lambda	value of lambda to be used for the EWMA estimation of residual variances. Default is 0.9.
GARCH.MLE	logical; If TRUE, GARCH parameters are estimated by maximum liklihood estimation. Default is FALSE.
std.return	logical; If TRUE, the returns will be standardized using GARCH(1,1) volatilities. Default is FALSE.
analysis	method used in the analysis of fundamental law of active management; one of "none", "ISM", or "NEW". Default is "none".
target.vol	numeric; the targeted portfolio volatility in the analysis. Default is 0.06.
...	potentially further arguments passed.
object	a fit object of class ffm which is returned by fitFfm

Details

Estimation method "LS" corresponds to ordinary least squares using lm and "Rob" is robust regression using lmRob. "WLS" is weighted least squares using estimates of the residual variances from LS regression as weights (feasible GLS). Similarly, "W-Rob" is weighted robust regression.

The weights to be used in "WLS" or "W-Rob" can be set using resid.scale.type argument which computes the residual variances in one of the following ways - sample variance, EWMA, Robust EWMA and GARCH(1,1). The inverse of these residual variances are used as the weights. For EWMA model, lambda = 0.9 is used as default and for GARCH(1,1) omega = 0.09, alpha = 0.1,

and beta = 0.81 are used as default as mentioned in Martin & Ding (2017). These default parameters can be changed using the arguments `lambda`, `GARCH.params` for EWMA and GARCH respectively. To compute GARCH parameters via MLE, set `GARCH.MLE` to `TRUE`.

Standardizing style factor exposures: The exposures can be standardized into z-scores using regular or robust (see `rob.stats`) measures of location and scale. Further, `weight.var`, a variable such as market-cap, can be used to compute the weighted mean exposure, and an equal-weighted standard deviation of the exposures about the weighted mean. This may help avoid an ill-conditioned covariance matrix. Default option equally weights exposures of different assets each period.

If `rob.stats=TRUE`, `covRob` is used to compute a robust estimate of the factor covariance/correlation matrix, and, `scaleTau2` is used to compute robust tau-estimates of univariate scale for residuals during "WLS" or "W-Rob" regressions. When standardizing style exposures, the `median` and `mad` are used for location and scale respectively. When `resid.scale.type` is EWMA or GARCH, the residual covariance is equal to the diagonal matrix of the estimated residual variances in last time period.

The original function was designed by Doug Martin and initially implemented in S-PLUS by a number of University of Washington Ph.D. students: Christopher Green, Eric Aldrich, and Yindeng Jiang. Guy Yollin ported the function to R and Yi-An Chen modified that code. Sangeetha Srinivasan re-factored, tested, and expanded the functionalities and S3 methods. Avinash Acharya and Chindhanai Uthaisaad further added new functionalities.

Value

`fitFfm` returns an object of class `"ffm"` 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 `"ffm"` is a list containing the following components:

`factor.fit`	list of fitted objects that estimate factor returns in each time period. Each fitted object is of class `lm` if `fit.method="LS"` or `"WLS"`, or, class `lmRob` if `fit.method="Rob"` or `"W-Rob"`.
`beta`	N x K matrix of factor exposures for the last time period.
`factor.returns`	xts object of K-factor returns (including intercept).
`residuals`	xts object of residuals for N-assets.
`r2`	length-T vector of R-squared values.
`factor.cov`	K x K covariance matrix of the factor returns.
`g.cov`	G x G covariance matrix of the coefficients for a Sector plus Market and Sector plus Country plus Global Market models.
`resid.cov`	N x N covariance matrix of residuals.
`return.cov`	N x N return covariance estimated by the factor model, using the factor exposures from the last time period.
`restriction.mat`	restriction matrix used in the computation of f=Rg.
`resid.var`	N x T matrix of estimated residual variances. It will be a length-N vector of sample residual variances when `resid.scale.type` is set to `stdDev`
`call`	the matched function call.
`data`	data frame object as input.

date.var	date.var as input
ret.var	ret.var as input
asset.var	asset.var as input.
exposure.vars	exposure.vars as input.
weight.var	weight.var as input.
fit.method	fit.method as input.
asset.names	length-N vector of asset names.
factor.names	length-K vector of factor.names.
time.periods	length-T vector of dates.
activeWeights	active weights obtaining from the fundamental law of active management
activeReturns	active returns corresponding to the active weights
IR	the vector of Granold-K, asymptotic IR, and finite-sample IR.

Where N is the number of assets, K is the number of factors (including the intercept or dummy variables) and T is the number of unique time periods.

Author(s)

Sangeetha Srinivasan, Guy Yollin, Yi-An Chen, Avinash Acharya and Chindhanai Uthaisaad

References

Menchero, J. (2010). The Characteristics of Factor Portfolios. Journal of Performance Measurement, 15(1), 52-62.

Grinold, R. C., & Kahn, R. N. (2000). Active portfolio management (Second Ed.). New York: McGraw-Hill.

Ding, Z., & Martin, R. D. (2017). The fundamental law of active management: Redux. Journal of Empirical Finance, 43, 91-114.

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

paFm for Performance Attribution.

Examples

```
# Load fundamental and return data
 data("factorDataSetDjia5Yrs")

# fit a fundamental factor model
exposure.vars <- c("P2B", "MKTCAP")
fit <- fitFfm(data=factorDataSetDjia5Yrs, asset.var="TICKER", ret.var="RETURN",
              date.var="DATE", exposure.vars=exposure.vars)
names(fit)

# fit a Industry Factor Model with Intercept
exposure.vars <- c("SECTOR","P2B")
fit1 <- fitFfm(data=factorDataSetDjia5Yrs, asset.var="TICKER", ret.var="RETURN",
               date.var="DATE", exposure.vars=exposure.vars, add.intercept=TRUE)

# Fit a SECTOR+COUNTRY+Style model with Intercept
```

```
# Create a COUNTRY column with just 3 countries

 factorDataSetDjia5Yrs$COUNTRY = rep(rep(c(rep("US",1),rep("GERMANY",1)),11),60)
 exposure.vars= c("SECTOR","COUNTRY","P2B","MKTCAP")

 fit.MICM <- fitFfm(data=factorDataSetDjia5Yrs, asset.var="TICKER", ret.var="RETURN",
                    date.var="DATE", exposure.vars=exposure.vars, add.intercept=TRUE)
```

`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 (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. 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.

When N >= T, Asymptotic Principal Component Analysis (APCA) is performed. The user can directly specify k similar to PCA above, or a method to automatically 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). Users can choose the maximum number of factors, max.k, to consider with these methods. The default for max.k is set to be 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 LS 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 LS 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-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 asset names, whose returns are the dependent variable in the factor model.
`factor.names`	vector containing names of the factors.
`mkt.name`	name of the column for market returns. Default is `NULL`.
`rf.name`	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 `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.

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 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="LS" or "DLS"`, class `lmRob` 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`	asset.names as input.
`factor.names`	factor.names as input.

`mkt.name` 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

```
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=1)

# plot(fit) # this presents a menu for group plots
# select desired plot from the menu (auto-looped for multiple plots)

# 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, efficiency = 0.9, mxr = 50,
  mxf = 50, mxs = 50, 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, 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.
`efficiency`	the asymptotic efficiency of the final estimate for `fit.method="Robust"`. Default is 0.9.
`mxr`	the maximum number of iterations in the refinement step. Default is 50.
`mxf`	the maximum number of iterations for computing final coefficient estimates. Default is 50.
`mxs`	the maximum number of iterations for computing scale estimate. Default is 50.
`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 lmRob, 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.
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.

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 "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`, `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-LS, 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("LS", "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 "LS", "DLS" or "Robust". See details. Default is "LS".
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="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 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 LS 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 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 names of assets, whose returns or excess returns are the dependent variable.
mkt.name	name of the column for market returns (required).
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".
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 from the database
data(managers)

# 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 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 "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 from the database
data(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",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, 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, 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)

# Fundamental factor Model
data(Stock.df)
exposure.vars <- c("BOOK2MARKET", "LOG.MARKETCAP", "GICS.SECTOR")
fit2 <- fitFfm(data=stock, asset.var="TICKER", ret.var="RETURN",
              date.var="DATE", exposure.vars=exposure.vars)
fmCov(fit2)
```

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

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, type="normal")
ES.decomp$cES

# Fundamental Factor Model
data(Stock.df)
exposure.vars <- c("BOOK2MARKET", "LOG.MARKETCAP")
fit <- fitFfm(data=stock, asset.var="TICKER", ret.var="RETURN",
              date.var="DATE", exposure.vars=exposure.vars)
ES.decomp <- fmEsDecomp(fit, type="normal")
head(ES.decomp$cES)
```

---

fmmc *Compute fmmc objects that can be used for calcuation of estimates and their standard errors*

Description

Compute fmmc objects that can be used for calcuation of estimates and their standard errors

Usage

```
fmmc(R, factors, parallel = FALSE, ...)
```

Arguments

R	matrix of returns in xts format
factors	matrix of factor returns in xts format
parallel	flag to utilize multiplecores on the cpu. All cores are used.
...	Arguments that must be passed to fitTsfm

Details

This method takes in data and factors as xts objects where multiple time series with different starting dates are merged together. It then computes FMMC objects as described in Jiang and Martin (2013)

Value

returns an list of fmmc objects

Author(s)

Rohit Arora

References

Yindeng Jiang and Richard Doug Martin. Better Risk and Performance Estimates with Factor Model Monte Carlo. SSRN Electronic Journal, July 2013.

`fmmc.estimate.se` *Main function to calculate the standard errror of the estimate*

Description

Main function to calculate the standard errror of the estimate

Usage

