Fitting Time Series Factor Models: factorAnalytics vignette

Sangeetha Srinivasan

March 20, 2015

Abstract

The purpose of this vignette is to demonstrate the use of `fitTsfm` 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:

- `fitTsfm(asset.names, factor.names, data, fit.method, variable.selection, ...)`: Fits a time series (a.k.a. macroeconomic) factor model for one or more asset returns or excess returns using time series regression. Least squares (LS), discounted least squares (DLS) and robust regression fitting are possible. Variable selection methods include "stepwise", "subsets" and "lars". An object of class "tsfm" containing the fitted objects, model coefficients, R-squared and residual volatility is returned.
- `coef(object, ...)`: Extracts the coefficient matrix (intercept and factor betas) for all assets fit by the "tsfm" object.
- `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. `"use"` specifies how missing values are to be handled.
- `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, method, invert, ...)`: 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. `"p"` and `"method"` specify the confidence level and method (one of "modified","gaussian", "historical" or "kernel") to calculate VaR. VaR is by default a positive quantity and specifying `"invert=TRUE"` allows the VaR value to be expressed as a negative quantity. These 3 arguments, `"p"`, `"method"` and `"invert"` are passed on to the `VaR` function in the `PerformanceAnalytics` package to calculate VaR.

- `fmEsDecomp(object, p, method, invert, ...)`: 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. Arguments `"p"`, `"method"` and `invert` are the same as above.

- `plot(x)`: The `plot` method for class "tsfm" 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, ...)`: The `predict` method for class "tsfm" returns a vector or matrix of predicted values for a new data sample or simulated values.

- `summary(object, se.type, ...)`: The `summary` method for class "tsfm" returns an object of class `"summary.tsfm"` containing the summaries of the fitted "lm", "lmRob" or "lars" objects and the chosen type (HC/HAC) of standard errors and t-statistics to display. Printing the factor model summary object outputs the call, coefficients (with standard errors and t-statistics), r-squared and residual volatility (under the homo-skedasticity assumption) for all assets.

1.3 Data

The following examples primarily use the `managers` dataset from the `PerformanceAnalytics` package. It's an "xts" data object with 132 observations on 10 variables; frequency is monthly.

```
data(managers)
colnames(managers)

##  [1] "HAM1"        "HAM2"        "HAM3"        "HAM4"        "HAM5"
```

```
##  [6] "HAM6"        "EDHEC.LS.EQ" "SP500.TR"    "US.10Y.TR"   "US.3m.TR"

range(index(managers))

## [1] "1996-01-31" "2006-12-31"
```

In the examples below, the monthly returns for the six hypothetical asset managers (HAM1 through HAM6) will be the explained asset returns. Columns 7 through 9, composed of the EDHEC Long-Short Equity hedge fund index, the S&P 500 total returns, and the total return series for the US Treasury 10-year bond will serve as explanatory factors. The last column (US 3-month T-bill) can be considered as the risk free rate. The series have unequal histories in this sample and `fitTsfm` removes asset-wise incomplete cases (asset's return data combined with respective factors' return data) before fitting a factor model.

```
asset.names <- colnames(managers[,1:6])
factor.names <- colnames(managers[,7:9])
mkt.name <- "SP500.TR"
rf.name <- "US.3m.TR"
```

Typically, factor models are fit using excess returns. If the asset and factor returns are not in excess return form, `rf.name` can be specified to convert returns into excess returns. Similarly, market returns can be specified via `mkt.name` to add market-timing factors to the factor model.

The `CommonFactors` dataset in the `factorAnalytics` package also provides a collection of common factors as both monthly (`factors.M`) and quarterly (`factors.Q`) time series. Refer to the help file for the dataset for more information.

```
data(CommonFactors)
names(factors.Q)

## [1] "SP500"         "GS10TR"        "USD.Index"     "Term.Spread"
## [5] "Credit.Spread" "dVIX"          "TED.Spread"    "OILPRICE"
## [9] "TB3MS"

range(index(factors.Q))

## [1] "1997-03-31" "2014-03-31"
```

2 Fit a time series factor model

In a time series or macroeconomic factor model, observable economic time series such as industrial production growth rate, interest rates, market returns and inflation are used as common factors that contribute to asset returns. For example, the famous single index model by Sharpe (1964) uses the market excess return as the common factor (captures economy-wide or market risk) for all assets and the unexplained returns in the error term represents the non-market firm specific risk. On the other hand, Chen et al. (1986) uses a multi-factor model to find that surprise inflation, the spread between long and short-term interest rates and between high and low grade bonds are significantly priced, while the market portfolio, aggregate consumption risk and oil price risk are not priced separately.

Let's take a look at the arguments for `fitTsfm`.

```
args(fitTsfm)

## function (asset.names, factor.names, mkt.name = NULL, rf.name = NULL,
##     data = data, fit.method = c("LS", "DLS", "Robust"), variable.selection = c("none",
##         "stepwise", "subsets", "lars"), control = fitTsfm.control(...),
##     ...)
## 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 `fitTsfm.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 <- fitTsfm(asset.names=asset.names, factor.names="SP500.TR",
                           rf.name="US.3m.TR", data=managers)
```

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

```
class(fit.singleIndex)

## [1] "tsfm"

names(fit.singleIndex)

##  [1] "asset.fit"          "alpha"              "beta"
##  [4] "r2"                 "resid.sd"           "call"
##  [7] "data"               "asset.names"        "factor.names"
## [10] "mkt.name"           "fit.method"         "variable.selection"
```

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

##
## Call:
## fitTsfm(asset.names = asset.names, factor.names = "SP500.TR",
##     rf.name = "US.3m.TR", data = managers)
##
## Model dimensions:
## Factors  Assets Periods
##       1       6     132
##
## Regression Alphas:
##                HAM1    HAM2    HAM3    HAM4    HAM5    HAM6
## (Intercept) 0.00577 0.00909 0.00622 0.00403 0.00173 0.00784
##
```

[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.

```
## Factor Betas:
##          HAM1  HAM2  HAM3  HAM4  HAM5  HAM6
## SP500.TR 0.39 0.338 0.552 0.691 0.321 0.324
##
## R-squared values:
##   HAM1   HAM2   HAM3   HAM4   HAM5   HAM6
## 0.4339 0.1673 0.4341 0.3148 0.0829 0.2601
##
## Residual Volatilities:
##   HAM1   HAM2   HAM3   HAM4   HAM5   HAM6
## 0.0193 0.0334 0.0274 0.0443 0.0441 0.0206
```

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. Specifying `mkt.timing="HM"`, 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 <- fitTsfm(asset.names=asset.names, factor.names="SP500.TR",
                         rf.name="US.3m.TR", mkt.name="SP500.TR",
                         mkt.timing="HM", data=managers)

## Error in fitTsfm.control(...):  unused argument (mkt.timing = "HM")

fit.mktTiming$beta

## Error in eval(expr, envir, enclos):  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
```

Similarly, to account for market timing with respect to volatility, one can specify `mkt.timing="TM"`. Following Treynor and Mazuy (1966), $market.sqd = (R_m - R_f)^2$ is added as a factor.

Note that, the user needs to specify which of the columns in `data` corresponds to the market returns using argument `mkt.name`. In the above case, `factor.names` were left out from the argument list and a pure market-timing model was fit.

