Fitting Fundamental Factor Models: factorAnalytics vignette

Sangeetha Srinivasan

March 19, 2018

Abstract

The purpose of this vignette is to demonstrate the use of `fitFfm` and related control, analysis and plot functions in the `factorAnalytics` package.

Contents

1 Overview

1.1 Load Package

The latest version of the `factorAnalytics` package can be downloaded from R-forge through the following command:

```
install.packages("factorAnalytics", repos="http://R-Forge.R-project.org")
```

Load the package and it's dependencies.

```
library(factorAnalytics)
options(digits=3)
```

1.2 Summary of related functions

Here's a list of the functions and methods demonstrated in this vignette:

- `fitFfm(data, asset.var, ret.var, date.var, exposure.vars, weight.var, fit.method, rob.stats, full.resid.cov, z.score, add.intercept, lag.exposures, resid.scale.type, lambda, GARCH.params, GARCH.MLE, std.return, analysis, target.vol, ...)`: Fits a fundamental factor model for one or more asset returns or excess returns using T cross-sectional regressions a.k.a. the "BARRA" approach (detailed in Grinold and Kahn (2000)), where T is the number of time periods. Least squares (LS), weighted least squares (WLS), robust (rob) and weighted-robust regression (W-Rob) fitting are possible. Options for computing residual variances include sample variance, EWMA, Robust EWMA and GARCH(1,1). An object of class "ffm" containing the fitted objects, factor exposures, factor returns, R-squared, residual volatility, etc. is returned.
- `coef(object, ...)`: Returns a data.frame containing the coefficients (intercept and factor exposures) for the last time period for all assets.
- `fitted(object, ...)`: Returns an "xts" data object of fitted asset returns from the factor model for all assets.
- `residuals(object, ...)`: Returns an "xts" data object of residuals from the fitted factor model for all assets.
- `fmCov(object, use, ...)`: Returns the `N x N` symmetric covariance matrix for asset returns based on the fitted factor model using exposures from the last time period.

- `fmSdDecomp(object, use, ...)`: Returns a list containing the standard deviation of asset returns based on the fitted factor model and the marginal, component and percentage component factor contributions estimated from the given sample. `"use"` specifies how missing values are to be handled.

- `fmVaRDecomp(object, p, ...)`: Returns a list containing the value-at-risk for asset returns based on the fitted factor model and the marginal, component and percentage component factor contributions estimated from the given sample. VaR computation can be non-parametric (sample quantile) or based on a Normal distribution. And, `"p"` specifies the confidence level.

- `fmEsDecomp(object, p, ...)`: Returns a list containing the expected shortfall for asset returns based on the fitted factor model and the marginal, component and percentage component factor contributions estimated from the given sample. Expected shortfall computation can be non-parametric (sample quantile) or based on a Normal distribution.

- `plot(x)`: The `plot` method for class "ffm" can be used for plotting factor model characteristics of a group of assets (default) or an individual asset. The user can select the type of plot either from the menu prompt 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`.

- `predict(object, newdata, pred.date, ...)`: The `predict` method for class "ffm" returns a vector or matrix of predicted values for a new data sample or simulated values. `pred.date` allows user to choose a date, and hence the estimated factor exposures from that date to be used in the prediction.

- `summary(object, ...)`: The `summary` method for class "ffm" returns an object of class `"summary.ffm"` containing the summaries of the fitted objects. Printing the factor model summary object outputs the call, estimated factor returns, r-squared and residual volatility for each time period.