```
fmmc.estimate.se(fmmcObjs, fun = NULL, se = FALSE, nboot = 100,
  parallel = FALSE)
```

Arguments

`fmmcObjs`	A list of fmmc objects computed using .fmmc.proc and containing bootstrapped returns
`fun`	A callback function where the first argument is returns and all the other arguments are bounded to values
`se`	A flag to indicate if standard error for the estimate must be calculated
`nboot`	Number of bootstrap samples
`parallel`	A flag to indicate if multiple cpu cores must be used

Details

This method takes in a list of fmmc objects and a callback function to compute an estimate. The first argument of the callback function must be the data bootstrapped using fmmc procedure. The remaining arguments can be suitably bound to the parameters as needed. This function can also be used to calculate the standard error using the se flag.

Value

returns the estimates and thier standard errors given fmmc objects

Author(s)

Rohit Arora

`fmmcSemiParam` *Semi-parametric factor model Monte Carlo*

Description

Simulate asset returns using semi-parametric Monte Carlo, by making use of a fitted factor model. Residuals are randomly generated from a chosen parametric distribution (Normal, Cornish-Fisher or Skew-t). Factor returns are resampled through non-parametric or stationary bootstrap.

Usage

```
fmmcSemiParam(B = 1000, factor.ret, beta, alpha, resid.par,
  resid.dist = c("normal", "Cornish-Fisher", "skew-t", "empirical"),
  boot.method = c("random", "block"), seed = 123)
```

Arguments

`B`	number of bootstrap samples. Default is 1000.
`factor.ret`	`T x K` matrix or data.frame of factor returns having a complete history of data.
`beta`	`N x K` matrix of factor betas.
`alpha`	`N x 1` matrix of factor alphas (intercepts). If missing, these are assumed to be 0 for all funds.
`resid.par`	matrix of parameters for the residual distribution. See `Details`.
`resid.dist`	the residual distribution; one of "normal", "Cornish-Fisher" or "skew-t". Default is "normal".
`boot.method`	the resampling method for factor returns; one of "random" or "block".
`seed`	integer to set random number generator state before resampling factor returns.

Details

Refer to Yindeng Jiang's PhD thesis referenced below for motivation and empirical results. An abstract can be found at <http://gradworks.umi.com/33/77/3377280.html>.

`T` is the no. of observations, `K` is the no. of factors, `N` is the no. of assets or funds, `P` is the no. of parameters for the residual distribution and `B` is the no. of bootstrap samples.

The columns in `resid.par` depend on the choice of `resid.dist`. If `resid.dist = "normal"`, `resid.par` has one column for standard deviation. If `resid.dist = "Cornish-Fisher"`, `resid.par` has three columns for sigma=standard deviation, skew=skewness and ekurt= excess kurtosis. If `resid.dist = "skew-t"`, `resid.par` has four columns for xi=location, omega=scale, alpha=shape, and nu=degrees of freedom. Cornish-Fisher distribution is based on the Cornish-Fisher expansion of the Normal quantile. If `resid.dist = "empirical"`, `resid.par` should be the TxN residuals retunred by the ffm object. Skew-t is the skewed Student's t-distribution– Azzalini and Captiano. The parameters can differ across funds, though the type of distribution is the same.

Bootstrap method: "random" corresponds to random sampling with replacement, and "block" corresponds to stationary block bootstrap– Politis and Romano (1994).

Value

A list containing the following components:

`sim.fund.ret` B x N matrix of simulated fund returns.

`boot.factor.ret` B x K matrix of resampled factor returns.

`sim.residuals` B x N matrix of simulated residuals.

Author(s)

Eric Zivot, Yi-An Chen, Sangeetha Srinivasan.

References

Jiang, Y. (2009). Factor model Monte Carlo methods for general fund-of-funds portfolio management. University of Washington.

See Also

http://gradworks.umi.com/33/77/3377280.html

Examples

```
# fit a time series factor model for all assets
data(managers)
fit <- fitTsfm(asset.names=colnames(managers[,(1:6)]),
               factor.names=colnames(managers[,(7:9)]), data=managers)

# bootstrap returns using the fitted factor model, Normal dist. for residuals
resid.par <- as.matrix(fit$resid.sd,1,6)
fmmc.returns <- fmmcSemiParam(factor.ret=managers[,(7:9)], beta=fit$beta,
                              alpha=fit$alpha, resid.par=resid.par)

# Cornish-Fisher distribution for residuals
resid.par <- cbind(c(1,2,1,3,0.1,0.5), rnorm(6), c(2,3,1,2,1,0))
colnames(resid.par) <- c("var","skew","xskurt")
rownames(resid.par) <- colnames(managers[,(1:6)])
fmmc.returns.CF <- fmmcSemiParam(factor.ret=managers[,(7:9)], beta=fit$beta,
                                 alpha=fit$alpha, resid.par=resid.par,
                                 resid.dist="Cornish-Fisher")

# skew-t distribution
resid.par <- cbind(rnorm(6), c(1,2,1,3,0.1,0.5), rnorm(6), c(2,3,1,6,10,100))
colnames(resid.par) <- c("xi","omega","alpha","nu")
rownames(resid.par) <- colnames(managers[,(1:6)])
fmmc.returns.skewt <- fmmcSemiParam(factor.ret=managers[,(7:9)],
                                    beta=fit$beta, alpha=fit$alpha,
                                    resid.dist="skew-t", resid.par=resid.par)

#Empirical deistribution
data("factorDataSetDjia5Yrs")
exposure.vars <- c("P2B", "MKTCAP", "SECTOR")
fit.ffm <- fitFfm(data=factorDataSetDjia5Yrs, asset.var="TICKER",
                  ret.var="RETURN", date.var="DATE",
                  exposure.vars=exposure.vars, addIntercept = FALSE)
resid.par = fit.ffm$residuals
```

```
fmmc.returns.ffm <- fmmcSemiParam(factor.ret=fit.ffm$factor.returns,
                                  beta=fit.ffm$beta, resid.par=resid.par,
                                  resid.dist = "empirical", boot.method = "block")
```

`fmRsq` *Factor Model R-Squared and Adj R-Squared Values*

Description

Calcluate and plot the Factor Model R-Squared, Adjusted R-Squared for a portfolio of assets

Usage

```
fmRsq(ffmObj, ...)

## S3 method for class 'ffm'
fmRsq(ffmObj, rsq = T, rsqAdj = F, plt.type = 2,
  digits = 2, isPrint = T, isPlot = T, lwd = 2, stripText.cex = 1,
  axis.cex = 1, title = TRUE, ...)
```

Arguments

`ffmObj`	an object of class `ffm` produced by `fitFfm`
`...`	potentially further arguments passed.
`rsq`	logical; if `TRUE`, Factor Model R-squared values are computed for the portfolio. Default is `TRUE`.
`rsqAdj`	logical; if `TRUE`, Adjusted R-squared values are computed for the portfolio. Default is `FALSE`.
`plt.type`	a number to indicate the type of plot for plotting Factor Model R-squared/Adj. R-squared values. 1 indicates barplot, 2 indicates time series xy plot. Default is 2.
`digits`	an integer indicating the number of decimal places to be used for rounding. Default is 2.
`isPrint`	logical. if `TRUE`, the time series of the computed factor model values is printed along with their mean values. Else, only the mean values are printed. Default is `TRUE`.
`isPlot`	logical. if `TRUE`, the time series of the output is plotted. Default is `TRUE`.
`lwd`	line width relative to the default. Default is 2.
`stripText.cex`	a number indicating the amount by which strip text in the plot(s) should be scaled relative to the default. 1=default, 1.5 is 50% larger, 0.5 is 50% smaller, etc.
`axis.cex`	a number indicating the amount by which axis in the plot(s) should be scaled relative to the default. 1=default, 1.5 is 50% larger, 0.5 is 50% smaller, etc.
`title`	logical. if `TRUE`, the plots will have the main tiltle. default is `TRUE`.

Value

fmRsq returns the sample mean values and plots the time series of corresponding R squared values and the Variance Inflation factors depending on the values of rsq, rsqAdj and VIF. The time series of the output values are also printed if isPrint is TRUE

Author(s)

Avinash Acharya and Doug Martin

Examples

```
#Load the data
 data("factorDataSetDjia5Yrs")

#Fit a Ffm
require(factorAnalytics)
 fit <- fitFfm(data=factorDataSetDjia5Yrs, asset.var="TICKER", ret.var="RETURN",
              date.var="DATE", exposure.vars="SECTOR")

#Calcuate and plot the portfolio R-squared values
 fmRsq(fit)

 fit1 <- fitFfm(data=factorDataSetDjia5Yrs, asset.var="TICKER", ret.var="RETURN",
           date.var="DATE", exposure.vars=c("SECTOR", "P2B", "EV2S", "MKTCAP"), addIntercept=TRUE)

#Plot and print the time series of Adj R-squared and VIF values
 fmRsq(fit1, rsqAdj=TRUE, isPrint=TRUE, plt.type = 2)
```

---

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, sfm 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, 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
```

---

fmTstats *t-stats and Plots for a fitted Fundamental Factor Model*

Description

Calculate and plot the time series of the t-statistic values and the number of risk indices with significant t-stats for a fundamentally fit object.

Usage

```
fmTstats(ffmObj, ...)

## S3 method for class 'ffm'
fmTstats(ffmObj, isPlot = TRUE, isPrint = FALSE,
  whichPlot = "all", color = c("black", "cyan"), lwd = 2, digits = 2,
  z.alpha = 1.96, layout = c(2, 3), type = "h", scale = "free",
  stripText.cex = 1, axis.cex = 1, title = TRUE, ...)
```

Arguments

ffmObj	an object of class ffm produced by fitFfm
...	potentially further arguments passed.
isPlot	logical. If FALSE no plots are displayed.
isPrint	logical. if TRUE, the time series of the computed factor model values is printed. default is FALSE,

whichPlot	string indicating the plot(s) to be plotted. Choose from ("all", "tStats", "significantTstatsV", "significantTstatsH", "significantTstatsLikert"). Three variants of significantTstats stand for vertical, horizontal and likert barplots. Default is all plotting t-stats and significant t-stats with vertical bars.
color	length 2 vector specifying the plotting color for t-stats plot and for barplot respectively. default is c("black", "cyan")
lwd	line width relative to the default. default is 2.
digits	an integer indicating the number of decimal places to be used for rounding. default is 2.
z.alpha	critical value corresponding to the confidence interval. default is 1.96 i.e 95% C.I
layout	numeric vector of length 2 or 3 giving the number of columns, rows, and pages (optional) in the xyplot of t-statistics. default is c(2,3).
type	character. Type of the xyplot of t-statistics; "l" for lines, "p" for points, "h" for histogram like (or high-density) vertical lines and "b" for both. Deafault is "h".
scale	character. It determines how axis limits are calculated for each panel. Possible values are "same" , "free" (default) and "sliced".
stripText.cex	a number indicating the amount by which strip text in the plot(s) should be scaled relative to the default. 1=default, 1.5 is 50% larger, 0.5 is 50% smaller, etc.
axis.cex	a number indicating the amount by which axis in the plot(s) should be scaled relative to the default. 1=default, 1.5 is 50% larger, 0.5 is 50% smaller, etc.
title	logical. if TRUE, the plots will have the main tiltle. default is TRUE.

