Package ‘facmodCS’

June 15, 2023

Type Package

Title Cross-Section Factor Models

Version 1.0

Date 2023-06-04

Maintainer Mido Shammaa <midoshammaa@yahoo.com>

Description Linear cross-section factor model fitting with least-squares and robust fitting the 'lmrobdetMM()' function from 'RobStatTM'; related volatility, Value at Risk and Expected Shortfall risk and performance attribution (factor-contributed vs idiosyncratic returns); tabular displays of risk and performance reports; factor model Monte Carlo. The package authors would like to thank Chicago Research on Security Prices,LLC for the cross-section of about 300 CRSP stocks data (in the data.table object 'stocksCRSP', and S&P GLOBAL MARKET INTELLIGENCE for contributing 14 factor scores (a.k.a ``alpha factors''.and ``factor exposures'') fundamental data on the 300 companies in the data.table object 'factorSPGMI'. The 'stocksCRSP' and 'factorsSPGMI' data are not covered by the GPL-2 license, are not provided as open source of any kind, and they are not to be redistributed in any form.

License GPL-2

Depends R (>= 3.5),

Imports data.table, xts, zoo, PerformanceAnalytics, lattice, methods, sn, tseries, robustbase, RobStatTM,

Suggests PCRA, corrplot, lmtest, rugarch, HH

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

RoxygenNote 7.2.3

Encoding UTF-8

NeedsCompilation no

Author Mido Shammaa [aut, cre],
Doug Martin [ctb, aut],
Kirk Li [aut, ctb],
Avinash Acharya [ctb],
Lingjie Yi [ctb]

Repository CRAN

Date/Publication 2023-06-15 17:20:08 UTC

R topics documented:

calcFLAM *calcFLAM*

Description

function to calculate fundamental law of active management

Usage

```
calcFLAM(
  specObj,
  modelStats,
  fitResults,
  analysis = c("ISM", "NEW"),
  targetedVol = 0.06,
  ...
)
```

Arguments

`specObj`	an object as the output from specFfm function
`modelStats`	output of the extractRegressionStats functions. Contains fit statistics of the factor model.
`fitResults`	output from fitFfmDT
`analysis`	type character, choice of c("none", "ISM","NEW"). Default = "none". Corresponds to methods used in the analysis of fundamental law of active management.
`targetedVol`	numeric; the targeted portfolio volatility in the analysis. Default is 0.06.
`...`	additional arguments

convert *convert*

Description

function to convert the new ffm spec object to ffm object to make it easier in plotting and reporting

Usage

```
convert(SpecObj, FitObj, RegStatsObj, ...)
```

Arguments

`SpecObj`	an object as the output from specFfm function
`FitObj`	an object as the output from fitFfmDT function
`RegStatsObj`	an object as the output from extractRegressionStats function
`...`	additional arguments

Value

returns an object of class ffm

`convert.ffmSpec` *Function to convert to current class # mido to change to retroFit*

Description

Function to convert to current class # mido to change to retroFit

Usage

```
## S3 method for class 'ffmSpec'
convert(SpecObj, FitObj, RegStatsObj, ...)
```

Arguments

`SpecObj`	an object as the output from specFfm function
`FitObj`	an object as the output from fitFfmDT function
`RegStatsObj`	an object as the output from extractRegressionStats function
`...`	additional arguments

Value

returns an object of class ffm

`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

```
# 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)
```

`extractRegressionStats` *extractRegressionStats*

Description

function to compute or Extract objects to be returned

Usage

```
extractRegressionStats(specObj, fitResults, full.resid.cov = FALSE)
```

Arguments

`specObj`	fitFM object that has been already fit
`fitResults`	output from fitFfmDT
`full.resid.cov`	an option to calculate the full residual covariance or not

Details

this function operates on the specObje data and the output of fitFfm to get information on the fundamental factor.

Value

a structure of class ffm holding all the information

See Also

`specFfm` and `fitFfmDT` for information on the definition of the specFfm object and the usage of fitFfmDT.