1.3 Data

The following examples primarily use the `Stock.df` dataset. It contains fundamental and monthly return data for 447 stocks listed on the NYSE. This is a balanced dataset - every asset has a complete set of observations for variables in all time periods.

```r
# load the dataset into the environment
data(Stock.df)
# get a list of the variable names
colnames(stock)
```

```
##  [1] "DATE"              "RETURN"            "TICKER"
##  [4] "PRICE"             "VOLUME"            "SHARES.OUT"
##  [7] "MARKET.EQUITY"     "LTDEBT"            "NET.SALES"
## [10] "COMMON.EQUITY"     "NET.INCOME"        "STOCKHOLDERS.EQUITY"
## [13] "LOG.MARKETCAP"     "LOG.PRICE"         "BOOK2MARKET"
## [16] "GICS"              "GICS.INDUSTRY"     "GICS.SECTOR"
```

```r
# time period covered in the data
range(stock[,"DATE"])
```

```
## [1] "1996-02-29" "2003-12-31"
```

```r
# number of stocks
length(unique(stock[,"TICKER"]))
```

```
## [1] 447
```

```r
# count stocks by GICS sector as of the last time period
stocklist<-subset(stock,DATE=="2003-12-31")
table(stocklist$GICS.SECTOR)
```

```
##
##     Consumer Discretionary            Consumer Staples
##                         86                          30
##                     Energy                  Financials
##                         17                          55
##                Health Care                 Industrials
##                         35                          89
##     Information Technology                   Materials
##                         57                          32
## Telecommunication Services                   Utilities
##                          6                          40
```

2 Fit a fundamental factor model

A fundamental factor model uses observed cross-sectional asset characteristics such as dividend yield, earnings yield, book-to-market ratio, market capitalization, sector or industry classification, price volatility, price momemtum, leverage, etc. to determine common risk factors that contribute to asset returns. There are 2 main approaches to estimating the fundamental factor model - the "BARRA" approach (detailed in Grinold and Kahn (2000)) and the "Fama-French" approach (introduced in Fama and French (1992)). In the "BARRA" approach, the obsered fundamental attributes are the factor betas and the unknown factor returns are estimated via cross-sectional regressions for each time period. In the "Fama-French" approach, the factor returns are the observed returns of a hypothetical hedge portfolio that's long/short the top/bottom quintile of stocks for a given attribute (ex: market cap). After the factor returns are estimated for each characterictic, each asset's factor exposures are estimated via a time series regression. **fitFfm** described in this vignette uses the "BARRA" approach.

Let's take a look at the arguments for **fitFfm**.

```
args(fitFfm)

## function (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,
##     ...)
## NULL
```

The default model fitting method is LS regression and the default variable selection method is "none" (that is, all factors are included in the model). The different model fitting and variable selection options are described in sections 2.3 and 2.4.

The default for **rf.name** and **mkt.name** are NULL. If **rf.name** is not specified by the user, perhaps because the data is already in excess return form, then no risk-free rate adjustment is made. Similarly, if **mkt.name** is not specified, market-timing factors are not added to the model.

All other optional control parameters passed through the ellipsis are processed and assimilated internally by **fitFfm.control**. More on that in section 2.5.

2.1 Single Index Model

Here's an implementation of the single index model for the 6 hypothetical assets described in section 1.3 earlier. Since `rf.name` was included, excess returns are computed and used for all variables during model fitting.

```
# Single Index Model using SP500
fit.singleIndex <- fitFfm(asset.names=asset.names, factor.names="SP500.TR",
                          rf.name="US.3m.TR", data=managers)

## Error in is.data.frame(data):  object 'managers' not found
```

The resulting object, `fit.singleIndex`, has the following attributes.

```
class(fit.singleIndex)

## Error in eval(expr, envir, enclos):  object 'fit.singleIndex' not found

names(fit.singleIndex)

## Error in eval(expr, envir, enclos):  object 'fit.singleIndex' not found
```

The component `asset.fit` contains a list of "lm" objects[1], one for each asset. The estimated coefficients[2] are in `alpha` and `beta`. R-squared and residual standard deviations are in `r2` and `resid.sd` respectively. The remaining components contain the input choices and the data.

```
fit.singleIndex # print the fitted "tsfm" object

## Error in eval(expr, envir, enclos):  object 'fit.singleIndex' not found
```

Figure 1 shows the single factor linear fits for the assets. (Plot options are explained later in section 4.)

```
# plot asset returns vs factor returns for the single factor models
plot(fit.singleIndex, which=12, f.sub=1)

## Error in plot(fit.singleIndex, which = 12, f.sub = 1):  object 'fit.singleIndex'
not found
```

[1]The fitted objects can be of class "lm", "lmRob" or "lars" depending on the fit and variable selection methods.
[2]Refer to the summary method in section 2.6 for standard errors, degrees of freedom, t-statistics etc.