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

##      EDHEC.LS.EQ SP500.TR US.10Y.TR
## HAM1       0.268    0.287   -0.2302
## HAM2       1.547   -0.195    0.0504
## HAM3       1.251    0.131    0.1437
## HAM4       1.222    0.273   -0.1391
## HAM5       1.621   -0.184    0.2712
## HAM6       1.250   -0.175   -0.1739

fit.ols$r2

##  HAM1  HAM2  HAM3  HAM4  HAM5  HAM6
## 0.501 0.514 0.657 0.413 0.232 0.564

fit.ols$resid.sd

##   HAM1   HAM2   HAM3   HAM4   HAM5   HAM6
## 0.0189 0.0253 0.0216 0.0427 0.0409 0.0161
```

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.

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

##      EDHEC.LS.EQ SP500.TR US.10Y.TR
## HAM1       0.157    0.277   -0.1635
## HAM2       1.151   -0.116   -0.0524
## HAM3       0.781    0.240    0.0500
## HAM4       1.613    0.209   -0.0829
## HAM5       1.341   -0.117    0.1920
## HAM6       1.255   -0.180   -0.1778

fit.robust$r2

##  HAM1  HAM2  HAM3  HAM4  HAM5  HAM6
## 0.313 0.231 0.461 0.316 0.240 0.470

fit.robust$resid.sd

##   HAM1   HAM2   HAM3   HAM4   HAM5   HAM6
## 0.0179 0.0202 0.0152 0.0370 0.0285 0.0156
```

Notice the lower R-squared values and smaller residual volatilities with robust regression. Figures 1 and 2 give a graphical comparison of the fitted returns for asset "HAM3" and residual volatilities from the factor model fits. Figure 1 depicts the smaller influence that the volatility of Jan 2000 has on the robust regression. (Plot options are explained later in section 4.)

```
par(mfrow=c(2,1))
plot(fit.ols, plot.single=TRUE, which=1, asset.name="HAM3")
mtext("LS", side=3)
plot(fit.robust, plot.single=TRUE, which=1, asset.name="HAM3")
mtext("Robust", side=3)
```

```
par(mfrow=c(1,2))
plot(fit.ols, which=5, xlim=c(0,0.043), sub="LS")
plot(fit.robust, which=5, xlim=c(0,0.043), sub="Robust")
```

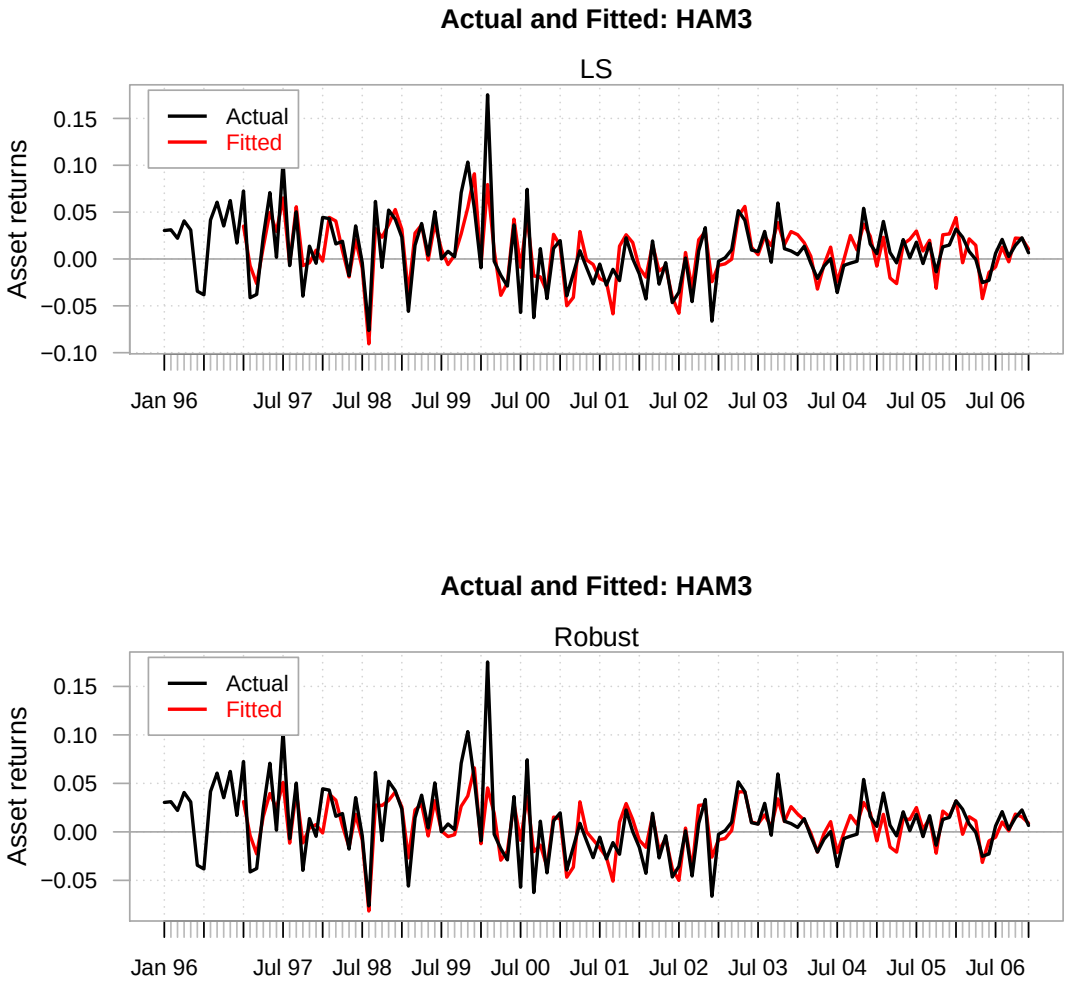

Figure 1: HAM3 Returns: LS (top) vs Robust (bottom)

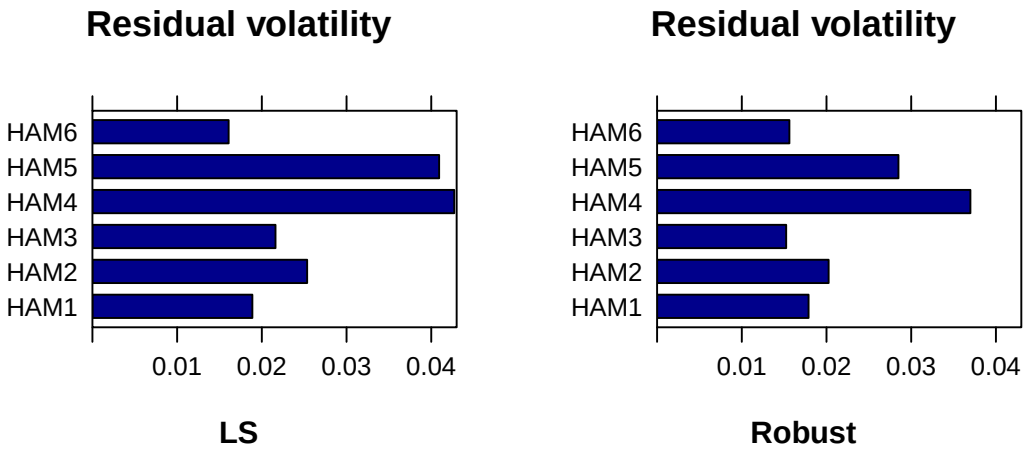

Figure 2: Residual vol: LS (left) vs Robust (right)

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.

[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.