---

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"),
  addIntercept = FALSE,
  lagExposures = TRUE,
  resid.scaleType = "stdDev",
  lambda = 0.9,
  GARCH.params = list(omega = 0.09, alpha = 0.1, beta = 0.81),
  GARCH.MLE = FALSE,
  stdReturn = FALSE,
  analysis = c("none", "ISM", "NEW"),
  targetedVol = 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"`.
`addIntercept`	logical; If `TRUE`, intercept is added in the exposure matrix. Note, if 2 or more variables are categorical, this must be false. Default is `FALSE`.
`lagExposures`	logical; If `TRUE`, the style exposures in the exposure matrix are lagged by one time period. Default is `TRUE`,
`resid.scaleType`	character; Only valid when fit.method is set to WLS or W-Rob. The weights used in the weighted regression are estimated using sample variance, classic EWMA, robust EWMA or GARCH model. Valid values are `stdDev`, `EWMA`, `robEWMA`, or `GARCH`.Default is `stdDev` where the inverse of residual sample variances are used as the weights. If using GARCH option, make sure to install and load rugarch package.
`lambda`	lambda value to be used for the EWMA estimation of residual variances. Default is 0.9
`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`. Estimation outsourced to the rugarch package, please load it first.

`GARCH.MLE`	boolean input (TRUE|FALSE), default value = `FALSE`. This argument allows one to choose to compute GARCH parameters by maximum likelihood estimation. Estimation outsourced to the rugarch package, please load it.
`stdReturn`	logical; If `TRUE`, the returns will be standardized using GARCH(1,1) volatilities. Default is `FALSE`. Make sure to load rugarch package.
`analysis`	method used in the analysis of fundamental law of active management; one of "none", "ISM", or "NEW". Default is "none".
`targetedVol`	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 `lmrobdetMM`. "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.scaleType` argument which computes the residual variances in one of the following ways - sample variace, 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`. Make sure you have the rugarch package installed and loaded, as is merely listed as SUGGESTS.

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.scaleType` 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, corrected and expanded the functionalities and S3 methods.

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 lmrobdetMM 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	covariance matrix of the g 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	The 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.scaleType 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.

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.

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. and Martin, R. D. (2016). "The Fundamental Law of Active Management Redux", SSRN 2730434.

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

Examples

```
library(PCRA)
# load data
data(stocksCRSP)
data(factorsSPGMI)
dateRange <- c("2006-01-31","2010-12-31")
stockItems <-  c("Date", "TickerLast", "CapGroupLast", "Return",
                "Ret13WkBill","MktIndexCRSP","Sector")
                factorItems <- c("BP","Beta60M","PM12M1M")
stocks_factors <- selectCRSPandSPGMI("monthly", dateRange = dateRange,
stockItems = stockItems, factorItems = factorItems, outputType ="data.table")


# fit a fundamental factor model with style variables BP and LogMktCap

fundamental_model <- fitFfm(data = stocks_factors,
                           asset.var = "TickerLast",
                           ret.var = "Return",
                           date.var = "Date",
                           exposure.vars = c("BP", "PM12M1M")
                           )

  summary(fundamental_model)

# Fit a Fundamental Sector Factor Model with Intercept

 sector_model <- fitFfm(data = stocks_factors,
                        asset.var = "TickerLast",
                        ret.var = "Return",
                        date.var = "Date",
                        exposure.vars = c("Sector", "BP"),
                        addIntercept = TRUE)

 summary(sector_model)
```

fitFfmDT *fitFfmDT*

Description

This function fits a fundamental factor model

Usage

```
fitFfmDT(
  ffMSpecObj,
  fit.method = c("LS", "WLS", "Rob", "W-Rob"),
  resid.scaleType = c("StdDev", "EWMA", "RobustEWMA", "GARCH"),
  lambda = 0.9,
  GARCH.params = list(omega = 0.09, alpha = 0.1, beta = 0.81),
  GARCH.MLE = FALSE,
  lmrobdet.control.para.list = lmrobdet.control(),
  ...
)
```

Arguments

`ffMSpecObj`	a specFFm object
`fit.method`	method for estimating factor returns; one of "LS", "WLS" "ROB" or "W-ROB". See details. Default is "LS".
`resid.scaleType`	one of 4 choices "StdDev","EWMA","RobustEWMA", "GARCH"
`lambda`	the ewma parameter
`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`. Estimation outsourced to the rugarch package, please load it first.
`GARCH.MLE`	boolean input (TRUE|FALSE), default value = `FALSE`. This argument allows one to choose to compute GARCH parameters by maximum likelihood estimation. Estimation outsourced to the rugarch package, please load it.
`lmrobdet.control.para.list`	list of parameters to pass to lmrobdet.control(). Sets tuning parameters for the MM estimator implemented in lmrobdetMM of the RobStatTM package. See `lmrobdetMM`.
`...`	additional pass through arguments

Details

this function operates on the data inside the specObj fits a fundamental factor model to the data

Value

`fitFfm` returns a list with two object of class `"data.table"` The first reg.listDT is object of class `"data.table"` is a list containing the following components:

- `DATE` — length-Time vector of dates.
- `id` — length-N vector of asset id's for each date.
- `reg.list` — 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 `lmrobdetMM` if `fit.method="Rob"` or `"W-Rob"`.