2.2 Market Timing Models

In the following example, we fit the Henriksson and Merton (1981) market timing model, using the SP500 as the market. Market timing accounts for the price movement of the general stock market relative to fixed income securities. The function `fitFfmMT`, a wrapper to `fitFfm`, includes $down.market = max(0, R_f - R_m)$ as a factor. To test market timing ability, this factor can be added to the single index model as shown below. The coefficient of this down-market factor can be interpreted as the number of "free" put options on the market provided by the manager's market-timings kills. That is, a negative value for the regression estimate would imply a negative value for market timing ability of the manager.

```
# Henriksson-Merton's market timing model
fit.mktTiming <-  fitFfmMT(asset.names=asset.names, mkt.name="SP500.TR",
                           rf.name="US.3m.TR", data=managers)

## Error in eval(expr, envir, enclos):  could not find function "fitFfmMT"

t(fit.mktTiming$beta)

## Error in t(fit.mktTiming$beta):  object 'fit.mktTiming' not found

fit.mktTiming$r2

## Error in eval(expr, envir, enclos):  object 'fit.mktTiming' not found

fit.mktTiming$resid.sd

## Error in eval(expr, envir, enclos):  object 'fit.mktTiming' not found
```

Note that, the user needs to specify which of the columns in `data` corresponds to the market returns using argument `mkt.name`.

2.3 Fit methods

The default fit method is LS regression. The next example performs LS regression using all 3 available factors in the dataset. Notice that the R-squared values have improved considerably when compared to the single index model as well as the market-timing model.

```
fit.ols <- fitFfm(asset.names=asset.names, factor.names=factor.names,
                  rf.name="US.3m.TR", data=managers)

## Error in is.data.frame(data):  object 'managers' not found
```

```r
fit.ols$beta

## Error in eval(expr, envir, enclos):  object 'fit.ols' not found

fit.ols$r2

## Error in eval(expr, envir, enclos):  object 'fit.ols' not found

fit.ols$resid.sd

## Error in eval(expr, envir, enclos):  object 'fit.ols' not found
```

Other options include discounted least squares ("DLS") and robust regression ("Robust"). DLS is least squares regression using exponentially discounted weights and accounts for time variation in coefficients. Robust regression is resistant to outliers.

```r
fit.robust <- fitFfm(asset.names=asset.names, factor.names=factor.names,
                      rf.name="US.3m.TR", data=managers, fit.method="Robust")

## Error in is.data.frame(data):  object 'managers' not found

fit.robust$beta

## Error in eval(expr, envir, enclos):  object 'fit.robust' not found

fit.robust$r2

## Error in eval(expr, envir, enclos):  object 'fit.robust' not found

fit.robust$resid.sd

## Error in eval(expr, envir, enclos):  object 'fit.robust' not found
```

Notice the lower R-squared values and smaller residual volatilities with robust regression. Figures 2 and 3 give a graphical comparison of the fitted returns for asset "HAM3" and residual volatilities from the factor model fits. Figure 2 depicts the smaller influence that the volatility of Jan 2000 has on the robust regression.

```r
par(mfrow=c(2,1))
plot(fit.ols, plot.single=TRUE, which=1, asset.name="HAM3")

## Error in plot(fit.ols, plot.single = TRUE, which = 1, asset.name = "HAM3"):  object
'fit.ols' not found
```

```
mtext("LS", side=3)

## Error in mtext("LS", side = 3):  plot.new has not been called yet

plot(fit.robust, plot.single=TRUE, which=1, asset.name="HAM3")

## Error in plot(fit.robust, plot.single = TRUE, which = 1, asset.name = "HAM3"):  object
'fit.robust' not found

mtext("Robust", side=3)

## Error in mtext("Robust", side = 3):  plot.new has not been called yet
```

```
par(mfrow=c(1,2))
plot(fit.ols, which=5, xlim=c(0,0.045), sub="LS")

## Error in plot(fit.ols, which = 5, xlim = c(0, 0.045), sub = "LS"):  object 'fit.ols'
not found

plot(fit.robust, which=5, xlim=c(0,0.045), sub="Robust")

## Error in plot(fit.robust, which = 5, xlim = c(0, 0.045), sub = "Robust"):  object
'fit.robust' not found
```