Value

fmTstats plots the t-stats and significant t-stats values if isPlot is TRUE and returns a list with following components:

tstats	an xts object of t-stats values.
z.alpha	critical value corresponding to the confidence interval.

Author(s)

Avinash Acharya and Doug Martin

Examples

```
 data("factorDataSetDjia5Yrs")

#Fit a Ffm with style factors only
 require(factorAnalytics)
 fit <- fitFfm(data = factorDataSetDjia5Yrs,exposure.vars = c("MKTCAP","ENTVAL","P2B","EV2S"),
          date.var = "DATE", ret.var = "RETURN", asset.var = "TICKER", fit.method="WLS",z.score = "crossSect

#Compute time series of t-stats and number of significant t-stats
 stats = fmTstats(fit, isPlot = TRUE, lwd = 2, color = c("blue", "blue"), z.alpha =1.96)

fit1 <- fitFfm(data=factorDataSetDjia5Yrs, asset.var="TICKER", ret.var="RETURN",
            date.var="DATE", exposure.vars=c("SECTOR","MKTCAP","ENTVAL","P2B"), addIntercept=TRUE)
```

```
#Compute time series of t-stats and number of significant t-stats
 stats = fmTstats(fit1, isPlot = TRUE, z.alpha =1.96)

# Fit a SECTOR+COUNTRY+Style model with Intercept
# Create a COUNTRY column with just 3 countries

 factorDataSetDjia5Yrs$COUNTRY = rep(rep(c(rep("US", 1 ),rep("GERMANY", 1 )), 11), 60)
 exposure.vars= c("SECTOR", "COUNTRY","P2B", "MKTCAP")

 fit.MICM <- fitFfm(data=factorDataSetDjia5Yrs, asset.var="TICKER", ret.var="RETURN",
                    date.var="DATE", exposure.vars=exposure.vars, addIntercept=TRUE)
 stats = fmTstats(fit.MICM, isPlot = TRUE, z.alpha =1.96)
```

`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 of historical factor returns (tsfm) or estimated factor returns (sfm and ffm).
`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`, `fitSfm`, `fitFfm` for the different factor model fitting functions.

`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, type="normal")
VaR.decomp$cVaR

# Fundamental Factor Model
data(Stock.df)
exposure.vars <- c("BOOK2MARKET", "LOG.MARKETCAP")
fit <- fitFfm(data=stock, asset.var="TICKER", ret.var="RETURN",
              date.var="DATE", exposure.vars=exposure.vars)
VaR.decomp <- fmVaRDecomp(fit, type="normal")
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)
```

`managers.ffm` *managers data for ffm*

Description

Hypothetical Alternative Asset Manager and Benchmark Data for Time Series Factor Model Fit

Usage

```
data("managers.ffm")
```

Source

TBA

`mktSP` *S&P 500 Returns*

Description

S&P 500 return from Yahoo

Usage

```
data("mktSP")
```

Source

Yahoo

mktUS *US Market Returns*

Description

Monthly returns including all distributions, on a value-weighted market portfolio of NYSE/AMEX/NASDAQ

Usage

```
data("mktUS")
```

Source

WRDS

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.ffm *Plots from a fitted fundamental factor model*

Description

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

Usage

```
## S3 method for class 'ffm'
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 `ffm` produced by `fitFfm`.
`which`	a number to indicate the type of plot. If multiple plots are required, specify a subset from 1:12 for group plots and 1:13 for individual plots. If `which=NULL` (default), the following menu appears: For plots of a group of assets: 1 = Distribution of factor returns, 2 = Factor exposures from the last period, 3 = Actual and Fitted asset returns, 4 = Time-series of R-squared values, 5 = Residual variance across assets, x 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 = Time series of factor returns,

For individual asset plots:
1 = Actual and fitted,
2 = Actual vs. fitted,
3 = Residuals vs. fitted,
4 = Residuals with standard error bands,
5 = Time series of squared residuals,
6 = Time series of absolute residuals,
7 = SACF and PACF of residuals,
8 = SACF and PACF of squared residuals,
9 = SACF and PACF of absolute residuals,
10 = Non-parametric density of residuals with normal overlaid,
11 = Non-parametric density of residuals with skew-t overlaid,
12 = Histogram of residuals with non-parametric density and normal overlaid,
13 = QQ-plot of residuals

`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 `ffm` object `x` contains only a single asset fit, plot.ffm can infer `asset.name` without user input.

Author(s)

Eric Zivot, Sangeetha Srinivasan and Yi-An Chen

See Also

fitFfm, residuals.ffm, fitted.ffm, fmCov.ffm and summary.ffm 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(3,4,12) - 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, I(18) - plot.zoo, G(1) - chart.Boxplot, G(2,5,9,10,11) - barchart, G(6) - chart.Correlation and G(7,8) - corrplot.mixed.

Examples

```
# load data from the database
data(Stock.df)
fit.style.sector <- fitFfm(data=stock, asset.var="TICKER", ret.var="RETURN",
                           exposure.vars=c("GICS.SECTOR","LOG.MARKETCAP",
                                           "BOOK2MARKET"), date.var="DATE")

# for group plots (default), user can select plot option from menu prompt
# menu is repeated to get multiple types of plots based on the same fit
# plot(fit.style.sector)

# choose specific plot option(s) using which
# plot all factor exposures from the last time period for 1st 10 assets
plot(fit.style.sector, which=2, f.sub=1:12, a.sub=1:10)

# plot factor model residuals scatterplot matrix, with histograms, density
# overlays, correlations and significance stars
plot(fit.style.sector, which=6)

# for individual plots: set plot.single=TRUE and specify asset.name
# histogram of residuals from an individual asset's factor model fit
plot(fit.style.sector, plot.single=TRUE, asset.name="MSFT", which=12)
```

---

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 = NULL, f.sub = 1:2, a.sub = 1:6, n.top = 3,
  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, eig.max = 0.9, cum.var = TRUE, ...)
```

Arguments

x	an object of class sfm produced by fitSfm.
which	a number to indicate the type of plot. If a subset of the plots is required, specify a subset of the numbers 1:13 for group plots and 1:18 for individual plots. If which=NULL (default), the following menu appears:

For plots of a group of assets:
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 residuals scatterplot matrix, 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 = Factor mimicking portfolio weights - top long and short positions in each factor,
13 = Asset correlations - top long and short positions in each factor

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,
16 = CUSUM test-LS residuals,
17 = Recursive estimates (RE) test of LS regression coefficients,
18 = Rolling regression over a 24-period observation window

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.
n.top	scalar; number of largest and smallest weights to display for each factor mimicking 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. 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.
eig.max	scalar in (0,1] for limiting the screeplot to factors that explain a given percent of the variance. Default is 0.9.
cum.var	logical; If TRUE, the cumulative fraction of the variance is printed above each bar in the screeplot of eigenvalues. Default is TRUE.
...	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.

Author(s)

Eric Zivot, Sangeetha Srinivasan and Yi-An Chen

See Also

fitSfm, residuals.sfm, fitted.sfm, fmCov.sfm and summary.sfm for statistical 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) - chart.TimeSeries, I(2,3,4) - plot.default, I(3,4) - panel.smooth, I(8,9,10) - chart.ACFplus, I(11,12) - plot.density,

I(13), G(4,5) - chart.Histogram, I(14) - chart.QQPlot, I(15,16,17) - plot.efp, I(18) - plot.zoo, G(1,12) - barplot, G(2) - xyplot, G(3,9,10,11) - barchart, G(6) - chart.Correlation and G(7,8,13) - corrplot.mixed.

Examples

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

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

# for group plots (default), user can select plot option from menu prompt
# menu is repeated to get multiple types of plots based on the same fit
# plot(fit.apca)

# choose specific plot option(s) using which
# plot the first 4 factor betas of the first 4 assets fitted above
plot(fit.apca, f.sub=1:4, a.sub=1:4, which=3)

# plot factor model residuals scatterplot matrix, with histograms, density
# overlays, correlations and significance stars
plot(fit.apca, which=6)

# for individual plots: set plot.single=TRUE and specify asset.name
# histogram of residuals from an individual asset's factor model fit
plot(fit.apca, plot.single=TRUE, asset.name="AFL", which=13)
```

`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,
16 = CUSUM test-LS residuals,
17 = Recursive estimates (RE) test of LS regression coefficients,
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.

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, I(18) - plot.zoo, G(1,2,4,5,9,10,11) - barchart, G(6) - chart.Correlation and G(7,8) - corrplot.mixed.

Examples

```
# load data from the database
data(managers)
fit.macro <- fitTsfm(asset.names=colnames(managers[,(1:6)]),
                     factor.names=colnames(managers[,(7:9)]),
                     rf.name="US.3m.TR", data=managers)

# for group plots (default), user can select plot option from menu prompt
# menu is repeated to get multiple types of plots based on the same fit
# plot(fit.macro)

# choose specific plot option(s) using which
# plot the first 2 factor betas of first 4 assets fitted above
plot(fit.macro, f.sub=1:2, a.sub=1:4, which=2)
```

```
# plot factor model residuals scatterplot matrix, with histograms, density
# overlays, correlations and significance stars
plot(fit.macro, which=6)

# for individual plots: set plot.single=TRUE and specify asset.name
# histogram of residuals from an individual asset's factor model fit
plot(fit.macro, plot.single=TRUE, asset.name="HAM1", which=13)
```

`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.