The second betasDT is object of class `"data.table"` is a list containing the following components:

- `DATE` — length-Time vector of dates.
- `R_matrix` — The K+1 by K restriction matrix where K is the number of categorical variables for each date.

See Also

`specFfm` for information on the definition of the specFfm object.

---

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, factor.cov, ...)

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

Arguments

- `object` — fit object of class `tsfm`, `sfm` or `ffm`.
- `factor.cov` — factor covariance matrix (optional); defaults to the sample covariance matrix.
- `...` — optional arguments passed to `cov`.
- `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

`fitFfm`

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

`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 '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

`fitFfm` for the different factor model fitting functions.

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

`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

```
## Not run:
#Empirical deistribution
data("factorDataSetDjia5Yrs")
exposure.vars <-
fit.ffm <- fitFfm(data = factorDataSetDjia5Yrs,
                  asset.var = "TICKER",
                   ret.var = "RETURN",
                   date.var = "DATE",
                   exposure.vars = c("P2B", "MKTCAP", "SECTOR"),
                   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")

## End(Not run)
```

---

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,
  rsq = TRUE,
  rsqAdj = FALSE,
  plt.type = 2,
  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
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.
...	potentially further arguments passed.

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
# Fundamental Factor Model
```

```
library(PCRA)

dateRange <- c("2006-01-31","2010-12-31")
stockItems <-  c("Date", "TickerLast",  "Return","Sector")
factorItems <- c("BP","Beta60M","PM12M1M")
facDatIT <- selectCRSPandSPGMI("monthly",
                               dateRange = dateRange,
                               stockItems = stockItems,
                               factorItems = factorItems,
                               outputType = "data.table")
asset.var="TickerLast"
ret.var="Return"
date.var = "Date"
exposure.vars= factorItems
asset.var="TickerLast"
ret.var="Return"
date.var = "Date"
spec1 <- specFfm(data = facDatIT,asset.var = asset.var, ret.var = ret.var,
                date.var = date.var, exposure.vars = exposure.vars,weight.var = NULL,
                 addIntercept = TRUE, rob.stats = FALSE)
# fit a fundamental factor model
mdlFit <- fitFfmDT(spec1)
mdlRes <- extractRegressionStats(spec1,mdlFit)
fit.cross <- convert(SpecObj = spec1,FitObj = mdlFit, RegStatsObj = mdlRes)
#Calculate and plot the portfolio R-squared values
fmRsq(fit.cross)

#Plot and print the time series of Adj R-squared and VIF values
fmRsq(fit.cross, 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, factor.cov, ...)

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

Arguments

`object`	fit object of class `tsfm` or `ffm`.
`factor.cov`	optional user specified factor covariance matrix with named columns; defaults to the sample covariance matrix.
`...`	optional arguments passed to `cov`.

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

`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.

---

fmTstats	*fmTstats.ffm t-stats and plots for a fitted Fundamental Factor Model object*

Description

Calculate and plot the time series of t-statistic values and the number of risk indices with significant t-stats for a fundamental factor model of class `ffm` produced by `fitFfm` or `fitFfmDT`

Usage

```
fmTstats(
  ffmObj,
  isPlot = TRUE,
  isPrint = FALSE,
  whichPlot = "tStats",
  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`
`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 `tStats` 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	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. Default 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. Default = 1. 1.5 is 50% larger, 0.5 is 50% smaller, etc.
title	logical. if TRUE, the plots will have the main title. Default is TRUE.
...	potentially further arguments passed.

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