2.4 Variable Selection

Though the R-squared values improved by adding more factors in fit.ols (compared to the single index model), one might prefer to employ variable selection methods such as "stepwise", "subsets" or "lars" to avoid over-fitting. The method can be selected via the `variable.selection` argument. The default "none", uses all the factors and performs no variable selection.

Specifying "stepwise" selects traditional stepwise[3] LS or robust regression using `step` or `step.lmRob` respectively. Starting from the given initial set of factors, factors are added (or subtracted) only if the regression fit, as measured by the Bayesian Information Criterion (BIC) or Akaike Information Criterion (AIC)[4], improves.

Specifying "subsets" enables subsets selection using `regsubsets`. The best performing subset of any given size or within a range of subset sizes is chosen. Different methods such as exhaustive search (default), forward or backward stepwise, or sequential replacement can be employed.

Finally, "lars" corresponds to least angle regression using `lars` with variants "lasso" (default), "lar", "forward.stagewise" or "stepwise".

The next example uses the `"lars"` variable selection method. The default type and criterion used are `"lasso"` and the `"Cp"` statistic.

```
fit.lars <- fitFfm(asset.names=asset.names, factor.names=factor.names,
                   data=managers, rf.name="US.3m.TR",
                   variable.selection="lars")

## Error in is.data.frame(data): object 'managers' not found

fit.lars

## Error in eval(expr, envir, enclos): object 'fit.lars' not found
```

[3] The direction for stepwise search can be one of "forward", "backward" or "both". See the help file for more details.

[4] AIC is the default. When the additive constant can be chosen so that AIC is equal to Mallows' Cp, this is done. The optional control parameter `k` can be used to switch to BIC instead.

Using the same set of factors for comparison, let's fit another model using the "subsets" variable selection method. Here, the best subset of size 2 for each asset is chosen by specifying $nvmin = nvmax = 2$. Note that when $nvmin < nvmax$, the best subset is chosen from a range of subset sizes $[nvmin, nvmax]$. Default is $nvmin = 1$.

```
(fit.sub <- fitFfm(asset.names=asset.names, factor.names=factor.names,
                   data=managers, rf.name="US.3m.TR",
                   variable.selection="subsets", nvmin=2, nvmax=2))

## Error in is.data.frame(data): object 'managers' not found
```

Comparing the coefficients and R-squared values from the two models, we find that the method that uses more factors for an asset have higher R-squared values as expected. However, when both "lars" and "subsets" chose the same number of factors, "lars" fits have a slightly higher R-squared values.

The Figures 4 and 5 display the factor betas from the two fits.

```
plot(fit.sub, which=2, f.sub=1:3)

## Error in plot(fit.sub, which = 2, f.sub = 1:3): object 'fit.sub' not found
```

```
plot(fit.lars, which=2, f.sub=1:3)

## Error in plot(fit.lars, which = 2, f.sub = 1:3): object 'fit.lars' not found
```

Remarks:

- Variable selection methods "stepwise" and "subsets" can be combined with any of the fit methods, "LS", "DLS" or "Robust". If variable selection method selected is "lars", `fit.method` will be ignored.
- Refer to the next section on `fitFfm control` for more details on the control arguments that can be passed to the different variable selection methods.

2.5 fitFfm control

Since `fitFfm` calls many different regression fitting and variable selection methods, it made sense to collect all the optional controls for these functions and process them via `fitFfm.control`. This function is meant to be used internally by `fitFfm` when arguments are passed to it via the ellipsis.