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

##
## Call:
## fitTsfm(asset.names = asset.names, factor.names = factor.names,
##     rf.name = "US.3m.TR", data = managers, variable.selection = "lars")
##
## Model dimensions:
## Factors  Assets Periods
##       3       6     132
##
## Regression Alphas:
##         HAM1    HAM2      HAM3    HAM4     HAM5    HAM6
## [1,] 0.00623 0.00249 -0.000348 0.00347 -0.00129 0.00479
##
## Factor Betas:
##               HAM1 HAM2   HAM3  HAM4  HAM5    HAM6
## EDHEC.LS.EQ 0.114 1.11 1.1798 0.608 0.878  0.9193
## SP500.TR    0.231    . 0.0895     .     .       .
## US.10Y.TR       .    .      .     .     . -0.0744
##
## R-squared values:
##  HAM1  HAM2  HAM3  HAM4  HAM5  HAM6
## 0.501 0.506 0.657 0.405 0.221 0.564
##
## Residual Volatilities:
##   HAM1   HAM2   HAM3   HAM4   HAM5   HAM6
## 0.0189 0.0254 0.0216 0.0428 0.0409 0.0161
```

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.

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

##
## Call:
## fitTsfm(asset.names = asset.names, factor.names = factor.names,
##     rf.name = "US.3m.TR", data = managers, variable.selection = "subsets",
##     nvmin = 2, nvmax = 2)
##
## Model dimensions:
## Factors  Assets Periods
##       3       6     132
##
## Regression Alphas:
##                 HAM1    HAM2      HAM3     HAM4     HAM5    HAM6
## (Intercept) 0.00614 0.00063 -0.000903 -0.00183 -0.00346 0.00366
##
## Factor Betas:
##                HAM1   HAM2  HAM3  HAM4  HAM5   HAM6
## EDHEC.LS.EQ       .  1.545 1.245 1.228 1.294  1.221
## SP500.TR      0.372 -0.199 0.119 0.285     . -0.119
## US.10Y.TR    -0.232      .     .     . 0.338      .
##
## R-squared values:
##  HAM1  HAM2  HAM3  HAM4  HAM5  HAM6
## 0.467 0.514 0.651 0.410 0.224 0.542
##
## Residual Volatilities:
##   HAM1   HAM2   HAM3   HAM4   HAM5   HAM6
## 0.0188 0.0253 0.0217 0.0426 0.0409 0.0163
```

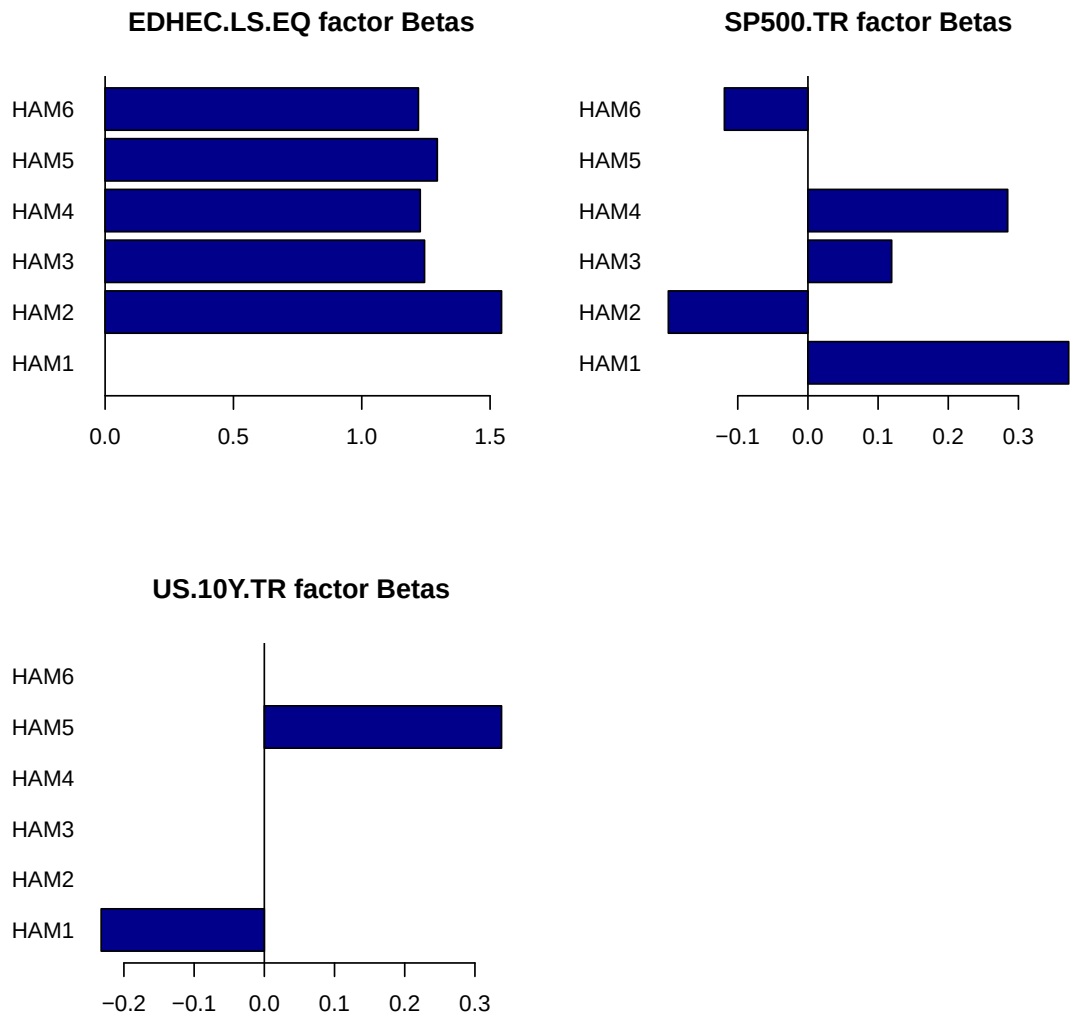

Figure 3: Factor betas: fit.sub

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 3 and 4 display the factor betas from the two fits.

```
plot(fit.sub, which=2)
```