```
library(PCRA)
# load data
data(stocksCRSP)
data(factorsSPGMI)
dateRange <- c("2006-01-31","2010-12-31")
stockItems <-  c("Date", "TickerLast", "CapGroupLast", "Return",
                "Ret13WkBill","MktIndexCRSP","Sector")
                factorItems <- c("BP","Beta60M","PM12M1M")
stocks_factors <- selectCRSPandSPGMI("monthly", dateRange = dateRange,
stockItems = stockItems, factorItems = factorItems, outputType ="data.table")
```

```
# fit a fundamental factor model with style variables BP and LogMktCap

fundamental_model <- fitFfm(data = stocks_factors,
                            asset.var = "TickerLast",
                            ret.var = "Return",
                            date.var = "Date",
                            exposure.vars = c("BP", "PM12M1M")
                            )

#Compute time series of t-stats and number of significant t-stats
 stats = fmTstats(fundamental_model, isPlot = TRUE, lwd = 2, color = c("blue", "blue"),
                  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)
#
#  fit.MICM <- fitFfm(data = factorDataSetDjia5Yrs,
#                    asset.var = "TICKER",
#                    exposure.vars = c("SECTOR", "COUNTRY","P2B", "MKTCAP"),
#                    ret.var = "RETURN",
#                    date.var = "DATE",
#                    addIntercept = FALSE)
#
 # Load library 'HH' to access the Likert option
#  library("HH")
#  stats = fmTstats(fit.MICM, isPlot = TRUE, z.alpha =1.96,
#                   whichPlot = "significantTstatsLikert")
```

---

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 'ffm'
fmVaRDecomp(
  object,
```

```
  factor.cov,
  p = 0.05,
  type = c("np", "normal"),
  use = "pairwise.complete.obs",
  ...
)
```

Arguments

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

Details

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

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

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

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

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

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

Value

A list containing

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

`pcVaR` N x (K+1) matrix of percentage component contributions to VaR.

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

Author(s)

Eric Zivot, Yi-An Chen and Sangeetha Srinivasan

References

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

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

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

See Also

`fitFfm` for the different factor model fitting functions.

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

`lagExposures` *lagExposures allows the user to lag exposures by one time period*

Description

Function lag the style exposures in the exposure matrix by one time period.

Usage

```
lagExposures(specObj)
```

Arguments

`specObj` an ffm specification object of of class `"ffmSpec"`

Details

this function operates on the data inside the specObj and applies a lag to it

Value

specObj an ffm spec Object that has been lagged

See Also

`specFfm` for information on the definition of the specFfm object.

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,
  asset.variable,
  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`.
`asset.variable`	the name of asset variable.
`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.

Value

Does not return a value, used for plotting

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` (requires strucchange package), 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` (requires corrplot package).

---

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

Arguments

object	fit object of class 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.
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 Es. Default is "np".
invert	a logical variable to choose if change ES to positive number, default is False

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

`fitFfm` for the different factor model fitting functions.

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

---

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 '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.

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

fitFfm for the different factor model fitting functions.

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

---

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 '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

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

fitFfm for the different factor model fitting functions.

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

---

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 fiited 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 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.

Value

Returns an object of class `print.ffm`.

Author(s)

Yi-An Chen and Sangeetha Srinivasan

See Also

`fitFfm`, `summary.ffm`

Examples

```
## Not run:
library(PCRA)
data(stocksCRSP)
data("factorDataSetDjia5Yrs")
# fit a fundamental factor model
fit.style.sector <- fitFfm(data=factorDataSetDjia5Yrs,
                           asset.var="TICKER",
                           ret.var="RETURN",
                           date.var="DATE",
                           exposure.vars = c("P2B", "MKTCAP"))
print(fit.style.sector)

## End(Not run)
```

print.ffmSpec *print.ffmSpec*

Description

print.ffmSpec

Usage

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

Arguments

x	an object of class ffmSpec
...	any other option

Value

No return value, called for displaying attributes

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

---

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

```
args(repReturn)
```

`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 '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".

`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

`fitFfm` for the different factor model fitting functions.