Author(s)

Yi-An Chen

See Also

`fitTsfmUpDn`

Examples

```
# load data from the database
 data(managers)
# example: Up and down market factor model with  fit
 fitUpDn <- fitTsfmUpDn(asset.names=colnames(managers[,(1:6)]),mkt.name="SP500.TR",
                        data=managers, fit.method="LS")
# plot the fitted model of every assets, press enter to show the next plot.
 plot(fitUpDn)

# or choose to plot one specific asset
 plot(fitUpDn,asset.name="HAM1")

# add a single market factor model fitted line
 plot(fitUpDn,SFM.line=TRUE,asset.name="HAM1")

# add Robust Up/Dn model fitted line and change legend to show the robust up/dn Beta
 plot(fitUpDn,LSandRob=TRUE,asset.name="HAM1")
```

`portEsDecomp` *Decompose portfolio ES into individual factor contributions*

Description

Compute the factor contributions to Expected Tail Loss or Expected Shortfall (ES) of portfolio 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 portfolio return is less than or equal to its value-at-risk (VaR). Option to choose between non-parametric and Normal.

Usage

```
portEsDecomp(object, ...)

## S3 method for class 'tsfm'
portEsDecomp(object, weights = NULL, p = 0.05,
  type = c("np", "normal"), invert = FALSE, use = "pairwise.complete.obs",
  ...)

## S3 method for class 'ffm'
portEsDecomp(object, weights = NULL, factor.cov, p = 0.05,
  type = c("np", "normal"), invert = FALSE, ...)
```

Arguments

object	fit object of class tsfm, or ffm.
...	other optional arguments passed to quantile and optional arguments passed to cov
weights	a vector of weights of the assets in the portfolio, names of the vector should match with asset names. Default is NULL, in which case an equal weights will be used.
p	tail probability for calculation. Default is 0.05.
type	one of "np" (non-parametric) or "normal" for calculating Es. Default is "np".
invert	a logical variable to choose if change ES to positive number, default is False
use	an optional character string giving a method for computing factor 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".
factor.cov	optional user specified factor covariance matrix with named columns; defaults to the sample covariance matrix.

Details

The factor model for a portfolio'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 portfolio'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 portVaR. This is estimated as a sample average of the observations in that data window.

Value

A list containing

portES	factor model ES of portfolio returns.

mES	length-(K + 1) vector of marginal contributions to Es.
cES	length-(K + 1) vector of component contributions to Es.
pcES	length-(K + 1) vector of percentage component contributions to Es.

Where, K is the number of factors.

Author(s)

Douglas Martin, Lingjie Yi

See Also

fitTsfm, fitFfm for the different factor model fitting functions.

portSdDecomp for factor model Sd decomposition. portVaRDecomp for factor model VaR decomposition.

Examples

```
# Time Series Factor Model
data(managers)
fit.macro <- factorAnalytics::fitTsfm(asset.names=colnames(managers[,(1:6)]),
                    factor.names=colnames(managers[,(7:9)]),
                    rf.name=colnames(managers[,10]), data=managers)
ES.decomp <- portEsDecomp(fit.macro,invert = TRUE)
# get the component contributions
ES.decomp$cES

# random weights
wts = runif(6)
wts = wts/sum(wts)
names(wts) <- colnames(managers)[1:6]
portEsDecomp(fit.macro, wts)

# Fundamental Factor Model
data("stocks145scores6")
dat = stocks145scores6
dat$DATE = as.yearmon(dat$DATE)
dat = dat[dat$DATE >=as.yearmon("2008-01-01") &
          dat$DATE <= as.yearmon("2012-12-31"),]

# Load long-only GMV weights for the return data
data("wtsStocks145GmvLo")
wtsStocks145GmvLo = round(wtsStocks145GmvLo,5)

# fit a fundamental factor model
fit.cross <- fitFfm(data = dat,
              exposure.vars = c("SECTOR","ROE","BP","MOM121","SIZE","VOL121",
              "EP"),date.var = "DATE", ret.var = "RETURN", asset.var = "TICKER",
              fit.method="WLS", z.score = "crossSection")

decomp = portEsDecomp(fit.cross)
#get the factor contributions of risk
decomp$cES
portEsDecomp(fit.cross, weights = wtsStocks145GmvLo)
```

`portSdDecomp` *Decompose portfolio standard deviation into individual factor contributions*

Description

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

Usage

```
portSdDecomp(object, ...)

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

## S3 method for class 'ffm'
portSdDecomp(object, weights = NULL, factor.cov, ...)
```

Arguments

`object`	fit object of class `tsfm`, or `ffm`.
`...`	optional arguments passed to `cov`.
`weights`	a vector of weights of the assets in the portfolio. Default is NULL, in which case an equal weights will be used.
`factor.cov`	optional user specified factor covariance matrix with named columns; defaults to the sample covariance matrix.
`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 a portfolio'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 portfolio's return is given as:

```
portSd = 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 `portSd` 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.

```
portSd = sqrt(beta.star''cov(F.star)beta.star)
mSd = cov(F.star)beta.star / portSd
```

Value

A list containing

portSd	factor model Sd of portfolio return.
mSd	length-(K + 1) vector of marginal contributions to Sd.
cSd	length-(K + 1) vector of component contributions to Sd.
pcSd	length-(K + 1) vector of percentage component contributions to Sd.

Where, K is the number of factors.

Author(s)

Douglas Martin, Lingjie Yi

See Also

fitTsfm, fitFfm for the different factor model fitting functions.

portVaRDecomp for portfolio factor model VaR decomposition. portEsDecomp for portfolio factor model ES decomposition.

Examples

```
# Time Series Factor Model
data(managers)
fit.macro <- factorAnalytics::fitTsfm(asset.names=colnames(managers[,(1:6)]),
                    factor.names=colnames(managers[,(7:9)]),
                    rf.name=colnames(managers[,10]), data=managers)
decomp <- portSdDecomp(fit.macro)
# get the factor contributions of risk
decomp$cSd

# random weights
wts = runif(6)
wts = wts/sum(wts)
names(wts) <- colnames(managers)[1:6]
portSdDecomp(fit.macro, wts)

# Fundamental Factor Model
data("stocks145scores6")
dat = stocks145scores6
dat$DATE = as.yearmon(dat$DATE)
dat = dat[dat$DATE >=as.yearmon("2008-01-01") &
          dat$DATE <= as.yearmon("2012-12-31"),]

# Load long-only GMV weights for the return data
data("wtsStocks145GmvLo")
wtsStocks145GmvLo = round(wtsStocks145GmvLo,5)

# fit a fundamental factor model
fit.cross <- fitFfm(data = dat,
              exposure.vars = c("SECTOR","ROE","BP","MOM121","SIZE","VOL121",
```

```
                "EP"),date.var = "DATE", ret.var = "RETURN", asset.var = "TICKER",
                fit.method="WLS", z.score = "crossSection")

decomp = portSdDecomp(fit.cross)
# get the factor contributions of risk
decomp$cSd
portSdDecomp(fit.cross, wtsStocks145GmvLo)
```

`portVaRDecomp` *Decompose portfolio VaR into individual factor contributions*

Description

Compute the factor contributions to Value-at-Risk (VaR) of portfolio 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 portfolio return is equal to its VaR and approximated by a kernel estimator. Option to choose between non-parametric and Normal.

Usage

```
portVaRDecomp(object, ...)

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

## S3 method for class 'ffm'
portVaRDecomp(object, weights = NULL, factor.cov, p = 0.05,
  type = c("np", "normal"), invert = FALSE, ...)
```

Arguments

`object`	fit object of class `tsfm`, or `ffm`.
`...`	other optional arguments passed to `quantile` and optional arguments passed to `cov`
`weights`	a vector of weights of the assets in the portfolio. Default is NULL, in which case an equal weights will be used.
`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".
`invert`	a logical variable to choose if change VaR to positive number, default is False
`use`	an optional character string giving a method for computing factor 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 a portfolio'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 `portVaR`. This is approximated as described in Epperlein & Smillie (2006); a triangular smoothing kernel is used here.

Value

A list containing

`portVaR`	factor model VaR of portfolio return.
`n.exceed`	number of observations beyond VaR.
`idx.exceed`	a numeric vector of index values of exceedances.
`mVaR`	length-(K + 1) vector of marginal contributions to VaR.
`cVaR`	length-(K + 1) vector of component contributions to VaR.
`pcVaR`	length-(K + 1) vector of percentage component contributions to VaR.

Where, K is the number of factors.

Author(s)

Douglas Martin, Lingjie Yi

See Also

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

`portSdDecomp` for factor model Sd decomposition. `portEsDecomp` for factor model ES decomposition.

Examples

```
# Time Series Factor Model
data(managers)
fit.macro <- factorAnalytics::fitTsfm(asset.names=colnames(managers[,(1:6)]),
                    factor.names=colnames(managers[,(7:9)]),
                    rf.name=colnames(managers[,10]), data=managers)
decomp <- portVaRDecomp(fit.macro,invert = TRUE)
# get the factor contributions of risk
decomp$cVaR

# random weights
wts = runif(6)
wts = wts/sum(wts)
names(wts) <- colnames(managers)[1:6]
```

```
portVaRDecomp(fit.macro, wts)


# Fundamental Factor Model
data("stocks145scores6")
dat = stocks145scores6
dat$DATE = as.yearmon(dat$DATE)
dat = dat[dat$DATE >=as.yearmon("2008-01-01") &
          dat$DATE <= as.yearmon("2012-12-31"),]

# Load long-only GMV weights for the return data
data("wtsStocks145GmvLo")
wtsStocks145GmvLo = round(wtsStocks145GmvLo,5)

# fit a fundamental factor model
fit.cross <- fitFfm(data = dat,
              exposure.vars = c("SECTOR","ROE","BP","MOM121","SIZE","VOL121",
              "EP"),date.var = "DATE", ret.var = "RETURN", asset.var = "TICKER",
              fit.method="WLS", z.score = "crossSection")

decomp = portVaRDecomp(fit.cross)
# get the factor contributions of risk
decomp$cVaR
portVaRDecomp(fit.cross, weights = wtsStocks145GmvLo)
```

`portVolDecomp` *Decompose portfolio variance risk into factor/residual risk*

Description

Decompose portfolio variance risk into factor/residual risk

Usage

```
portVolDecomp(object, ...)