The use of control parameters was demonstrated with `nvmax` and `nvmin` in the fit.sub example earlier.

For easy reference, here's a classified list of control parameters accepted and passed by `fitFfm` to their respective model fitting (or) model selection functions in other packages. See the corresponding help files for more details on each parameter.

- `lm`: "weights","model","x","y","qr"
- `lmRob`: "weights","model","x","y","nrep","efficiency","mxr","mxf","mxs","trace"
- `step`: "scope","scale","direction","trace","steps","k"
- `regsubsets`: "weights","nvmax","force.in","force.out","method","really.big"
- `lars`: "type","normalize","eps","max.steps","trace"
- `cv.lars`: "K","type","normalize","eps","max.steps","trace"

There are 3 other significant arguments that can be passed through the `...` argument to `fitFfm`.

- `decay`: Determines the decay factor for DLS fit method, which corresponds to exponentially weighted least squares, with weights adding to unity.
- `nvmin`: The lower limit for the range of subset sizes from which the best model (BIC) is found when performing "subsets" selection. Note that the upper limit was already passed to `regsubsets` function. By specifying `nvmin=nvmax`, users can obtain the best model of a particular size (meaningful to those who want a parsimonious model, or to compare with a different model of the same size, or perhaps to avoid over-fitting/ data dredging etc.).
- `lars.criterion`: An option (one of "Cp" or "cv") to assess model selection for the `"lars"` variable selection method. "Cp" is Mallow's Cp statistic and "cv" is K-fold cross-validated mean squared prediction error.

2.6 S3 generic methods

```
methods(class="tsfm")

##  [1] coef          fitted        fmCov         fmEsDecomp    fmSdDecomp
##  [6] fmVaRDecomp   plot          portEsDecomp  portSdDecomp  portVaRDecomp
## [11] portVolDecomp predict       print         repRisk       residuals
## [16] riskDecomp    summary
## see '?methods' for accessing help and source code
```

Many useful generic accessor functions are available for "tsfm" fit objects. `coef()` returns a matrix of estimated model coefficients including the intercept. `fitted()` returns an xts data object of the component of asset returns explained by the factor model. `residuals()` returns an xts data object with the component of asset returns not explained by the factor model. `predict()` uses the fitted factor model to estimate asset returns given a set of new or simulated factor return data.

`summary()` prints standard errors and t-statistics for all estimated coefficients in addition to R-squared values and residual volatilities. Argument `se.type`, one of "Default", "HC" or "HAC", allows for heteroskedasticity and auto-correlation consistent estimates and standard errors whenever possible. A "summary.tsfm" object is returned which contains a list of summary objects returned by "lm", "lm.Rob" or "lars" for each asset fit.

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

Factor model covariance and risk decomposition functions are explained in section 3 and the `plot` method is discussed separately in Section 4.

Here are some examples using the time series factor models fitted earlier.

```
# all estimated coefficients from the LS fit using all 3 factors
coef(fit.ols)

## Error in coef(fit.ols): object 'fit.ols' not found

# compare returns data with fitted and residual values for HAM1 from fit.lars
HAM1.ts <- merge(fit.lars$data[,1], fitted(fit.lars)[,1], residuals(fit.lars)[,1])

## Error in merge(fit.lars$data[, 1], fitted(fit.lars)[, 1], residuals(fit.lars)[,
: object 'fit.lars' not found

colnames(HAM1.ts) <- c("HAM1.return","HAM1.fitted","HAM1.residual")
```

```
## Error in colnames(HAM1.ts) <- c("HAM1.return", "HAM1.fitted", "HAM1.residual"):
object 'HAM1.ts' not found

tail(HAM1.ts)

## Error in tail(HAM1.ts):  object 'HAM1.ts' not found

# summary for fit.sub computing HAC standard erros
summary(fit.sub, se.type="HAC")

## Error in summary(fit.sub, se.type = "HAC"):  object 'fit.sub' not found
```