Examples

```
# Fundamental Factor Model
library(PCRA)
```

```
dateRange <- c("2006-01-31","2010-12-31")
stockItems <-  c("Date", "TickerLast",  "Return","Sector")
factorItems <- c("BP","Beta60M","PM12M1M")
facDatIT <- selectCRSPandSPGMI("monthly",
                               dateRange = dateRange,
                               stockItems = stockItems,
                               factorItems = factorItems,
                               outputType = "data.table")
asset.var="TickerLast"
ret.var="Return"
date.var = "Date"
exposure.vars= factorItems
asset.var="TickerLast"
ret.var="Return"
date.var = "Date"
spec1 <- specFfm(data = facDatIT,asset.var = asset.var, ret.var = ret.var,
                date.var = date.var, exposure.vars = exposure.vars,weight.var = NULL,
                 addIntercept = TRUE, rob.stats = FALSE)
# fit a fundamental factor model
mdlFit <- fitFfmDT(spec1)
mdlRes <- extractRegressionStats(spec1,mdlFit)
fit.cross <- convert(SpecObj = spec1,FitObj = mdlFit, RegStatsObj = mdlRes)
repRisk(fit.cross, risk = "Sd", decomp = 'FCR', nrowPrint = 10,  digits = 4)
# get the factor contributions of risk
repRisk(fit.cross, risk = "Sd", decomp = 'FPCR',
        nrowPrint = 10)

# portfolio only decomposition
repRisk(fit.cross, risk = c("VaR", "ES"), decomp = 'FPCR',
        portfolio.only = TRUE)

# plot
repRisk(fit.cross, risk = "Sd", decomp = 'FPCR',
        isPrint = FALSE, nrowPrint = 15, isPlot = TRUE, layout = c(4,2))
```

`residualizeReturns` *residualizeReturns*

Description

#' function to Residualize the returns via regressions

Usage

```
residualizeReturns(specObj, benchmark, rfRate, isBenchExcess = FALSE)
```

Arguments

`specObj`	specObj is a ffmSpec object,
`benchmark`	we might need market returns
`rfRate`	risk free rate
`isBenchExcess`	toggle to select whether to calculate excess returns

Details

this function operates on the data inside the specObj and residualizes the returns to create residual return using regressions of returns on a benchmark.

Value

the ffmSpec object with resturns residualized

See Also

`specFfm` for information on the definition of the specFfm object.

`riskDecomp.ffm` *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.ffm(
  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.
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
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 Es. Default is "np".
invert	a logical variable to choose if change ES to positive number, default is False
...	other optional arguments passed to quantile and optional arguments passed to cov

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

fitFfm for the different factor model fitting functions.

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

roll.fitFfmDT *roll.fitFfmDT*

Description

roll.fitFfmDT rolls the fundamental factor model

Usage

```
roll.fitFfmDT(
  ffMSpecObj,
  windowSize = 60,
  refitEvery = 1,
  refitWindow = c("Expanding", "Rolling"),
  stdExposuresControl = list(Std.Type = "timeSeries", lambda = 0.9),
 stdReturnControl = list(GARCH.params = list(omega = 0.09, alpha = 0.1, beta = 0.81)),
 fitControl = list(fit.method = c("LS", "WLS", "Rob", "W-Rob"), resid.scaleType =
  c("STDDEV", "EWMA", "ROBEWMA", "GARCH"), lambda = 0.9, GARCH.params = list(omega =
   0.09, alpha = 0.1, beta = 0.81), GARCH.MLE = FALSE),
  full.resid.cov = TRUE,
  analysis = c("ISM", "NEW")
)
```

Arguments

`ffMSpecObj`	a specFFm object
`windowSize`	the size of the fit window
`refitEvery`	the frequency of fitting
`refitWindow`	choice of expanding or rolling
`stdExposuresControl`	for exposure standardization; (give the Std.Type and lambda)
`stdReturnControl`	choices to standardize the returns using GARCH controls
`fitControl`	list of options for fitting the ffm
`full.resid.cov`	True or False toggle
`analysis`	choice of "ISM" or "NEW"

Value

a list object containing a list of objects describing the fitted analysis.

specFfm *Specifies the elements of a fundamental factor model*

Description

Factor models have a few parameters that describe how the fitting is done. This function summarizes them and returns a spec object for cross-sectional regressions. It also preps the data. An object of class "ffmSpec" is returned.

Usage

```
specFfm(
  data,
  asset.var,
  ret.var,
  date.var,
  exposure.vars,
  weight.var = NULL,
  addIntercept = FALSE,
  rob.stats = FALSE
)
```

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 standardizing style factor exposures. Default is NULL. See Details.
addIntercept	logical; If TRUE, intercept is added in the exposure matrix. Default is FALSE,
rob.stats	logical; If TRUE, robust estimates of covariance, correlation, location and univariate scale are computed as appropriate (see Details). Default is FALSE.