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

## S3 method for class 'ffm'
portVolDecomp(object, weights = NULL, factor.cov, ...)
```

Arguments

`object`	fit object of class `tsfm`, or `ffm`.
`...`	optional arguments passed to `cov`.
`weights`	a vector of weights of the assets in the portfolio. Default is NULL, in which case an equal weights will be used.
`factor.cov`	optional user specified factor covariance matrix with named columns; defaults to the sample covariance matrix.

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".

Value

A vector containing: percent factor contribution to risk portfolio volatility risk, factor volatility risk and residual/specific volatility risk

Author(s)

Douglas Martin, Lingjie Yi

See Also

fitTsfm, fitFfm for the different factor model fitting functions.

portSdDecomp for portfolio factor model VaR decomposition. portVaRDecomp for portfolio factor model VaR decomposition. portEsDecomp for portfolio factor model ES decomposition.

Examples

```
# Time Series Factor Model
data(managers)
fit.macro <- factorAnalytics::fitTsfm(asset.names=colnames(managers[,(1:6)]),
                     factor.names=colnames(managers[,(7:9)]),
                     rf.name=colnames(managers[,10]), data=managers)
decomp <- portVolDecomp(fit.macro)
decomp

# Fundamental Factor Model
data("stocks145scores6")
dat = stocks145scores6
dat$DATE = as.yearmon(dat$DATE)
dat = dat[dat$DATE >=as.yearmon("2008-01-01") &
          dat$DATE <= as.yearmon("2012-12-31"),]

# Load long-only GMV weights for the return data
data("wtsStocks145GmvLo")
wtsStocks145GmvLo = round(wtsStocks145GmvLo,5)

# fit a fundamental factor model
fit.cross <- fitFfm(data = dat,
              exposure.vars = c("SECTOR","ROE","BP","MOM121","SIZE","VOL121",
              "EP"),date.var = "DATE", ret.var = "RETURN", asset.var = "TICKER",
              fit.method="WLS", z.score = "crossSection")

decomp = portVolDecomp(fit.cross)
# get the factor contributions of risk
decomp
```

predict.ffm *Predicts asset returns based on a fitted fundamental factor model*

Description

S3 predict method for object of class ffm.

Usage

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

Arguments

object	an object of class ffm produced by fitFfm.
newdata	data.frame containing the variables asset.var, date.var and the same exact exposure.vars used in the fitted ffm object. If omitted, the predictions are based on the data used for the fit.
pred.date	character; unique date used to base the predictions. Should be coercible to class Date and match one of the dates in the data used in the fitted object.
...	optional arguments passed to predict.lm or predict.lmRob.

Details

The estimated factor returns and potentially new factor exposures are used to predict the asset returns during all dates from the fitted ffm object. For predictions based on estimated factor returns from a specific period use the pred.date argument.

Value

predict.ffm produces a N x T matrix of predicted asset returns, where T is the number of time periods and N is the number of assets. T=1 if pred.date is specified.

Author(s)

Sangeetha Srinivasan

See Also

fitFfm, summary.ffm, predict.lm, predict.lmRob

Examples

```
# Load fundamental and return data
data(Stock.df)

# fit a fundamental factor model
exposure.vars <- c("BOOK2MARKET", "LOG.MARKETCAP")
fit <- fitFfm(data=stock, asset.var="TICKER", ret.var="RETURN",
              date.var="DATE", exposure.vars=exposure.vars)
```

```
# generate random data
newdata <- as.data.frame(unique(stock$TICKER))
newdata$BOOK2MARKET <- rnorm(nrow(newdata))
newdata$LOG.MARKETCAP <- rnorm(nrow(newdata))
pred.fund <- predict(fit, newdata)
```

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

predict(fitUpDn)
```

print.ffm *Prints a fitted fundamental factor model*

Description

S3 print method for object of class ffm. Prints the call, factor model dimension and summary statistics for the estimated factor returns, cross-sectional r-squared values and residual variances from the fitted object.

Refer to summary.ffm for a more detailed summary of the fit at each time period.

Usage

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

Arguments

x	an object of class ffm produced by fitFfm.
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

fitFfm, summary.ffm

Examples

```
data(Stock.df)
exposure.vars <- c("BOOK2MARKET", "LOG.MARKETCAP")
fit <- fitFfm(data=stock, asset.var="TICKER", ret.var="RETURN",
              date.var="DATE", exposure.vars=exposure.vars)
print(fit)
```

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 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.sfm *Prints a fitted statistical factor model*

Description

S3 print method for object of class sfm. Prints the call, factor model dimensions and summary statistics for the estimated factor loadings, r-squared values 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 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.

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", 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 LS fit
fitUpDn <- fitTsfmUpDn(asset.names=colnames(managers[,(1:6)]),mkt.name="SP500.TR",
                       data=managers, fit.method="LS",control=NULL)

print(fitUpDn)
```

repExposures *Portfolio Exposures Report*

Description

Calculate k factor time series based on fundamental factor model. This method takes fundamental factor model fit, 'ffm' object, and portfolio weight as inputs and generates numeric summary and plot visualization.

Usage

```
repExposures(ffmObj, weights = NULL, isPlot = TRUE, isPrint = TRUE,
  scaleType = "free", stripText.cex = 1, axis.cex = 1, stripLeft = TRUE,
  layout = NULL, color = "blue", notch = FALSE, digits = 1,
  titleText = TRUE, which = NULL, type = "b", ...)
```

Arguments

ffmObj	an object of class ffm returned by fitFfm.
weights	a vector of weights of the assets in the portfolio. Default is NULL.
isPlot	logical variable to generate plot or not.
isPrint	logical variable to print numeric summary or not.
scaleType	scaleType controls if use a same scale of y-axis, choose from c('same', 'free')
stripText.cex	a number indicating the amount by which strip text in the plot(s) should be scaled relative to the default. 1=default, 1.5 is 50% larger, 0.5 is 50% smaller, etc.
axis.cex	a number indicating the amount by which axis in the plot(s) should be scaled relative to the default. 1=default, 1.5 is 50% larger, 0.5 is 50% smaller, etc.
stripLeft	logical variable to choose the position of strip, 'TRUE' for drawing strips on the left of each panel, 'FALSE' for drawing strips on the top of each panel. Used only when isPlot = 'TRUE'
layout	layout is a numeric vector of length 2 or 3 giving the number of columns, rows, and pages (optional) in a multipanel display. Used only when isPlot = 'TRUE'
color	character specifying the plotting color for all the plots
notch	logical. if notch is TRUE, a notch is drawn in each side of the boxes. If the notches of two plots do not overlap this is strong evidence that the two medians differ (Chambers et al, 1983, p. 62).Default values is FALSE.
digits	digits of printout numeric summary. Used only when isPrint = 'TRUE'
titleText	logical varible to choose display plot title or not. Default is 'TRUE', and used only when isPlot = 'TRUE'.
which	a number to indicate the type of plot. If a subset of the plots is required, specify a subset of the numbers 1:3 for plots. If which=NULL (default), the following menu appears: For plots of a group of assets: 1 = Time series plot of style factor exposures, 2 = Boxplot of style factor exposures, 3 = Barplot of means and vols of style factor exposures, and means of sector exposures (which have no vol).
type	character. type of lattice plot when which=1; 'l' denotes a line, 'p' denotes a point, and 'b' and 'o' both denote both together.deafault is 'b'.
...	other graphics parameters available in tsPlotMP(time series plot only) can be passed in through the ellipses

Value

A list containing mean and standard deviation of all the factors

Author(s)

Douglas Martin, Lingjie Yi, Avinash

Examples

```
#Load fundamental and return data
data("stocks145scores6")
dat = stocks145scores6
dat$DATE = as.yearmon(dat$DATE)
dat = dat[dat$DATE >=as.yearmon("2008-01-01")
          & dat$DATE <= as.yearmon("2012-12-31"),]

#Load long-only GMV weights for the return data
data("wtsStocks145GmvLo")
wtsStocks145GmvLo = round(wtsStocks145GmvLo,5)

# fit a fundamental factor model
fit.cross <- fitFfm(data = dat,
              exposure.vars = c("SECTOR","ROE","BP","MOM121","SIZE","VOL121",
              "EP"),date.var = "DATE", ret.var = "RETURN", asset.var = "TICKER",
              fit.method="WLS", z.score = "crossSection")

repExposures(fit.cross, wtsStocks145GmvLo, isPlot = FALSE, digits = 4)
repExposures(fit.cross, wtsStocks145GmvLo, isPrint = FALSE, isPlot = TRUE,
             which = 2, add.grid = TRUE, scaleType = 'same')
repExposures(fit.cross, wtsStocks145GmvLo, isPlot = TRUE, which = 1,
             add.grid = FALSE, zeroLine = TRUE, color = 'Blue')
repExposures(fit.cross, wtsStocks145GmvLo, isPrint = FALSE, isPlot = TRUE,
             which = 3, add.grid = FALSE, zeroLine = FALSE, color = 'Blue')
```

---

repReturn *Portfolio return decomposition report*

Description

Decompostite return of portfolio into return of different factors based on fundamental factor model. This method takes fundamental factor model fit, "ffm" object, and portfolio weight as inputs and generates numeric summary and plot visualization.

Usage

```
repReturn(ffmObj, weights = NULL, isPlot = TRUE, isPrint = TRUE,
  layout = NULL, scaleType = "free", stripLeft = TRUE,
  stripText.cex = 1, axis.cex = 1, digits = 1, titleText = TRUE,
  which = NULL, ...)
```

Arguments

ffmObj	an object of class ffm returned by fitFfm.
weights	a vector of weights of the assets in the portfolio. Default is NULL.
isPlot	logical variable to generate plot or not.

isPrint	logical variable to print numeric summary or not.
layout	layout is a numeric vector of length 2 or 3 giving the number of columns, rows, and pages (optional) in a multipanel display.
scaleType	scaleType controls if use a same scale of y-axis, choose from c('same', 'free')
stripLeft	logical variable to choose the position of strip, "TRUE" for drawing strips on the left of each panel, "FALSE" for drawing strips on the top of each panel. Used only when isPlot = 'TRUE'
stripText.cex	a number indicating the amount by which strip text in the plot(s) should be scaled relative to the default. 1=default, 1.5 is 50% larger, 0.5 is 50% smaller, etc.
axis.cex	a number indicating the amount by which axis in the plot(s) should be scaled relative to the default. 1=default, 1.5 is 50% larger, 0.5 is 50% smaller, etc.
digits	digits of printout numeric summary. Used only when isPrint = 'TRUE'
titleText	logical varible to choose display plot title or not. Default is 'TRUE', and used only when isPlot = 'TRUE'.
which	a number to indicate the type of plot. If a subset of the plots is required, specify a subset of the numbers 1:4 for plots. If which=NULL (default), the following menu appears: For plots of a group of assets: 1 = Time Series plot of portfolio returns decomposition, 2 = Time Series plot of portfolio style factors returns, 3 = Time Series plot of portfolio sector returns, 4 = Boxplot of Portfolio Factor Returns Components.
...	other graphics parameters available in tsPlotMP(time series plot only) can be passed in through the ellipses

Value

A K x 2 matrix containing mean and standard deviation of K factors

Author(s)

Douglas Martin, Lingjie Yi

Examples