3 Factor Model Covariance & Risk Decomposition

3.1 Factor model covariance

Following Zivot and Jia-hui (2006), $R_{i,t}$, the return on asset i ($i = 1, ..., N$) at time t ($t = 1, ..., T$), is fitted with a factor model of the form,

$$R_{i,t} = \alpha_i + \boldsymbol{\beta}_i' \, \mathbf{f_t} + \epsilon_{i,t} \tag{1}$$

where, α_i is the intercept, $\mathbf{f_t}$ is a $K \times 1$ vector of factor returns at time t, $\boldsymbol{\beta}_i$ is a $K \times 1$ vector of factor exposures for asset i and the error terms $\epsilon_{i,t}$ are serially uncorrelated across time and contemporaneously uncorrelated across assets so that $\epsilon_{i,t} \sim iid(0, \sigma_i^2)$. Thus, the variance of asset i's return is given by

$$var(R_{i,t}) = \boldsymbol{\beta}_i' \, var(\mathbf{f_t}) \, \boldsymbol{\beta}_i + \sigma_i^2 \tag{2}$$

And, the $N \times N$ covariance matrix of asset returns is

$$var(\mathbf{R}) = \boldsymbol{\Omega} = \mathbf{B} \, var(\mathbf{F}) \, \mathbf{B}' + \mathbf{D} \tag{3}$$

where, R is the $N \times T$ matrix of asset returns, B is the $N \times K$ matrix of factor betas, $\mathbf{F}$ is a $K \times T$ matrix of factor returns and D is a diagonal matrix with σ_i^2 along the diagonal.

`fmCov()` computes the factor model covariance from a fitted factor model. The covariance of factor returns is the sample covariance matrix by default, but the option exists for the user to specify their own. Options for handling missing observations include "pairwise.complete.obs" (default), "everything", "all.obs", "complete.obs" and "na.or.complete".

```
fmCov(fit.sub)

## Error in inherits(object, c("tsfm", "sfm", "ffm")):  object 'fit.sub' not found

# factor model return correlation plot
plot(fit.sub, which=8)

## Error in plot(fit.sub, which = 8):  object 'fit.sub' not found
```

3.2 Standard deviation decomposition

Given the factor model in equation 1, the standard deviation of the asset i's return can be decomposed as follows (based on Meucci (2007)):

$$\begin{aligned} R_{i,t} &= \alpha_i + \boldsymbol{\beta}_i' \, \mathbf{f_t} + \epsilon_{i,t} && (4) \\ &= \boldsymbol{\beta}_i^{*'} \, \mathbf{f_t^*} && (5) \end{aligned}$$

where, $\boldsymbol{\beta}_i^{*'} = (\boldsymbol{\beta}_i' \, \sigma_i)$ and $\mathbf{f_t}^{*'} = (\mathbf{f_t}' \, z_t)$, with $z_t \sim iid(0,1)$ and σ_i is asset i's residual standard deviation.

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

$$Sd.fm_i = \sum_{k=1}^{K+1} cSd_{i,k} = \sum_{k=1}^{K+1} \beta_{i,k}^* \, mSd_{i,k} \tag{6}$$

where, summation is across the K factors and the residual, $\mathbf{cSd_i}$ and $\mathbf{mSd_i}$ are the component and marginal contributions to $Sd.fm_i$ respectively. Computing $Sd.fm_i$ and $\mathbf{mSd_i}$ is very straight forward. The formulas are given below and details are in Meucci (2007). The covariance term is approximated by the sample covariance and $\odot$ represents element-wise multiplication.