Value

an object of class ffmSpec holding the details of the analysis

Examples

```
library(PCRA)
```

```
dateRange <- c("2006-01-31","2010-12-31")
stockItems <-  c("Date", "TickerLast",  "Return","Sector")
factorItems <- c("BP","Beta60M","PM12M1M")
facDatIT <- selectCRSPandSPGMI("monthly",
                               dateRange = dateRange,
                               stockItems = stockItems,
                               factorItems = factorItems,
                               outputType = "data.table")
asset.var="TickerLast"
ret.var="Return"
date.var = "Date"
exposure.vars= c("BP","Beta60M","PM12M1M")
spec1 <- specFfm(data = facDatIT,asset.var = asset.var, ret.var = ret.var,
                 date.var = date.var, exposure.vars = exposure.vars,weight.var = NULL,
                 addIntercept = TRUE, rob.stats = FALSE)
spec1$exposure.vars

#lag the exposures
spec1 <- lagExposures(spec1)
# standardize the exposures Cross-Sectionally
spec1 <- standardizeExposures(spec1, Std.Type = "CrossSection")
# fit the model
mdlFit <- fitFfmDT(spec1)
class(mdlFit)
class(mdlFit$reg.listDT)
```

`standardizeExposures` *standardizeExposures*

Description

function to calculate z-scores for numeric exposure using weights weight.var

Usage

```
standardizeExposures(
  specObj,
  Std.Type = c("None", "CrossSection", "TimeSeries"),
  lambda = 0.9
)
```

Arguments

`specObj`	is a ffmSpec object,
`Std.Type`	method for exposure standardization; one of "none", "CrossSection", or "TimeSeries". Default is "none".
`lambda`	lambda value to be used for the EWMA estimation of residual variances. Default is 0.9

Details

this function operates on the data inside the specObj and applies a standardization to it. The user can choose CrossSectional or timeSeries standardization

Value

the ffmSpec object with exposures z-scored

See Also

`specFfm` for information on the definition of the specFfm object.

`standardizeReturns` *standardizeReturns*

Description

Standardize the returns using GARCH(1,1) volatilities.

Usage

```
standardizeReturns(
  specObj,
  GARCH.params = list(omega = 0.09, alpha = 0.1, beta = 0.81)
)
```

Arguments

`specObj`	is a ffmSpec object
`GARCH.params`	fixed Garch(1,1) parameters

Details

this function operates on the data inside the specObj and standardizes the returns to create scaled return.

Value

an ffmSpec Object with the standardized returns added

See Also

`specFfm` for information on the definition of the specFfm object.

`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 significant digits to use when printing. Default is 3.
`labels`	option to print labels and legend in the summary. Default is `TRUE`. When `FALSE`, only the coefficient matrx with standard errors is printed.

Details

The default `summary` method for a fitted `lm` object computes the standard errors and t-statistics under the assumption of homoscedasticity.

Note: This gives a summary of the fitted factor returns at each time period. If `T` is large, you might prefer the more succinct 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

---

tsPlotMP *Time Series Plots*

Description

Plot time series with specific plotting parameters

Usage

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

Arguments

`ret`	an time series exposure/return object
`add.grid`	logical variable.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 expression giving label(s) for the y-axis
`Pct`	Pct controls if use the percentage value.
`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

`panel` function to add customized lines to the lattice plot. See examples.

Value

No return value, called for plotting

Author(s)

Douglas Martin, Lingjie Yi

`vif` *Factor Model Variance Inflaction 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

```
library(PCRA)
# load data
data(stocksCRSP)
data(factorsSPGMI)
dateRange <- c("2006-01-31","2010-12-31")
stockItems <-  c("Date", "TickerLast", "CapGroupLast", "Return",
                 "Ret13WkBill","MktIndexCRSP","Sector")
                 factorItems <- c("BP","Beta60M","PM12M1M")
stocks_factors <- selectCRSPandSPGMI("monthly", dateRange = dateRange,
stockItems = stockItems, factorItems = factorItems, outputType ="data.table")


# fit a fundamental factor model with style variables BP and LogMktCap

fundamental_model <- fitFfm(data = stocks_factors,
                            asset.var = "TickerLast",
```

```
                              ret.var = "Return",
                              date.var = "Date",
                              exposure.vars = c("BP", "PM12M1M")
                              )

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

Index