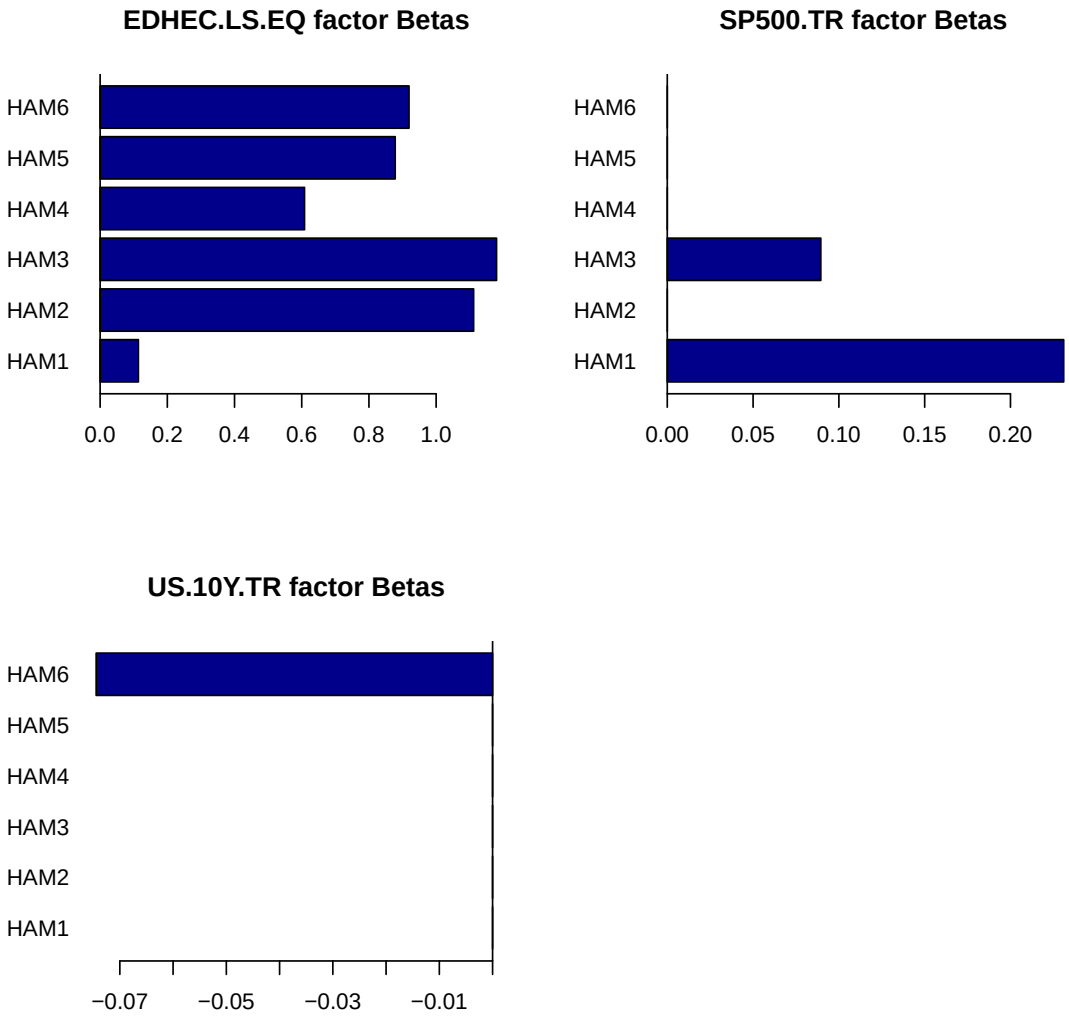

Figure 4: Factor betas: fit.lars

```
plot(fit.lars, which=2)
```

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 `fitTsfm control` for more details on the control arguments that can be passed to the different variable selection methods.

2.5 fitTsfm control

Since `fitTsfm` 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 `fitTsfm.control`. This function is meant to be used internally by `fitTsfm` 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 `fitTsfm` 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"
- `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 `fitTsfm`.

- `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.tsfm*         fitted.tsfm*       fmCov.tsfm*
##  [4] fmEsDecomp.tsfm*   fmSdDecomp.tsfm*   fmVaRDecomp.tsfm*
##  [7] plot.tsfm*         predict.tsfm*      print.tsfm*
## [10] residuals.tsfm*    summary.tsfm*
##
##    Non-visible functions are asterisked
```

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)

##      (Intercept) EDHEC.LS.EQ SP500.TR US.10Y.TR
## HAM1    0.005371       0.268    0.287   -0.2302
## HAM2    0.000512       1.547   -0.195    0.0504
## HAM3   -0.001240       1.251    0.131    0.1437
## HAM4   -0.001503       1.222    0.273   -0.1391
## HAM5   -0.004447       1.621   -0.184    0.2712
```

```
## HAM6    0.004038       1.250   -0.175   -0.1739
```

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

```
##            HAM1.return HAM1.fitted HAM1.residual
## 2006-07-31    -0.01863     0.00585     -0.024483
## 2006-08-31     0.01169     0.01151      0.000182
## 2006-09-30     0.00224     0.01063     -0.008392
## 2006-10-31     0.03889     0.01466      0.024231
## 2006-11-30     0.00740     0.01142     -0.004017
## 2006-12-31     0.00709     0.00970     -0.002605
```

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

```
##
## Call:
## fitTsfm(asset.names = asset.names, factor.names = factor.names,
##     rf.name = "US.3m.TR", data = managers, variable.selection = "subsets",
##     nvmin = 2, nvmax = 2)
##
## Factor Model Coefficients:
##
## Asset1: HAM1
## (HAC Standard Errors & T-stats)
##
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.00614    0.00181    3.40   0.0009 ***
## SP500.TR     0.37163    0.04930    7.54  7.4e-12 ***
## US.10Y.TR   -0.23242    0.07091   -3.28   0.0013 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## R-squared: 0.467, Residual Volatility: 0.0188
##
## Asset2: HAM2
## (HAC Standard Errors & T-stats)
##
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.00063    0.00239    0.26    0.792
## EDHEC.LS.EQ  1.54468    0.24583    6.28  5.9e-09 ***
## SP500.TR    -0.19897    0.10595   -1.88    0.063 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-squared: 0.514, Residual Volatility: 0.0253
##
## Asset3: HAM3
## (HAC Standard Errors & T-stats)
##
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.000903   0.001871   -0.48  0.63029
## EDHEC.LS.EQ  1.244687   0.328379    3.79  0.00024 ***
## SP500.TR     0.119303   0.124063    0.96  0.33822
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-squared: 0.651, Residual Volatility: 0.0217
##
## Asset4: HAM4
## (HAC Standard Errors & T-stats)
##
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.00183    0.00428   -0.43  0.66977
## EDHEC.LS.EQ  1.22790    0.34069    3.60  0.00046 ***
## SP500.TR     0.28467    0.12115    2.35  0.02046 *
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-squared: 0.41, Residual Volatility: 0.0426
##
## Asset5: HAM5
## (HAC Standard Errors & T-stats)
##
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.00346    0.00396   -0.87     0.38
## EDHEC.LS.EQ  1.29442    0.28326    4.57  1.9e-05 ***
## US.10Y.TR    0.33791    0.21106    1.60     0.11
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-squared: 0.224, Residual Volatility: 0.0409
##
## Asset6: HAM6
## (HAC Standard Errors & T-stats)
##
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.00366    0.00286    1.28     0.20
## EDHEC.LS.EQ  1.22084    0.21548    5.67  4.2e-07 ***
## SP500.TR    -0.11910    0.10905   -1.09     0.28
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-squared: 0.542, Residual Volatility: 0.0163
```

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 $1 \times K$ 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. Options for handling missing observations include "pairwise.complete.obs" (default), "everything", "all.obs", "complete.obs" and "na.or.complete".

```
fmCov(fit.sub)

##           HAM1     HAM2     HAM3     HAM4     HAM5     HAM6
## HAM1 0.000661 0.000257 0.000411 0.000527 0.000286 0.000231
## HAM2 0.000257 0.001297 0.000710 0.000807 0.000631 0.000545
## HAM3 0.000411 0.000710 0.001334 0.001024 0.000732 0.000601
## HAM4 0.000527 0.000807 0.001024 0.003051 0.000855 0.000689
## HAM5 0.000286 0.000631 0.000732 0.000855 0.002348 0.000529
## HAM6 0.000231 0.000545 0.000601 0.000689 0.000529 0.000720

# return correlation plot; Angular Order of the Eigenvectors
plot(fit.sub, which=7, order="AOE", method="ellipse", tl.pos = "d")
```