```
#Load fundamental and return data
data("stocks145scores6")
dat = stocks145scores6
dat$DATE = as.yearmon(dat$DATE)
dat = dat[dat$DATE >=as.yearmon("2008-01-01")
          & dat$DATE <= as.yearmon("2012-12-31"),]

#Load long-only GMV weights for the return data
data("wtsStocks145GmvLo")
wtsStocks145GmvLo = round(wtsStocks145GmvLo,5)
```

```
#fit a fundamental factor model
# fit a fundamental factor model
fit.cross <- fitFfm(data = dat,
              exposure.vars = c("SECTOR","ROE","BP","MOM121","SIZE","VOL121",
              "EP"),date.var = "DATE", ret.var = "RETURN", asset.var = "TICKER",
              fit.method="WLS", z.score = "crossSection")

repReturn(fit.cross, wtsStocks145GmvLo, isPlot = FALSE, digits = 4)
repReturn(fit.cross, wtsStocks145GmvLo, isPrint = FALSE, isPlot = TRUE,
          which = 4)
repReturn(fit.cross, wtsStocks145GmvLo, isPrint = FALSE, isPlot = TRUE,
          which = 1, add.grid = TRUE, scaleType = 'same')
repReturn(fit.cross, wtsStocks145GmvLo, isPrint = FALSE, isPlot = TRUE,
          which = 2, add.grid = FALSE, zeroLine = TRUE, color = 'Blue',
          scaleType = 'free')
```

---

repRisk *Decompose portfolio risk into individual factor contributions and provide tabular report*

Description

Compute the factor contributions to standard deviation (SD), Value-at-Risk (VaR), Expected Tail Loss or Expected Shortfall (ES) of the return of individual asset within a portfolio return of a portfolio based on Euler's theorem, given the fitted factor model.

Usage

```
repRisk(object, ...)

## S3 method for class 'tsfm'
repRisk(object, weights = NULL, risk = c("Sd", "VaR", "ES"),
  decomp = c("FPCR", "FCR", "FMCR"), digits = NULL, invert = FALSE,
  nrowPrint = 20, p = 0.05, type = c("np", "normal"),
  use = "pairwise.complete.obs", sliceby = c("factor", "asset"),
  isPrint = TRUE, isPlot = FALSE, layout = NULL, stripText.cex = 1,
  axis.cex = 1, portfolio.only = FALSE, ...)

## S3 method for class 'ffm'
repRisk(object, weights = NULL, risk = c("Sd", "VaR", "ES"),
  decomp = c("FMCR", "FCR", "FPCR"), digits = NULL, invert = FALSE,
  nrowPrint = 20, p = 0.05, type = c("np", "normal"),
  sliceby = c("factor", "asset", "riskType"), isPrint = TRUE,
  isPlot = FALSE, layout = NULL, stripText.cex = 1, axis.cex = 1,
  portfolio.only = FALSE, ...)
```

Arguments

object	fit object of class `tsfm`, or `ffm`.
...	other optional arguments passed to `quantile` and optional arguments passed to `cov`

weights	a vector of weights of the assets in the portfolio, names of the vector should match with asset names. Default is NULL, in which case an equal weights will be used.
risk	one of 'Sd' (standard deviation), 'VaR' (Value-at-Risk) or 'ES' (Expected Tail Loss or Expected Shortfall for calculating risk decompositon. Default is 'Sd'
decomp	one of 'FMCR' (factor marginal contribution to risk), 'FCR' 'factor contribution to risk' or 'FPCR' (factor percent contribution to risk).
digits	digits of number in the resulting table. Default is NULL, in which case digtis = 3 will be used for decomp = ('FMCR', 'FCR'), digits = 1 will be used for decomp = 'FPCR'. Used only when isPrint = 'TRUE'
invert	a logical variable to change VaR/ES to positive number, default is False and will return positive values.
nrowPrint	a numerical value deciding number of assets/portfolio in result vector/table to print or plot
p	tail probability for calculation. Default is 0.05.
type	one of "np" (non-parametric) or "normal" for calculating VaR & Es. Default is "np".
use	an optional character string giving a method for computing factor 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".
sliceby	one of 'factor' (slice/condition by factor) or 'asset' (slice/condition by asset) or 'riskType' Used only when isPlot = 'TRUE'
isPrint	logical variable to print numeric output or not.
isPlot	logical variable to generate plot or not.
layout	layout is a numeric vector of length 2 or 3 giving the number of columns, rows, and pages (optional) in a multipanel display.
stripText.cex	a number indicating the amount by which strip text in the plot(s) should be scaled relative to the default. 1=default, 1.5 is 50% larger, 0.5 is 50% smaller, etc.
axis.cex	a number indicating the amount by which axis in the plot(s) should be scaled relative to the default. 1=default, 1.5 is 50% larger, 0.5 is 50% smaller, etc.
portfolio.only	logical variable to choose if to calculate portfolio only decomposition, in which case multiple risk measures are allowed.

Value

A table containing

decomp = 'FMCR'	(N + 1) * (K + 1) matrix of marginal contributions to risk of portfolio return as well assets return, with first row of values for the portfolio and the remaining rows for the assets in the portfolio, with (K + 1) columns containing values for the K risk factors and the residual respectively
decomp = 'FCR'	(N + 1) * (K + 2) matrix of component contributions to risk of portfolio return as well assets return, with first row of values for the portfolio and the remaining rows for the assets in the portfolio, with first column containing portfolio and asset risk values and remaining (K + 1) columns containing values for the K risk factors and the residual respectively

decomp = 'FPCR'
: (N + 1) * (K + 1) matrix of percentage component contributions to risk of portfolio return as well assets return, with first row of values for the portfolio and the remaining rows for the assets in the portfolio, with (K + 1) columns containing values for the K risk factors and the residual respectively

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

Author(s)

Douglas Martin, Lingjie Yi

See Also

fitTsfm, fitFfm for the different factor model fitting functions.

Examples

```
# Time Series Factor Model
data(managers)
fit.macro <- factorAnalytics::fitTsfm(asset.names=colnames(managers[,(1:6)]),
                    factor.names=colnames(managers[,(7:9)]),
                    rf.name=colnames(managers[,10]), data=managers)
report <- repRisk(fit.macro, risk = "ES", decomp = 'FPCR',
                  nrowPrint = 10)
report

# plot
repRisk(fit.macro, risk = "ES", decomp = 'FPCR', isPrint = FALSE,
        isPlot = TRUE)

# Fundamental Factor Model
data("stocks145scores6")
dat = stocks145scores6
dat$DATE = as.yearmon(dat$DATE)
dat = dat[dat$DATE >=as.yearmon("2008-01-01") &
          dat$DATE <= as.yearmon("2012-12-31"),]

# Load long-only GMV weights for the return data
data("wtsStocks145GmvLo")
wtsStocks145GmvLo = round(wtsStocks145GmvLo,5)

# fit a fundamental factor model
fit.cross <- fitFfm(data = dat,
              exposure.vars = c("SECTOR","ROE","BP","MOM121","SIZE","VOL121",
              "EP"),date.var = "DATE", ret.var = "RETURN", asset.var = "TICKER",
              fit.method="WLS", z.score = "crossSection")
repRisk(fit.cross, risk = "Sd", decomp = 'FCR', nrowPrint = 10,
        digits = 4)
# get the factor contributions of risk
repRisk(fit.cross, wtsStocks145GmvLo, risk = "Sd", decomp = 'FPCR',
        nrowPrint = 10)
# portfolio only decomposition
repRisk(fit.cross, wtsStocks145GmvLo, risk = c("VaR", "ES"), decomp = 'FPCR',
        portfolio.only = TRUE)
# plot
repRisk(fit.cross, wtsStocks145GmvLo, risk = "Sd", decomp = 'FPCR',
```

```
isPrint = FALSE, nrowPrint = 15, isPlot = TRUE, layout = c(4,2))
```

`riskDecomp` *Decompose Risk into individual factor contributions*

Description

Compute the factor contributions to Sd, VaR and ES of returns based on Euler's theorem, given the fitted factor model.

Usage

```
riskDecomp(object, ...)

## S3 method for class 'tsfm'
riskDecomp(object, risk, weights = NULL, portDecomp = TRUE,
  p = 0.05, type = c("np", "normal"), factor.cov, invert = FALSE,
  use = "pairwise.complete.obs", ...)