$$Sd.fm_i = \sqrt{\boldsymbol{\beta}_i^{*'} \, cov(\mathbf{F}^*) \, \boldsymbol{\beta}_i^*} \tag{7}$$

$$\mathbf{mSd_i} = \frac{cov(\mathbf{F}^*) \, \boldsymbol{\beta}_i^*}{Sd.fm_i} \tag{8}$$

$$\mathbf{cSd_i} = \boldsymbol{\beta}_i^* \odot \mathbf{mSd_i} \tag{9}$$

`fmSdDecomp` performs this decomposition for all assets in the given factor model fit object as shown below. The total standard deviation and component, marginal and percentage component contributions for each asset are returned.

```
decomp <- fmSdDecomp(fit.sub)

## Error in inherits(object, c("tsfm", "sfm", "ffm")):  object 'fit.sub' not found

names(decomp)

## Error in eval(expr, envir, enclos):  object 'decomp' not found

# get the factor model standard deviation for all assets
decomp$Sd.fm

## Error in eval(expr, envir, enclos):  object 'decomp' not found

# get the component contributions to Sd
decomp$cSd

## Error in eval(expr, envir, enclos):  object 'decomp' not found

# get the marginal factor contributions to Sd
decomp$mSd

## Error in eval(expr, envir, enclos):  object 'decomp' not found
```

```
# get the percentage component contributions to Sd
decomp$pcSd

## Error in eval(expr, envir, enclos):  object 'decomp' not found

# plot the percentage component contributions to Sd
plot(fit.sub, which=9, f.sub=1:3)

## Error in plot(fit.sub, which = 9, f.sub = 1:3):  object 'fit.sub' not found
```

3.3 Value-at-Risk decomposition

The VaR version of equation 6 is given below. By Euler's theorem, the value-at-risk of asset i's return is:

$$VaR.fm_i = \sum_{k=1}^{K+1} cVaR_{i,k} = \sum_{k=1}^{K+1} \beta^*_{i,k}\, mVaR_{i,k} \tag{10}$$

The marginal contribution to $VaR.fm$ is defined as the expectation of $F.star$, conditional on the loss being equal to $VaR.fm$. This is approximated as described in Epperlein and Smillie (2006) using a triangular smoothing kernel. $VaR.fm$ is calculated as the sample quantile.

`fmVaRDecomp` performs this decomposition for all assets in the given factor model fit object as shown below.

```
decomp1 <- fmVaRDecomp(fit.sub)

## Error in inherits(object, c("tsfm", "sfm", "ffm")):  object 'fit.sub' not found

names(decomp1)

## Error in eval(expr, envir, enclos):  object 'decomp1' not found

# get the factor model value-at-risk for all assets
decomp1$VaR.fm

## Error in eval(expr, envir, enclos):  object 'decomp1' not found

# get the percentage component contributions to VaR
decomp1$pcVaR

## Error in eval(expr, envir, enclos):  object 'decomp1' not found

# plot the percentage component contributions to VaR
plot(fit.sub, which=11, f.sub=1:3)

## Error in plot(fit.sub, which = 11, f.sub = 1:3):  object 'fit.sub' not found
```

3.4 Expected Shortfall decomposition

The Expected Shortfall (ES) version of equation 6 is given below. By Euler's theorem, the expected shortfall of asset i's return is:

$$ES.fm_i = \sum_{k=1}^{K+1} cES_{i,k} = \sum_{k=1}^{K+1} \beta^*_{i,k}\, mES_{i,k} \tag{11}$$

The marginal contribution to $ES.fm$ is defined as the expectation 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. Once again, $VaR.fm$ is the sample quantile.

`fmEsDecomp` performs this decomposition for all assets in the given factor model fit object as shown below.

```
decomp2 <- fmEsDecomp(fit.sub, method="historical")

## Error in inherits(object, c("tsfm", "sfm", "ffm")):  object 'fit.sub' not found

names(decomp2)

## Error in eval(expr, envir, enclos):  object 'decomp2' not found

# get the factor model expected shortfall for all assets
decomp2$ES.fm

## Error in eval(expr, envir, enclos):  object 'decomp2' not found

# get the component contributions to Sd
decomp2$cES

## Error in eval(expr, envir, enclos):  object 'decomp2' not found

# get the marginal factor contributions to ES
decomp2$mES

## Error in eval(expr, envir, enclos):  object 'decomp2' not found

# get the percentage component contributions to ES
decomp2$pcES

## Error in eval(expr, envir, enclos):  object 'decomp2' not found

# plot the percentage component contributions to ES
plot(fit.sub, which=10, f.sub=1:3)

## Error in plot(fit.sub, which = 10, f.sub = 1:3):  object 'fit.sub' not found
```