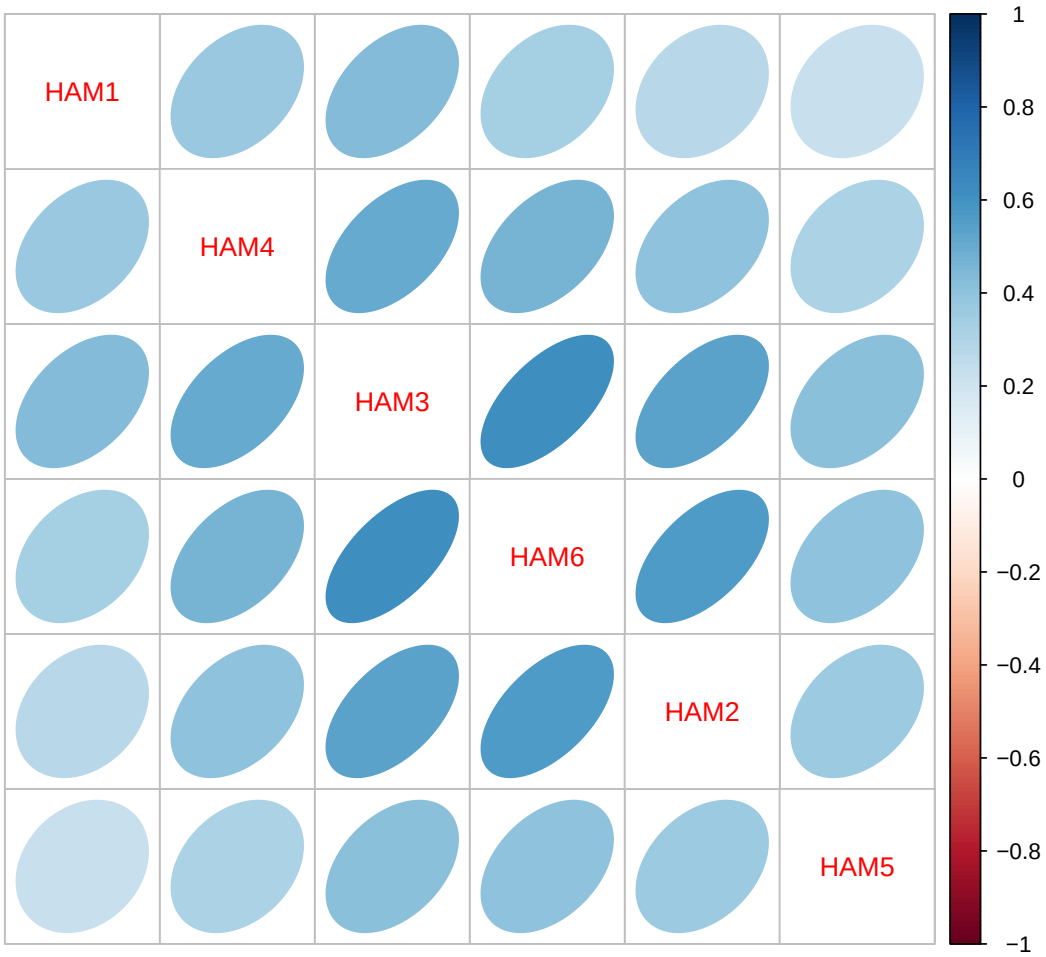

Figure 5: Factor model return correlation (pairwise complete obs)

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)):

$$R_{i,t} = \alpha_i + \boldsymbol{\beta}_i \, \mathbf{f_t} + \epsilon_{i,t} \tag{4}$$

$$= \boldsymbol{\beta}_i^* \, \mathbf{f_t^*} \tag{5}$$

where, $\boldsymbol{\beta}_i^* = (\boldsymbol{\beta}_i \, \sigma_i)$ and $\mathbf{f_t^*} = [\mathbf{f_t'} \, z_t]'$, with $z_t \sim iid(0, 1)$.

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.

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

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

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

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

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

## [1] "Sd.fm" "mSd"   "cSd"   "pcSd"

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

##   HAM1   HAM2   HAM3   HAM4   HAM5   HAM6
## 0.0257 0.0360 0.0365 0.0552 0.0485 0.0268

# get the component contributions to Sd
decomp$cSd
```

```
##      EDHEC.LS.EQ SP500.TR US.10Y.TR residuals
## HAM1      0.0000  0.01054  0.001365   0.01381
## HAM2      0.0218 -0.00354  0.000000   0.01772
## HAM3      0.0202  0.00339  0.000000   0.01291
## HAM4      0.0154  0.00689  0.000000   0.03289
## HAM5      0.0137  0.00000  0.000322   0.03447
## HAM6      0.0194 -0.00256  0.000000   0.00996

# get the marginal factor contributions to Sd
decomp$mSd

##      EDHEC.LS.EQ SP500.TR US.10Y.TR residuals
## HAM1      0.0101   0.0284 -0.005874     0.733
## HAM2      0.0141   0.0178 -0.002272     0.701
## HAM3      0.0162   0.0284 -0.002941     0.594
## HAM4      0.0126   0.0242 -0.002367     0.772
## HAM5      0.0106   0.0165  0.000953     0.843
## HAM6      0.0159   0.0215 -0.002621     0.609

# get the percentage component contributions to Sd
decomp$pcSd

##      EDHEC.LS.EQ SP500.TR US.10Y.TR residuals
## HAM1         0.0    41.00     5.309      53.7
## HAM2        60.6    -9.82     0.000      49.2
## HAM3        55.4     9.29     0.000      35.3
## HAM4        28.0    12.48     0.000      59.6
## HAM5        28.2     0.00     0.664      71.1
## HAM6        72.4    -9.54     0.000      37.1

# plot the percentage component contributions to Sd
plot(fit.sub, which=8)
```

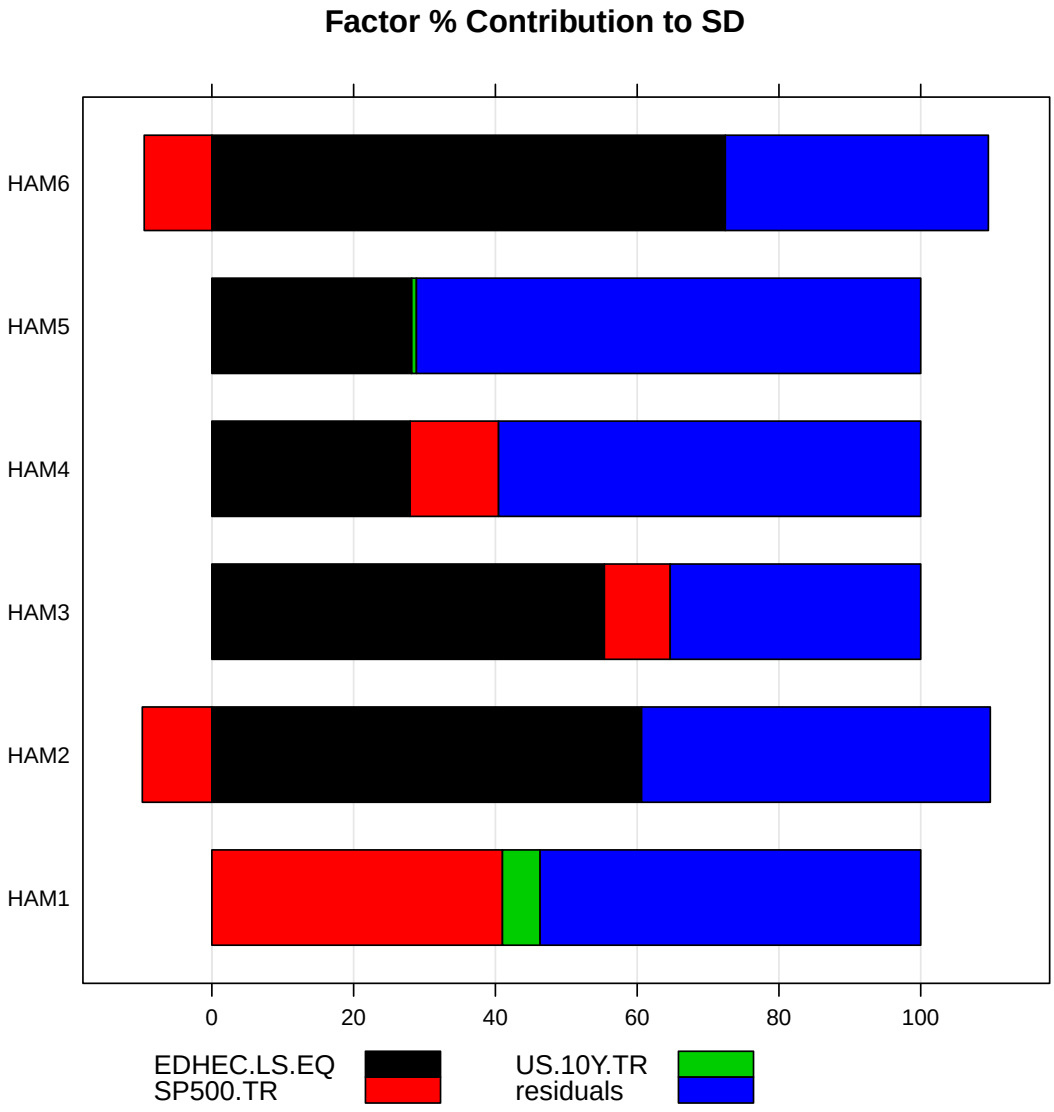

Figure 6: Percentage factor contribution to SD

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. $VaR.fm$ calculation is performed using the function `VaR` from the `PerformanceAnalytics` package. Refer to their help file for details and more options.

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

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

## [1] "VaR.fm"     "n.exceed"   "idx.exceed" "mVaR"       "cVaR"
## [6] "pcVaR"

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

##   HAM1   HAM2   HAM3   HAM4   HAM5   HAM6
## 0.0372 0.0310 0.0400 0.0848 0.0708 0.0317

# get the percentage component contributions to VaR
decomp1$pcVaR

##      EDHEC.LS.EQ SP500.TR US.10Y.TR residuals
## HAM1         0.0     49.9      3.87      46.3
## HAM2        92.3    -13.2      0.00      20.9
## HAM3        76.8     13.2      0.00      10.0
## HAM4        37.3     12.6      0.00      50.1
## HAM5        32.8      0.0      3.83      63.4
## HAM6        85.4    -10.3      0.00      24.9

# plot the percentage component contributions to VaR
plot(fit.sub, which=10)
```