## S3 method for class 'ffm'
riskDecomp(object, risk, weights = NULL, portDecomp = TRUE,
  factor.cov, p = 0.05, type = c("np", "normal"), invert = FALSE, ...)
```

Arguments

`object`	fit object of class `tsfm`, or `ffm`.
`...`	other optional arguments passed to `quantile` and optional arguments passed to `cov`
`risk`	one of "Sd" (Standard Deviation) or "VaR" (Value at Risk) or "ES" (Expected Shortfall)
`weights`	a vector of weights of the assets in the portfolio, names of the vector should match with asset names. Default is NULL, in which case an equal weights will be used.
`portDecomp`	logical. If `True` the decomposition of risk is done for the portfolio based on the weights. Else, the decomposition of risk is done for each asset. `Default` is `TRUE`
`p`	tail probability for calculation. Default is 0.05.
`type`	one of "np" (non-parametric) or "normal" for calculating Es. Default is "np".
`factor.cov`	optional user specified factor covariance matrix with named columns; defaults to the sample covariance matrix.
`invert`	a logical variable to choose if change ES to positive number, default is False
`use`	an optional character string giving a method for computing factor 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".

Value

A list containing

`portES`	factor model ES of portfolio returns.
`mES`	length-(K + 1) vector of marginal contributions to Es.
`cES`	length-(K + 1) vector of component contributions to Es.
`pcES`	length-(K + 1) vector of percentage component contributions to Es.

Where, K is the number of factors.

Author(s)

Eric Zivot, Yi-An Chen, Sangeetha Srinivasan, Lingjie Yi and Avinash Acharya

See Also

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

`portSdDecomp` for factor model Sd decomposition. `portVaRDecomp` for factor model VaR decomposition.

Examples

```
# Time Series Factor Model
data(managers)
fit.macro <- factorAnalytics::fitTsfm(asset.names=colnames(managers[,(1:6)]),
                     factor.names=colnames(managers[,(7:9)]),
                     rf.name=colnames(managers[,10]), data=managers)
decompSd <- riskDecomp(fit.macro,risk = "Sd")
decompVaR <- riskDecomp(fit.macro,invert = TRUE, risk = "VaR")
decompES <- riskDecomp(fit.macro,invert = TRUE, risk = "ES")
# get the component contribution

# random weights
wts = runif(6)
wts = wts/sum(wts)
names(wts) <- colnames(managers)[1:6]
portSd.decomp <- riskDecomp(fit.macro, wts, portDecomp = TRUE, risk = "Sd")
portVaR.decomp <- riskDecomp(fit.macro, wts, portDecomp = TRUE, risk = "VaR")
portES.decomp <- riskDecomp(fit.macro, wts, portDecomp = TRUE, risk = "ES")

# Fundamental Factor Model
data("stocks145scores6")
dat = stocks145scores6
dat$DATE = as.yearmon(dat$DATE)
dat = dat[dat$DATE >=as.yearmon("2008-01-01") &
          dat$DATE <= as.yearmon("2012-12-31"),]

# Load long-only GMV weights for the return data
data("wtsStocks145GmvLo")
wtsStocks145GmvLo = round(wtsStocks145GmvLo,5)

# fit a fundamental factor model
fit.cross <- fitFfm(data = dat,
              exposure.vars = c("SECTOR","ROE","BP","MOM121","SIZE","VOL121",
              "EP"),date.var = "DATE", ret.var = "RETURN", asset.var = "TICKER",
```

```
                fit.method="WLS", z.score = "crossSection")

decompES = riskDecomp(fit.cross, risk = "ES")
#get the factor contributions of risk
portES.decomp = riskDecomp(fit.cross, weights = wtsStocks145GmvLo, risk = "ES", portDecomp = TRUE)
```

`riskFreeRate` *Risk-free rates*

Description

10 year US Bond yields

Usage

```
data("riskFreeRate")
```

Source

TBA

`robRiskBudget` *Simple and Robust Risk Budgeting with Expected Shortfall*

Description

This function implements the Philips and Liu method to compute an optimal set of risk budgets. It takes into account both the volatility and the tail risk of strategies to create a portfolio with a targeted level of volatility but with a lower level of tail risk than achieved by a mean-variance risk budget. One of the option it provides is the option to average all off diagonal entries in the correlation matrix, and with this option in use, it works particularly well with weakly correlated strategies, as one often finds in multi-strategy hedge funds and absolute return portfolios.

Usage

```
robRiskBudget(returns = NULL, rf = 0, ER = NULL, IR = NULL, TE = NULL,
  corMat = NULL, ES = NULL, ESMethod = c("modified", "gaussian",
  "historical"), corMatMethod = c("auto", "mcd", "pairwiseQC", "pairwiseGK"),
  targetVol = 0.02, shrink = FALSE, avgCor = FALSE, p = 0.95,
  lower = 0, upper = 1, K = 100, maxit = 50, tol = 1e-05)
```

Arguments

`returns`	A matrix with a time series of returns for each asset / strategy
`rf`	A vector with the risk free rate in each time period; could also be the returns of a common benchmark
`ER`	A vector of exogeneously derived prospective expected returns. Either IR or ER must be specified.
`IR`	A vector of exogeneously derived prospective information ratios. Either IR or ER must be specified.

TE	A vector of exogneously derived prospective volatilites (tracking errors). If it is not specified, it will be estimated from the time series of asset / strategy returns.
corMat	Exogneously derived correlation matrix. If it is not specified, it will be estimated from the time series of asset / strategy returns.
ES	The user's speculated expected shortfall. If it is not specified, it will be estimated from the time series of asset / strategy returns using one of 3 user-selectable methods.
ESMethod	The method used to compute ES (the methods are available in PerformanceAnalytics)
corMatMethod	the method used to estimate the robust correlation matrix, (the methods are available in library robust, and the default is "auto")
targetVol	The target volatility
shrink	Logical value that determines whether or not we want to shrink the IRs toward their grand mean. using James-Stein shrinkage.
avgCor	Logical value that determines if we want to set each off-diagonal element in the correlation matrix to the average of all its off-diagonal elements
p	User-specified confidence level for computing ES
lower	A numeric or vector of lower bounds $\vec{\sigma}^L$ of risk budgets. Default is 0.
upper	A numeric or vector of upper bounds $\vec{\sigma}^U$ of risk budgets. Default is 1.
K	A tuning factor for the iterative algorithm. Default is 100
maxit	A number of maximum iterations. Default is 50.
tol	An accuracy of the estimation with respect to the targeted volatility. Default is 10^-5

Details

In the absence of any constraints, mean-variance risk budgets are given in closed form in terms of the Information Ratio IR and the correlation matrix C. In general, it is not obvious to allocate Expected Shortfall between strategies (or securities) in a way that achieves a target level of Expected Shortfall for the portfolio. This algorithm finds a pragmatic middle way to include tail risk in optimization - it starts by allocating risk using volatility as the measure of risk, but then uses Expected Shortfall to modify its allocations in such a way that the Expected Shortfall of the overall portfolio is reduced.

This simple closed form solution, with the inclusion of tail risk and with a stabilized correlation matrix, works surprisingly well in practice - it does not often result in negative solutions, and obviates the need for a long-only constraint - we just round up to 0 on the few occasions when a small negative risk budget appears. However, in the general case, if we want to bound the risk budgets between upper bound and lower bound, we either have to solve a full mean-variance optimization (and ignore tail risk) or create a simulation based historical optimization that includes it. Instead, we approximate the solution using an iterative scheme that clamps each risk budget between its applicable bounds, and uses a simple heuristic to increment or decrement all risk budgets in a way that allows the process to convergence in a few (usually two) iterations

Value

robRiskBudget returns a list containing the following objects:

initialRiskBudget
: The vector of unconstrained risk budgets

`finalRiskBudget`
: The vector of final risk budgets for each asset

`iterativeRiskBudget`
: The matrix of risk budgets in each iteration columnwise

`targetVol`
: The target volatility for the portfolio

`avgCor`
: The average of all off-diagonal correlations

`ER`
: The average returns (shrunk if called for) or the user supplied expected returns

`TE`
: The estimated tracking errors or the user supplied tracking errors

`ES`
: The vector of estimated ESs or the user supplied ESs

`IR`
: The shrunk average IRs or the user supplied IRs

`modIR`
: The vector of modified IRs

Author(s)

Thomas Philips, Chindhanai Uthaisaad

References

Thomas Philips and Michael Liu. "Simple and Robust Risk Budgeting with Expected Shortfall", The Journal of Portfolio Management, Fall 2011, pp. 1-13.

Examples

```
data("RussellData")
rf = RussellData[, 16]
robRiskData = RussellData[, 1:15]

riskBudget = robRiskBudget(robRiskData, rf = rf, shrink = TRUE, avgCor = TRUE,
ESMethod = "historical", corMatMethod = "mcd")
robRiskBudget(robRiskData, shrink = TRUE, corMatMethod = "pairwiseQC", avgCor = TRUE)
robRiskBudget(robRiskData, shrink = TRUE, corMatMethod = "mcd", avgCor = TRUE, upper = 0.0123)
```

RussellData *Russell data*

Description

16 Russell data

Usage

```
data("RussellData")
```

Source

TBA

simulateARL *Simulation for thresholds of the Lindley's recursion*

Description

This function simulates thresholds of Lindley's recursion used in the function cusumActMgr.

Usage

```
simulateARL(mu, Threshold, delta, k = 3, EW_constant = 0.9,
  Fixed_Sigma = 1)
```

Arguments

mu	A numeric value that determines the information ratio we want to simulate thresholds for. No default value is set.
Threshold	A numeric value that determines the threshold for the Lindley's recursion to be updated in the recursion. No default value is set.
delta	A numeric value representing the simulation accuracy. In other words, we will simulate until k * sigma / mu < precision.
k	A numeric value that determines the level of cut-off. Default is 3.
EW_constant	A numeric value representing the ratio between the former sigma and the new sigma. The default is set to 0.9
Fixed_Sigma	The logical value representing if the sigma should be constant of not. 0 represents fixed sigma, 1 represents weighted sigma. The default is set to 1.

Details

Simulate the thresholds used in cusumActMgr

Value

simulateARL returns a vector of the following:

ARL	The average return length
s	The standard deviation of the ARLs

Author(s)

Thomas Philips, Chindhanai Uthaisaad.

References

Philips, T. K., Yashchin, E. and Stein, D. M. (2003). "Using Statistical Process Control to Monitor Active Managers", Journal of Portfolio Management, Fall 2003, pp. 86-94.

Examples

```
Lower_Threshold = 1.00
Upper_Threshold = 11.0
#Monthly mu's for monthly sigma=1, annualized IR = +0.5, 0
mu = c(0.5, 0) / sqrt(12)
Seq_M = 30:40
Thresholds = Lower_Threshold + Seq_M * (Upper_Threshold - Lower_Threshold) / 100
Threshold_upper = sapply(Thresholds, FUN = simulateARL, mu = mu[1], delta = 0.05)
Threshold_lower = sapply(Thresholds, FUN = simulateARL, mu = mu[2], delta = 0.05)
```

Stock.df *Fundamental and return data for 447 NYSE stocks*

Description

Fundamental and return data: Assets: 447 stocks listed on the NYSE Frequency: Monthly Date range: 1996-02-29 through 2003-12-31

Usage

```
data(Stock.df)
```

Format

data.frame

Details

Date variable: DATE

Stock ID: TICKER

Stock return and price variables: RETURN, PRICE

Numeric exposures: VOLUME, SHARES.OUT, MARKET.EQUITY, LTDEBT, NET.SALES, COMMON.EQUITY, NET.INCOME, STOCKHOLDERS.EQUITY, LOG.MARKETCAP, LOG.PRICE, BOOK2MARKET

Note: Numeric exposures are standardized as z-scores.

Categorical variables: GICS, GICS.INDUSTRY, GICS.SECTOR

Examples

```
data(Stock.df)
str(stock)
```

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))
```

stocks145scores6 *CRSP stocks Capital IQ scores*

Description

Contains returns for 145 stocks starting from Jan 1990 to Dec 2014 spanned across 10 Sectors-ENERGY, COSTAP, INDUS,T MATRLS, FINS, INFOTK, HEALTH, CODISC, UTILS and TELCOM along with 6 factors: ROE, BP, MOM121, SIZE, VOL121, EP

The 10 Sectors correspond to Energy, ConsumerStaples, Industrials, Materials, Financials, InformationTechnology, HealthCare, ConsumerDiscretionary, Utilities and Telecommunications respectively.