4 Plot

Some types of individual asset (Figure 2) and group plots (Figures 1, 3-9) have already been demonstrated. Let's take a look at all available arguments for plotting a "tsfm" object.

```
## 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, ...)
```

4.1 Group plots

This is the default option for plotting. Simply running `plot(fit)`, where `fit` is any "tsfm" object, will bring up a menu (shown below) for group plots.

```
plot(fit.sub)

# Make a plot selection (or 0 to exit):

#  1: Factor model coefficients: Alpha
#  2: Factor model coefficients: Betas
#  3: Actual and Fitted asset returns
#  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)
#
# Selection:
```

Remarks:

- Only a subset of assets and factors selected by `a.sub` and `f.sub` are plotted. The first 2 factors and first 6 assets are shown by default.
- The last option for plotting asset returns vs. factor returns (group plot option 12 and individual asset plot option 19) are only applicable for single factor models.

```
# Examples of group plots: looping disabled & no. of assets displayed = 4.
plot(fit.sub, which=3, a.sub=1:4, legend.loc=NULL, lwd=1)

## Error in plot(fit.sub, which = 3, a.sub = 1:4, legend.loc = NULL, lwd = 1):  object
'fit.sub' not found
```

```
plot(fit.sub, which=6) # residual scatter plot matrix with correlations

## Error in plot(fit.sub, which = 6):  object 'fit.sub' not found
```

4.2 Menu and looping

If the plot type argument `which` is not specified, a menu prompts for user input. 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`.

4.3 Individual plots

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

Here's the individual plot menu.

```
plot(fit.sub, plot.single=TRUE, asset.name="HAM1")


# Make a plot selection (or 0 to exit):
#  1: Actual and fitted asset returns
#  2: Actual vs fitted asset returns
#  3: Residuals vs fitted asset returns
#  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)
#
# Selection:
```

Remarks:

- 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"`.

Here are a few more examples which don't need interactive user input.

```
plot(fit.sub, plot.single=TRUE, asset.name="HAM1", which=5, ylim=c(-0.06,0.06))

## Error in plot(fit.sub, plot.single = TRUE, asset.name = "HAM1", which = 5, :  object
'fit.sub' not found
```

```
plot(fit.sub, plot.single=TRUE, asset.name="HAM1", which=10)

## Error in plot(fit.sub, plot.single = TRUE, asset.name = "HAM1", which = 10):  object
'fit.sub' not found
```

```
plot(fit.sub, plot.single=TRUE, asset.name="HAM1", which=14)

## Error in plot(fit.sub, plot.single = TRUE, asset.name = "HAM1", which = 14):  object
'fit.sub' not found

grid()

## Error in int_abline(a = a, b = b, h = h, v = v, untf = untf, ...):  plot.new has
not been called yet
```

```
plot(fit.sub, plot.single=TRUE, asset.name="HAM1", which=11)

## Error in plot(fit.sub, plot.single = TRUE, asset.name = "HAM1", which = 11):  object
'fit.sub' not found
```

```
plot(fit.sub, plot.single=TRUE, asset.name="HAM1", which=12)

## Error in plot(fit.sub, plot.single = TRUE, asset.name = "HAM1", which = 12):  object
'fit.sub' not found
```

References

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

E. F. Fama and K. R. French. The cross-section of expected stock returns. *the Journal of Finance*, 47(2):427–465, 1992.

R. C. Grinold and R. N. Kahn. Active portfolio management. 2000.

R. D. Henriksson and R. C. Merton. On market timing and investment performance. II. Statistical procedures for evaluating forecasting skills. *Journal of business*, pages 513–533, 1981.

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

E. Zivot and W. Jia-hui. Modeling Financial Time Series with S-Plus Springer-Verlag. 2006.