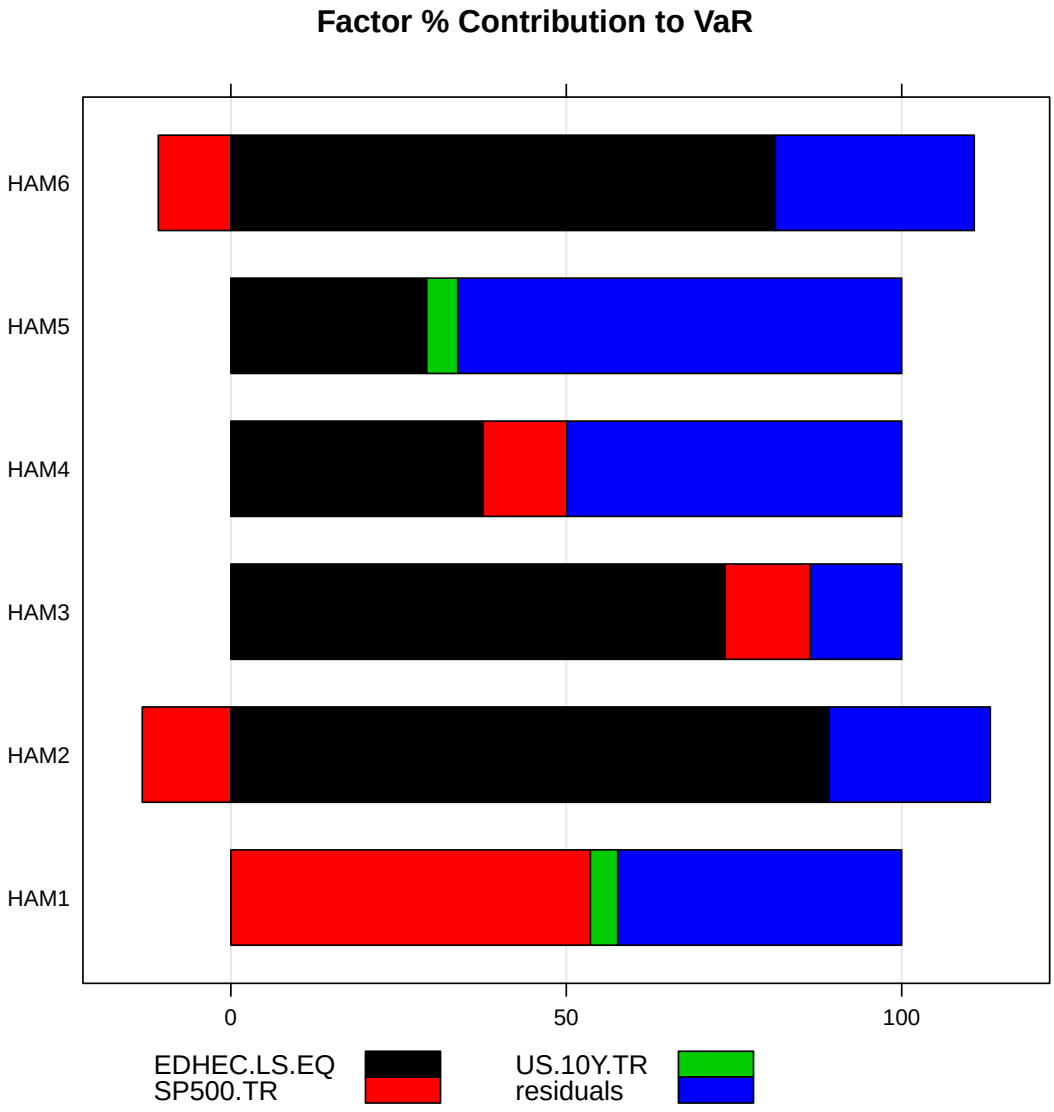

Figure 7: Percentage factor contribution to VaR

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, $VaR.fm$ calculation is performed using the function `VaR` from the `PerformanceAnalytics` package. Refer to their help file for details and more options.

`fmEsDecomp` performs this decomposition for all assets in the given factor model fit object as shown below. In this example, `method` to calculate VaR is "historical" instead of the default "modified".

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

## [1] "ES.fm"      "n.exceed"   "idx.exceed" "mES"        "cES"
## [6] "pcES"

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

##   HAM1   HAM2   HAM3   HAM4   HAM5   HAM6
## 0.0538 0.0370 0.0586 0.1153 0.1067 0.0411

# get the component contributions to Sd
decomp2$cES

##      EDHEC.LS.EQ SP500.TR US.10Y.TR residuals
## HAM1     0.00000  0.02592   0.00441    0.0235
## HAM2     0.01232 -0.00111   0.00000    0.0258
## HAM3     0.02715  0.00873   0.00000    0.0227
## HAM4     0.03576  0.02132   0.00000    0.0582
## HAM5     0.00309  0.00000  -0.00302    0.1067
## HAM6     0.02737 -0.00422   0.00000    0.0180

# get the marginal factor contributions to ES
decomp2$mES
```

```
##      EDHEC.LS.EQ SP500.TR US.10Y.TR residuals
## HAM1     0.02593  0.06975  -0.01898      1.25
## HAM2     0.00798  0.00559  -0.00658      1.02
## HAM3     0.02181  0.07315  -0.01157      1.05
## HAM4     0.02912  0.07488  -0.01364      1.37
## HAM5     0.00239  0.00771  -0.00894      2.61
## HAM6     0.02242  0.03544  -0.01181      1.10

# get the percentage component contributions to ES
decomp2$pcES

##      EDHEC.LS.EQ SP500.TR US.10Y.TR residuals
## HAM1         0.0     48.1      8.19      43.7
## HAM2        33.3     -3.0      0.00      69.7
## HAM3        46.3     14.9      0.00      38.8
## HAM4        31.0     18.5      0.00      50.5
## HAM5         2.9      0.0     -2.83      99.9
## HAM6        66.6    -10.3      0.00      43.7