Usage

```
data("stocks145scores6")
```

Source

TBA

summary.cusumActMgr *Summarizing a cusumActMgr object*

Description

summary method for object of class cusumActMgr. Returned object is of class summary.cusumActMgr. The resulting object is fed to print.summary.cusumActMgr to print all the summarized objects.

Usage

```
## S3 method for class 'cusumActMgr'
summary(object, digits = 3, ...)

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

Arguments

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

Value

Returns an object of class summary.ffm.

Object of class summary.cusumActMgr is a list of length 10 containing:

Logarithmic Excess Returns
: The logarithmic excess returns of the fund relative to the benchmark

Annualized Moving Average
: The annualized moving average of the logarithmic excess returns

Tracking Error
: The monthly tracking error of the logarithmic excess returns

Information Ratios
: The vector of monthly information ratios

Lindley's Recursion
: The vector Lindley's recursion with a reset after the detection threshold (6.81) is passed.

Annualized Cusum IR
: The vector annualized CUSUM of the information ratios

Annualized Cusum Excess Return
: The vector annualized CUSUM of the excess returns

Excess Volatility
: Excess volatility of the fund, the benchmark and the excess return

Summary Annualized cusumIR
: The summary of annualized cusum IR

Summary Annualized Cusum Excess Returns
: The summary of annualized cusum excess returns

Author(s)

Chindhanai Uthaisaad.

Examples

```
data(cusumData)
results = cusumActMgr(portfolioName = "Parvest", benchmarkName = "RUS2500", data = cusumData)
x = summary(results)
print(x)
```

---

summary.ffm *Summarizing a fitted fundamental factor model*

Description

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

Usage

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

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

Arguments

object	an object of class ffm returned by fitFfm.
...	futher arguments passed to or from other methods.
x	an object of class summary.ffm.
digits	number of significants 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.

Note: This gives a summary of the fited factor returns at each time period. If T is large, you might prefer the more succint summary produced by print.ffm.

Value

Returns an object of class summary.ffm. The print method for class summary.ffm 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.ffm is a list of length N + 2 containing:

call	the function call to fitFfm
sum.list	list of summaries of the T fit objects (of class lm or lmRob) for each time period in the factor model.

Author(s)

Sangeetha Srinivasan & Yi-An Chen.

See Also

fitFfm, summary.lm

Examples

```
data(Stock.df)
exposure.vars <- c("BOOK2MARKET", "LOG.MARKETCAP", "GICS.SECTOR")
fit2 <- fitFfm(data=stock, asset.var="TICKER", ret.var="RETURN",
               date.var="DATE", exposure.vars=exposure.vars)

# summary of factor returns estimated in each time period
summary(fit2)

# summary of lm fit for a single period
summary(fit2$factor.fit[[1]])
```

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 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.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, 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".
...	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.
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

```
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(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

Examples

```
# load data from the database
data(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",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)
```

`tsPlotMP` *Time Series Plots*

Description

Plot time series with specific plotting parameters

Usage

```
tsPlotMP(ret, add.grid = FALSE, layout = NULL, type = "l",
  yname = "RETURNS (%)", scaleType = "free", stripLeft = TRUE,
  main = NULL, lwd = 1, stripText.cex = 1, axis.cex = 1,
  color = "black", zeroLine = TRUE)
```

Arguments

`ret`	an time series exposure/return object
`add.grid`	logical varible.If 'TRUE', type = c('l', 'g'); If 'FALSE', type = c('l')
`layout`	layout is a numeric vector of length 2 or 3 giving the number of columns, rows, and pages (optional) in a multipanel display.
`type`	character. type of the plot; 'l' denotes a line, 'p' denotes a point, and 'b' and 'o' both denote both together.deafault is 'l'.
`yname`	character or espression giving label(s) for the y-axis
`scaleType`	scaleType controls if use a same scale of y-axis, choose from c('same', 'free')
`stripLeft`	logical variable to choose the position of strip, 'TRUE' for drawing strips on the left of each panel, 'FALSE' for drawing strips on the top of each panel
`main`	Typically a character string or expression describing the main title.
`lwd`	The line width, a positive number, defaulting to 1
`stripText.cex`	a number indicating the amount by which strip text in the plot(s) should be scaled relative to the default. 1=default, 1.5 is 50% larger, 0.5 is 50% smaller, etc.

axis.cex	a number indicating the amount by which axis in the plot(s) should be scaled relative to the default. 1=default, 1.5 is 50% larger, 0.5 is 50% smaller, etc.
color	A specification for the default plotting color. Default is black.
zeroLine	logical varible to choose add a dotted horizontal line at the zero vertical distance

Author(s)

Douglas Martin, Lingjie Yi

Examples

```
#Load the data
data("stocks145scores6")
dat = stocks145scores6
returns = tapply(dat$RETURN,list(dat$DATE,dat$TICKER),I)
ret = xts(returns[,1:5],as.yearmon(rownames(returns)))

#generate return time series plot
tsPlotMP(ret, color = 'Blue')
tsPlotMP(ret, scaleType = "same", zeroLine = FALSE)
tsPlotMP(ret, stripLeft = FALSE, main = 'Time Series Plot')
```

---

vif *Factor Model Variance Inflation Factor Values*

Description

Calculate and plot the Factor Model Variance Inflaction Factor Values for a fitted model. A VIF for a single explanatory variable (style factor) is obtained using the time series of R-squared values obtained from the regression of that variable against all other explanatory variables. So, at least 2 explanatory variables are required in `exposure.vars` of fitted model to find the VIF.

Usage

```
vif(ffmObj, digits = 2, isPrint = TRUE, isPlot = TRUE, lwd = 2,
  stripText.cex = 1, axis.cex = 1, title = TRUE, ...)
```

Arguments

ffmObj	an object of class `ffm` produced by `fitFfm`
digits	an integer indicating the number of decimal places to be used for rounding. Default is 2.
isPrint	logical. if `TRUE`, the time series of the computed factor model values is printed along with their mean values. Else, only the mean values are printed. Default is `TRUE`.
isPlot	logical. if `TRUE`, the time series of the output is plotted. Default is `TRUE`.
lwd	line width relative to the default. Default is 2.

stripText.cex	a number indicating the amount by which strip text in the plot(s) should be scaled relative to the default. 1=default, 1.5 is 50% larger, 0.5 is 50% smaller, etc.
axis.cex	a number indicating the amount by which axis in the plot(s) should be scaled relative to the default. 1=default, 1.5 is 50% larger, 0.5 is 50% smaller, etc.
title	logical. This argument is mainly used for the documentation purpose when you need a plot without any title. If TRUE, the plots will have the main tiltle. default is TRUE.
...	potentially further arguments passed.

Value

ffmRsq returns the sample mean values and plots the time series of corresponding R squared values and the Variance Inflation factors depending on the values of rsq, rsqAdj and VIF. The time series of the output values are also printed if isPrint is TRUE.

Author(s)

Avinash Acharya

Examples

```
# Load the data
 data("factorDataSetDjia5Yrs")

# Fit a Ffm
require(factorAnalytics)
fit <- fitFfm(data=factorDataSetDjia5Yrs, asset.var="TICKER", ret.var="RETURN",
              date.var="DATE", exposure.vars=c("SECTOR", "P2B", "EV2S", "MKTCAP"))

# Plot and print the time series of VIF values
vif(fit,isPrint=TRUE)
```

---

wtsDjiaGmv *DJIA GMV portfolio weights*

Description

Contains weights obtained after optimizing the portfolio returns of the 30 DJIA stocks (from dataset factorDataSetDjia5Yrs) for a global minimum variance portfolio starting from Jan 2008 to Dec 2012.

Usage

```
data("wtsDjiaGmv")
```

Source

TBA

wtsDjiaGmvLo *DJIA GMV long-only portfolio weights*

Description

Contains weights obtained after optimizing the portfolio returns of the 30 DJIA stocks (from dataset factorDataSetDjia5Yrs) for a long-only global minimum variance portfolio starting from Jan 2008 to Dec 2012.

Usage

```
data("wtsDjiaGmvLo")
```

Source

TBA

wtsStocks145Gmv *CRSP 145 stocks GMV portfolio weights*

Description

Contains weights obtained after optimizing the portfolio returns of 145 stocks (from dataset stocks145scores6) for a global minimum variance portfolio starting from Jan 1990 to Dec 2014.

Usage

```
data("wtsStocks145Gmv")
```

Source

TBA

wtsStocks145GmvLo *CRSP 145 stocks GMV long-only weights*

Description

Contains weights obtained after optimizing the portfolio returns of 145 stocks (from dataset stocks145scores6) for a long-only global minimum variance portfolio starting from Jan 1990 to Dec 2014.

Usage

```
data("wtsStocks145GmvLo")
```

Source

TBA

Index