# plot the percentage component contributions to ES
plot(fit.sub, which=9)
```

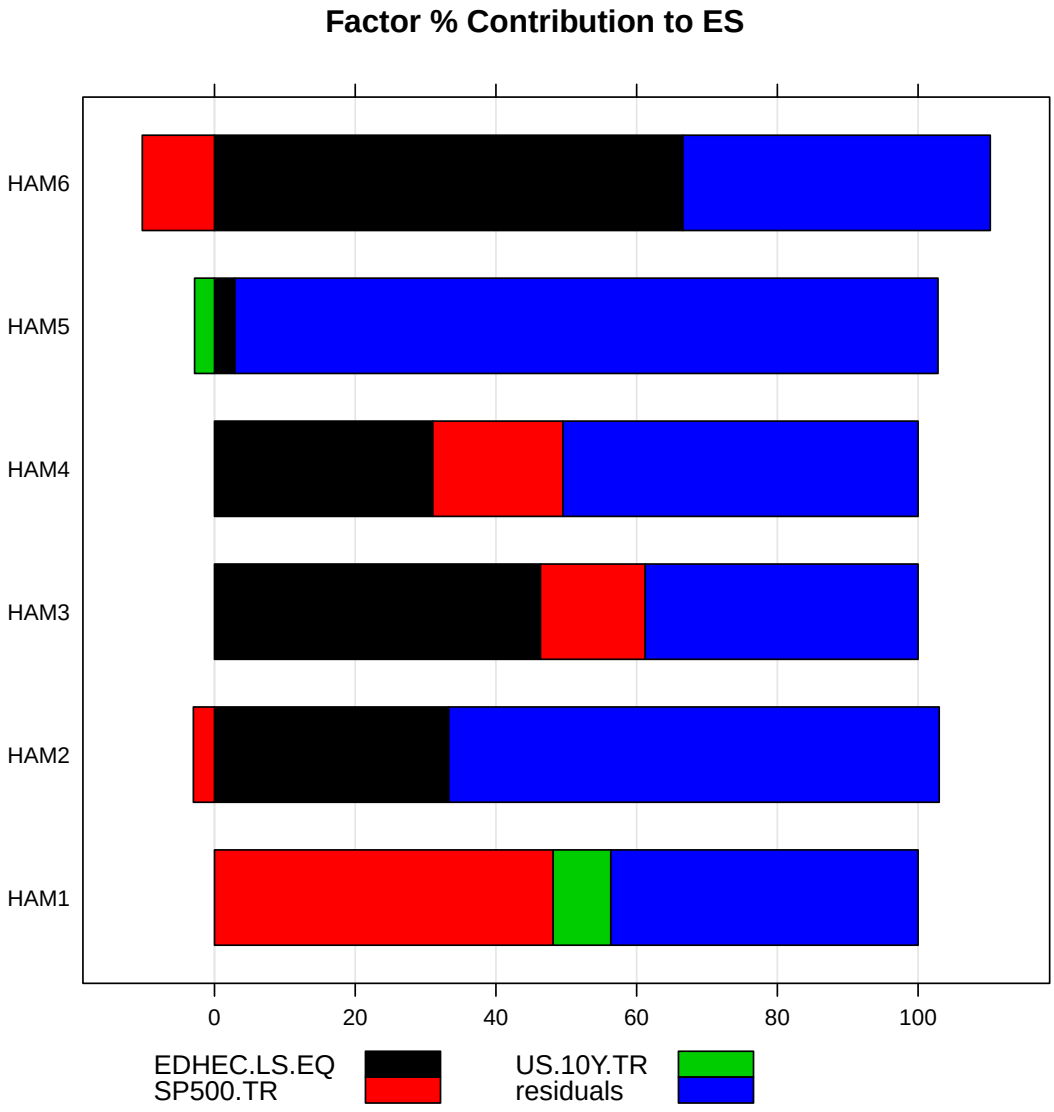

Figure 8: Percentage factor contribution to ES

4 Plot

Many types of individual asset (Figure 1) and group plots (Figures 2-8) 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, max.show=6, plot.single=FALSE, asset.name, colorset=(1:12),
      legend.loc="topleft", las=1, VaR.method="historical", ...)
```

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: Factor Model Residual Correlation
##  7: Factor Model Return Correlation
##  8: Factor Contribution to SD
##  9: Factor Contribution to ES
## 10: Factor Contribution to VaR
##
## Selection:
```

Remarks:

- For group plots, only the first `max.show` assets are plotted.
- `VaR.method` applies to group plots 9 and 10, which are factor model risk ES and VaR decompositions respectively.

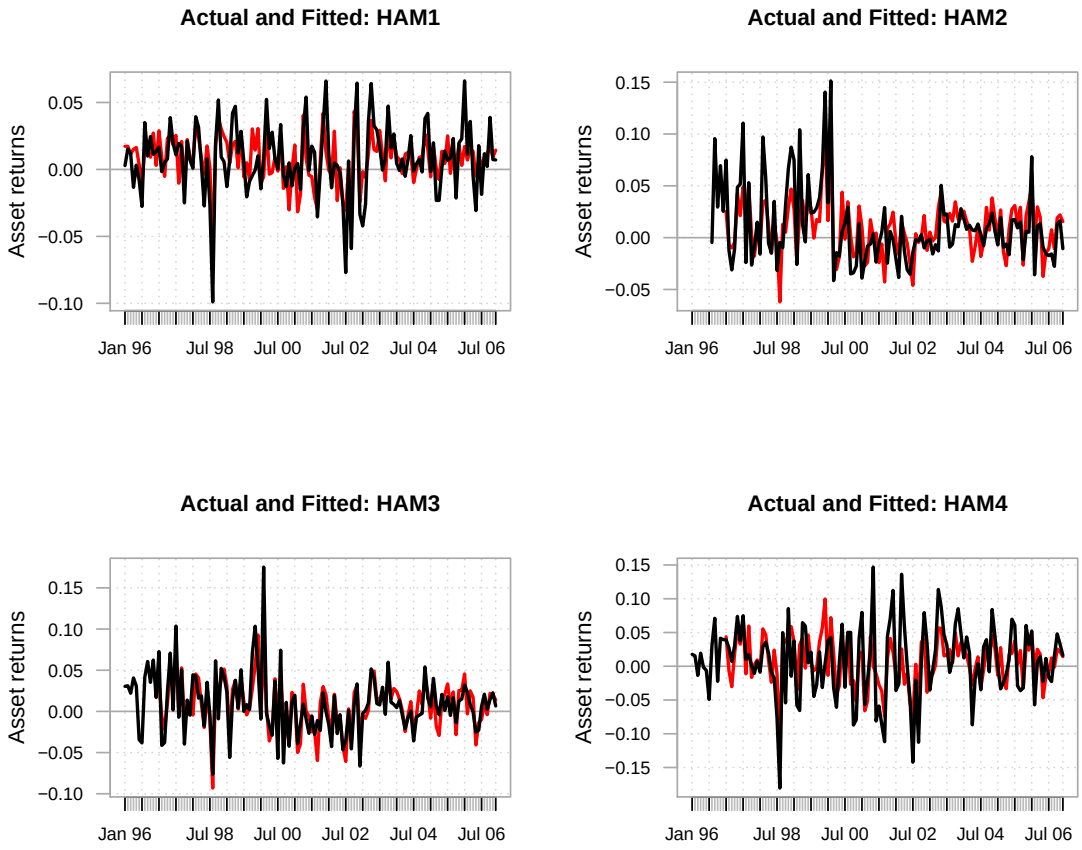

Figure 9: Actual and fitted factor model returns for the 1st 4 assets

```
# Example of a group plot: looping disabled & no. of assets displayed = 4.
plot(fit.sub, which=3, max.show=4, legend.loc=NULL)

## Displaying only the first 4 assets, since the number of assets > 'max.show'
```

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 and fitted asset returns
#  4: Sqrt. of Std. 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: Density Estimate of Residuals
# 12: Histogram of residuals with normal curve overlayed
# 13: Normal qq-plot of residuals
# 14: CUSUM test-Recursive residuals
# 15: CUSUM test-LS residuals
# 16: Recursive estimates (RE) test of LS regression coefficients
# 17: Rolling estimates over a 24-period observation window
#
# Selection:
```

Remarks:

- CUSUM plots (individual asset plot options 14, 15 and 16) are applicable only for `fit.method="LS"`.
- Rolling estimates (individual asset plot option 17) is not applicable for `variable.slection="lars"`.

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

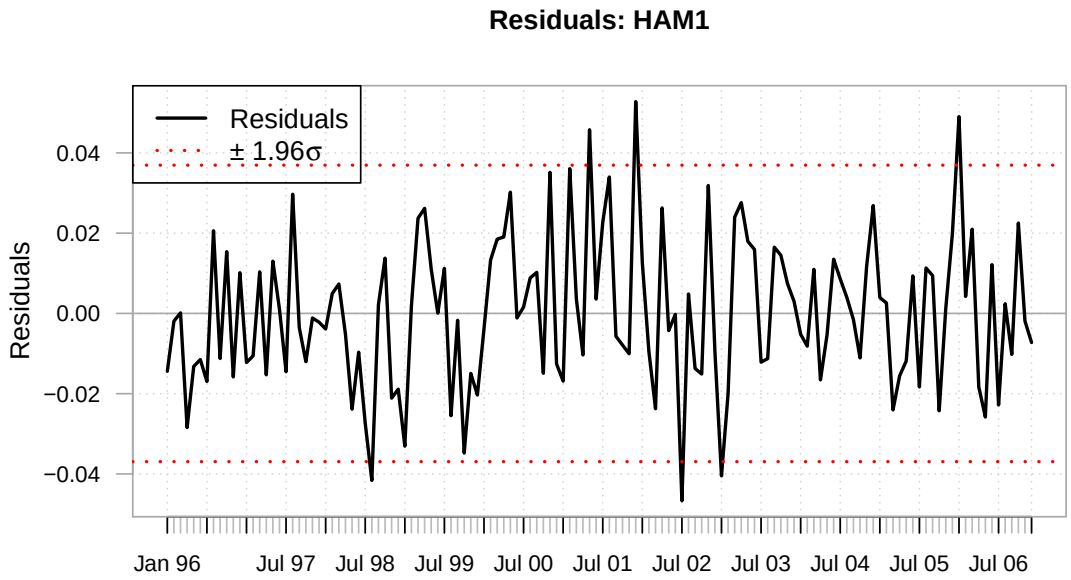

Figure 10: Time series plot of residuals with standard error bands: HAM1

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

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

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

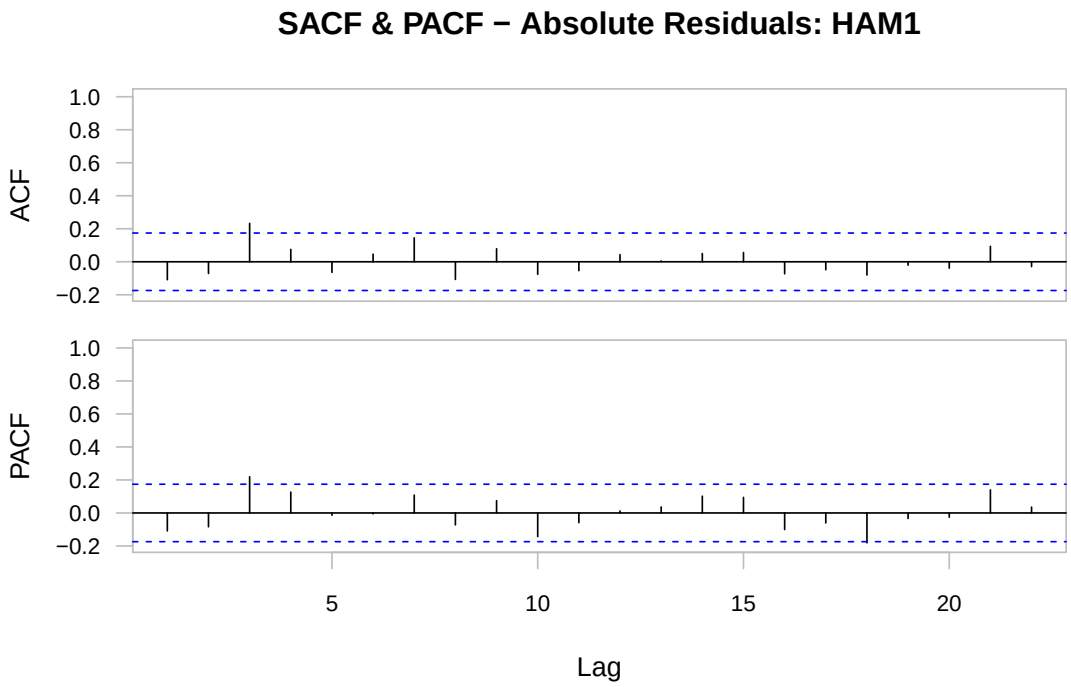

Figure 11: SACF and PACF of absolute residuals: HAM1

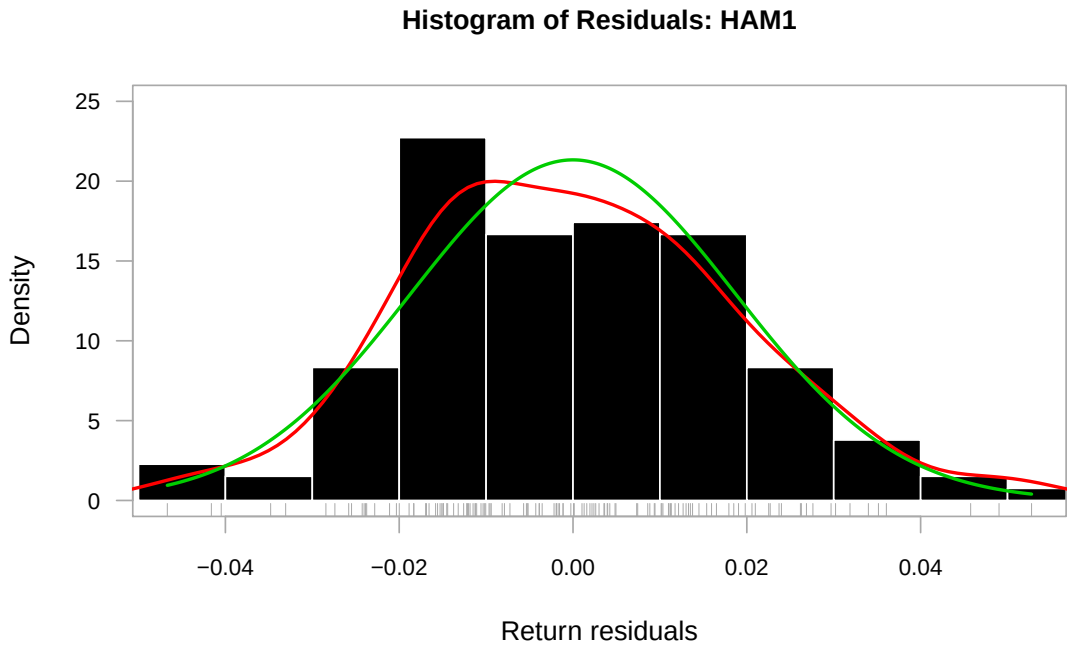

Figure 12: Histogram of residuals with normal curve overlayed for HAM1

References

N.-F. Chen, R. Roll, and S. A. Ross. Economic forces and the stock market. *Journal of business*, pages 383–403, 1986.

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

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.

W. F. Sharpe. Capital asset prices: A theory of market equilibrium under conditions of risk*. *The journal of finance*, 19(3):425–442, 1964.

J. Treynor and K. Mazuy. Can mutual funds outguess the market. *Harvard business review*, 44 (4):131–136, 1966.

E. Zivot and W. Jia-hui. Modeling financial time series with s-plus springer-verlag. 2006.